Graph Convolutional Networks for Complex Traffic Scenario Classification

Tobias Hoek

Delft University of Technology

TU Delft

KOGNIC

Graph Convolutional Networks for Complex Traffic Scenario Classification

by

Tobias Hoek

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of Master of Science, to be defended publicly on Wednesday, August 31, 2023, at 9:30.

Student number:	4592212	
Project duration:	December 1, 2022 – August 31, 2023	
Thesis committee:	Dr. H. Caesar,	TU Delft, supervisor
	Dr. C. Pek,	TU Delft
	Dr. J.F.P. Kooij	TU Delft
	A. Falkovén	Kognic, supervisor
	T. Johansson	Kognic, supervisor

TU Delft KOGNIC

Graph Convolutional Networks for Complex Traffic Scenario Classification

Tobias Hoek

Abstract

A scenario-based testing approach can reduce the time required to obtain statistically significant evidence of the safety of Automated Driving Systems (ADS). Identifying these scenarios in an automated manner is a challenging task. Most methods on scenario classification do not work for complex scenarios with diverse environments (highways, urban) and interaction with other traffic agents. This is mirrored in their approaches which model an individual vehicle in relation to its environment, but neglect the interaction between multiple vehicles (e.g. cut-ins, stationary lead vehicle). Furthermore, existing datasets lack diversity and do not have per-frame annotations to accurately learn the start and end time of a scenario.

We propose a method for complex traffic scenario classification that is able to model the interaction of a vehicle with the environment, as well as other agents. We use Graph Convolutional Networks to model spatial and temporal aspects of these scenarios. Expanding the nuScenes and Argoverse 2 driving datasets, we introduce a scenario-labeled dataset, which covers different driving environments and is annotated per frame. Training our method on this dataset, we present a promising baseline for future research on per-frame complex scenario classification.

1. Introduction

Self-driving or autonomous vehicles (AVs) have received significant attention in recent years due to their potential to revolutionize transportation. These vehicles offer a promising solution to many of the drawbacks associated with traditional commuting methods. AVs have the potential to enhance the commuting experience in terms of comfort and productivity during the ride, while also addressing societal challenges such as emissions reduction [24], traffic congestion resolution [38], and lower travel costs [14]. However, one of the most significant advantages of AVs is their potential to improve overall road safety for all traffic participants. The existing simpler automation systems for vehicles known as Advanced driver-assistance systems (ADAS) show promise in reducing traffic incidents [23] already. Ongoing research and car development aims to enhance traffic safety through higher-level Automated Driving Systems (ADS). To ensure superior performance of ADS compared to human drivers, proper development and testing are crucial. However, conducting test drives in real traffic poses safety risks and requires an impractical amount of driving miles to gather statistically significant evidence. According to [15], obtaining such evidence would require 275 million failure-free miles, given the rarity of critical situations in regular traffic scenarios. This timeframe is unfeasible for the production of AVs using regular driving speeds. An alternative solution involves conducting smaller test drives where critical situations are simulated. By leveraging these simulated scenarios, it is possible to obtain the same statistical evidence of ADS performance in critical situations within a more manageable timeframe [31, 29, 28]. To keep pace with the rapid development of these systems, multiple countries are updating their legislation for the acceptance of AVs. A clear example of such a change in legislation is the regulation that is proposed by the EU (EU2019/2144 [8]). This regulation establishes type-approval requirements for vehicles and components, emphasizing safety for all, including occupants and vulnerable road users. While existing regulations covered ADAS and ADS evaluations, advancements towards SAE level 4 self-driving vehicles and effective scenario-based testing led to adding critical scenarios to mandatory acceptance tests. These scenarios play a vital role in gathering the required statistical evidence to validate the safety performance necessary for regulatory approval. Consequently, it becomes crucial to determine if your system is ready for these particular scenarios. Detecting such scenarios within your dataset not only reveals insights about dataset quality but also streamlines both validation and training processes. This could be done with the use of a classification algorithm for scenarios. However, this gives rise to two issues. Firstly, the most current scenario classification methods target simpler situations. These situations involve either a single vehicle or the vehicle's interaction with its surroundings. However, the few existing approaches dealing with complex scenarios perform classification per-agent instead of per frame (where each frame is a snapshot at a certain interval in the time dimension). Secondly, no comprehensive publicly available dataset exists that provides per-frame labeling of scenarios. To address these issues we present the following contributions:

- We designed a supervised scenario classification approach that is able to classify complex ego-centered scenarios, which are not constrained to specific environments (e.g. highways) and that requires modeling the relation between agents, as well as agents and the environment based on their position, direction, and velocity.
- We extend Graph Convolutional Networks to incorporate the latest advances for representing agent-agent and agent-environment interaction, as well as temporal aggregation.
- We created a scenario classification dataset by hand-selecting scenarios and annotating every frame. This is made as an extension of the publicly available datasets nuScenes [5] and Argoverse 2 [42].
- We evaluate our method and related works on our dataset and compare it against baselines, creating a reference approach for future work on our dataset.

2. Related work

The literature on scenario classification is limited. Additionally looking into closely related tasks such as maneuver detection or trajectory prediction can be insightful. These methodologies have in common that there are challenges in the spatial and in the temporal aspect. The existing works will be elaborated accordingly.

2.1. Scenario classification

Few existing works focus on scenario classification. A method by [30] is based on rules and detects lane changes of surrounding vehicles. It relies on distance measurements between the ego vehicle, other vehicles and environmental features like lane markings. However, this rule-based method will show its limitations when trying to classify more complex scenarios involving multiple cars or strong variations within a specific scenario.

[4] proposed a method that also uses the sensor measurements taken from various sensors of the car, such as the inertial measurement unit (IMU) or distance measures to lanes or other cars. These raw measurements are used as input channels for their CNN. This approach shows more promise in terms of scalability compared to the rule-based approach, but it still has limitations as it does not consider the presence of other vehicles. In addition to the sensor measurements, [33] uses dashcam footage within their pipeline. This footage is merged into one feature block with the help of intermediate object detection steps. This work is limited because it is solely based on the detection of the cars and does not use information such as lane markings.

Spatial aggregation. There are also models that use a more comprehensive spatial aggregation. These works use a form of intermediate representation. Methods that employ a grid as an intermediate representation are proposed by [12] and [3]. The former suggests a grid representation that incorporates occupancy and velocity. The latter also developed a grid representation, but in contrast, this grid is based on polar coordinates and the velocity relative to the ego vehicle. However, the main limitation of both these methods is their inability to incorporate environment information (e.g. road markings or centerlines of driveable road). In general, grids have limitations. Low resolutions cause rasterization artifacts and hinder the accurate shape depiction. Raising resolution may alleviate these issues but will enlarge the grid with excessive and unnecessary information.

Maintaining high resolution but a small input size, which has a positive effect on computational efficiency, is addressed via graphs in [27], which is an approach designed for vehicle behavior classification. In this work, the graph encodes agent locations and points sampled on lane markings as vertices. Using a Graph Convolutional Network [18] (GCN), the relation between all these vertices is processed for further steps. This model can classify scenarios involving actor-environment relationships and relationships among various actors. However, it lacks critical information about agent direction and velocity, which is essential for distinguishing scenarios in diverse situations, such as the contrast between highway and urban driving settings. Additionally, their method focuses on actions of all agents, rather than actions involving or around the ego vehicle. Several works use GCNs for trajectory prediction models. In [32, 20, 21], a comparable graph input approach is employed. [32] differs from the other two because the edge weights are normalized based on the distance between agents. The convolution in LaneGCN [22] differs from these three works because it uses dilated convolution [43] between the graph layers for a larger receptive field. LaneGCN uses multiple different GCNs based on the directional relations of the selected waypoints. Our work differs from LaneGCN in terms of how the GCN is used. They apply GCN solely to static map data, excluding agents. Their temporal focus is on initial agent trajectories, ignoring map evolution and agent-map relationships using GCN. Our approach integrates both aspects, leveraging the temporal evolution of the map and usage of GCN for agent-map and agent-agent interactions.

Temporal aggregation. Several methods are used in literature to encode the temporal aspect of the scenario. Some methods use Recurrent Neural Networks, e.g. LSTMs [27, 20, 4, 16, 12] or GRUs [9]. These methods can suffer from training inefficiency, slow computational speed, and are prone to overfitting for small datasets due to their large number of parameters [34]. A newer approach is the use of attention mechanisms [37]. This is used in [33] or

in combination with LSTMs in [27]. These attention models show very promising results on temporal data, although they also increase complexity significantly and require large amounts of data. A simpler alternative involves applying a conventional CNN across the temporal dimension. On smaller datasets used for scenario classification tasks, this shows good results either by performing this convolution on crafted or learned features [32, 22, 12, 3] or merged deeper within the model where the consecutive CNNS are alternately on the spatial and the temporal aspect [21].

2.2. Datasets

Numerous datasets have been proposed for autonomous vehicle perception [10, 5, 42, 35], prediction [13, 5, 42] and planning [6, 1]. Unfortunately, the situation differs for the scenario classification task. For this task, real-world traffic scenarios are categorized into predefined classes per interval of frequency f. Existing datasets for scenario classification use either simulated data [3, 9] or data obtained in limited environments, e.g. only highway data [30, 26]. Furthermore, many datasets are not publicly available [12, 4]. While [6] offers information about scenarios, they label entire sequences as scenarios, which is not suitable for precise scenario classification. Instead, we label individual frames. Furthermore, their work is auto-labeled and manually reviewed to guarantee high precision. In contrast, we manually reviewed two datasets to also guarantee a high recall. This enables us to phrase scenario classification as a multi-class classification problem.

3. Dataset

In order to develop and assess a scenario classification technique, a corresponding dataset is essential. Given the absence of an existing or accessible one, we generated our own dataset. The process for creating this dataset is outlined in this section.

3.1. Scenario definition

According to [36], a scenario defines as follows:

> *A* scenario depicts the temporal evolution between scenes within a sequence, starting with an initial scene and covering a specified duration.

Here a scene is defined as:

> *A* snapshot of the environment, encompassing scenery, dynamic elements, actors' self-representations, and entity relationships.

To create a list of relevant scenarios, we start from the scenarios proposed in the EU type-approval regulation [8, 7] and remove scenarios that cannot be detected in public datasets. Examples of these removed scenarios are collision avoidance, emergency brake scenarios, and specific scenarios such as blocking toll gates. Finally, we select 8 scenario categories (Tab. 1). The frequency and duration statistics in this table correspond to our dataset. Further details on this will be provided in the scenario extraction paragraph.

#	Scenario	Frequency	Duration	
			Mean(s)	StDev (s)
0	No scenario	-	-	-
1	Cut-in	77	4.7	1.8
2	Stationary vehicle in lane	42	8.1	3.8
3	Ego lane change right	47	4.3	1.5
4	Ego lane change left	43	4.6	1.2
5	Right turn at crossing	136	7.0	2.6
6	Left turn at crossing	117	6.7	2.4
7	Straight ahead at crossing	175	5.1	2.0

Table 1. Overview of the selected scenarios, their frequency within the dataset and mean duration and corresponding standard deviation (SD).

Here a *cut-in* (1) represents a scenario where another vehicle changes lanes into the ego vehicles lane. *Stationary vehicle in lane* (2) is a variation on a cut-out scenario, where a stationary vehicle is in the ego lane, such that the ego vehicle has to either brake or perform an obstacle avoidance maneuver. 3 and 4 are ego lane changes in both directions. $5, 6, 7$ represent the actions at crossings. *No scenario* (0) indicates all other driving scenarios, including lane keeping and more complex maneuvers not included in the list. This list of scenarios is mutually exclusive and complete, thus making it suitable for the scenario classification task.

3.2. Dataset creation

After defining the scenarios of interest we created the dataset based on nuScenes and Argoverse 2. This process involved three main phases. Initially, data was chosen and labeled. Then, a preprocessing step aligned the differing frequencies between the two datasets. Finally, to ensure a better balance and eliminate less relevant timeframes, we removed unnecessary timesteps in the dataset's final stage.

Data selection and labeling. The traffic information used for this dataset is obtained from existing public driving datasets, specifically nuScenes [5] and Argoverse 2 [41]. In the selection phase, all the front-camera videos in the datasets are inspected manually. A sequence is selected for the dataset if it includes at least one of the explicitly defined scenario classes (classes $1-7$) from Sec. 3.1). Meaning that sequences with only the presence of class 0 are not taken into account. This mitigates the extreme class imbalance inherent in the task, as class 0 dominates the datasets. These class 0 timeframes around labeled scenarios are taken into account resulting in a sufficient number of occurrences within the dataset. For each keyframe in the dataset, anno-

tated with bounding boxes for each agent, we label the current scenario. This results in 312 sequences of 20 seconds obtained from nuScenes, and 253 sequences of 15 seconds from Argoverse, or a total of 565 sequences.

Frequency alignment. We use nuScenes and Argoverse 2, which are annotated at 2Hz and 10Hz respectively. We use linear interpolation to bring both datasets to the same frequency (4Hz). For Argoverse, this means that we interpolate between every 2nd and 3rd keyframe. This enables us to train the same scenario classification model on both.

Scenario extraction. Instead of using complete sequences from the original datasets, we extract shorter sequences for each scenario. Our interest extends beyond classification; we also need to determine precise scenario start and end times, which requires temporal context. We obtain this by cutting out all scenarios (except *no scenario*) with a random amount of timesteps before and after each scenario. This is limited to a maximum of 8, if available in the original sequence, and a minimum of zero. This procedure has the advantage that it further reduces class imbalance since most of the frames in the full sequences are labeled as *no scenario*. This results in 652 sequences of varying lengths, since the full sequences may contain multiple scenarios. Sequence durations range from 2 to 23 seconds, with an average of 8 seconds. All data is sampled at 4Hz. The distribution between classes can be seen in Tab. 1. We notice that the more complex scenarios (1,2,3,4) occur less often than the crossing related scenarios. The standard deviation is notably significant compared to the mean duration. This is unsurprising, as scenarios can be executed at different speeds, leading to a broad range of durations.

4. Method

Our method utilizes Graph Convolutional Networks to classify complex traffic scenarios, capturing agent-agent and agent-environment interactions. Shown in Fig. 1, our model takes graph inputs, comprising three core components: spatial aggregation, temporal aggregation, and a classification head producing frame-wise class probabilities.

4.1. Graph construction

We represent each frame of the traffic scenario by a graph. This graph is given by $G_t = \{V_t, A_t\}$, where t is the frame index with $t \in \{1, ..., T\}$ and T represents the sequence length. For each frame, V_t denotes the graph's vertices with $V \in \mathbb{R}^{N \times c}$. Here is N the number of vertices present in the graph and c represents the feature channels. In this case $V_i = (x, y, \phi, v)$ and $c = 4$. Here x, y represents the bird's eye view location of the vertex, and ϕ is the heading angle both in an ego-centered frame. v is the velocity in m/s of the agent represented by the i^{th} index. See Fig. 2 for a visual explanation. For our method, the nuScenes x, y, and ϕ had to be transformed to the ego frame. In Argoverse 2 this was already the case.

Waypoints are added to the graph similarly, at 3-meter intervals along the centerlines of the driveable road. These vertices have zero velocity such that $V_i = (x, y, \phi, 0)$. ϕ is the driving direction of the road segment at this particular waypoint. The model will learn to distinguish road waypoints and vehicles in a later stage. The edges between vertices are denoted by adjacency matrices $A_{ti} \in \mathbb{R}^{N \times N}$. Here i represents 5 different adjacency matrices employed to learn diverse relationships. The first is A_{suc}, which covers relations between waypoints and it successive waypoints. A_{pre} is the same for preceding waypoints. A_{W2A} is the connection between waypoints and agents (excluding ego). A_{E2W} covers the relation between the ego-vehicle and the waypoints, and at last A_{E2A} is between the ego-vehicle and the other agents.

Each adjacency matrix is applied in different stages outlined in Sec. 4.2.1 and 4.2.2. To illustrate, we show how the graph G_t for a scenario is constructed in Fig.2 and Eq.1. These illustrations provide insight into the creation of V_t and an Adjacency matrix respectively. The depicted relation is A_{E2A} in this matrix, which includes self-connections via the addition of the identity matrix. Without self-connections, only the neighboring vertices are taken into account in the GCN, not the vertice itself.

$$A + I = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & \cdots & 0 \\ 1 & 1 & 0 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 1 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix}_{N \times N} \tag{1}$$

4.2. Spatial aggregation

Spatial feature extraction occurs across three stages, each stage is based on a different relation, such that the model can learn whether a vertex is a waypoint or an agent. The first one learns the spatial aspect of the map data and their relation, the second one does the same for all the other agents. The last stage is where the relation between the environmental and agent features and the ego vehicle is learned. Every stage is built upon a GCN [18]. The matrix operations required to compute the new hidden layer are as follows:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \tag{2}$$

To include self-loops, $\tilde{A}_t$ is obtained by adding the identity matrix I to A_t. Here A is the Adjacency corresponding to the stage. The diagonal degree matrix $\tilde{D}$ is employed to

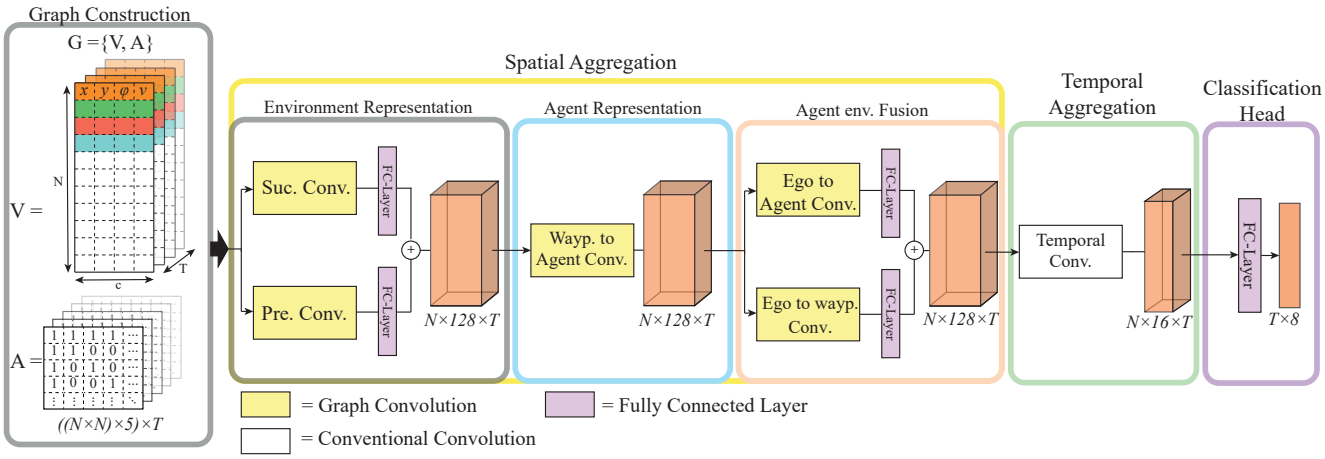

Figure 1. The pipeline of our proposed scenario classification method. The titles of the outer boxes match the sections where they are explained in more detail.

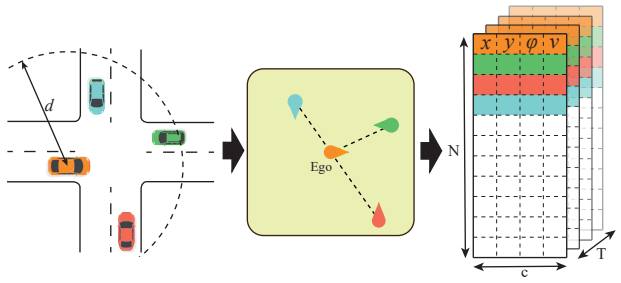

Figure 2. Forming of the vertices V out of the graph G based on a traffic scenario at a crossing consisting T frames. For illustration purposes the waypoints are neglected in this figure

calculate the average of neighboring vertices. H denotes the feature matrix prior to convolution, and W represents the trained weights. Finally, σ represents an activation function, ReLu [11] in this case.

4.2.1 Environment representation

We represent the centerline of each drivable lane on the road as a static vertex in the graph. ϕ is the direction pointing towards its successive waypoint. Compared to the agents $v = 0$ because a waypoint is static. Such that $V_i = (x, y, \phi, 0)$. The spatial dependencies of these waypoints are extracted in two parallel convolution blocks. See the environment representation part of Fig. 1. The objective of these two blocks is to learn the directional relation between the centerline points.

The adjacency matrices used in these two steps are $\{A_t\}_{i \in \{suc,pre\}}$, (successive, preceding). A_{suc} is obtained by using directional connections between a centerline waypoint and its successive point. Since the waypoints of lane segments are ordered from start to end. The successive adjacency for this specific segment is obtained by shifting this identity matrix one place to the right. The connection is now between a vertex and its successive vertex. A_{suc} is assembled by combining these segment adjacency blocks into a single matrix. Extra connections are added between the end of one segment with the start of its succeeding segment. If a lane segment is two or more successive lane segments (e.g. a fork crossing), connections to the first points of both segments are added. A_{pre} is constructed likewise but in the opposite way as A_{suc}. Each of the parallel graph convolution blocks consists of 4 layers of graph convolution followed by a linear layer. The outcomes of both blocks are summed together and fed through a fully connected layer before passing to the next step. This approach is inspired by the MapNet part of LaneGCN. [22]. For simplicity, we don't use the relations between the waypoint and their left and right neighbors, which is the case in LaneGCN.

4.2.2 Agent representation

As mentioned earlier, agents are represented as graph vertices $V_i = (x, y, \phi, v)$. The relationships among all agents and the ego vehicle are encoded in the Agent-Environment fusion part of Fig. 1. A graph convolution using A_{W2V} (Waypoint to vehicles) precedes this step. A_{W2V} captures relations between vehicles (excluding the ego vehicle) and environment features from Sec. 4.2.1 if they are within the distance threshold of $d = 30m$. To limit the first layer of GCN to cars in the direct environment. The purpose of this operation is to "update" the spatial information of the other vehicles according to their relation with the environment. This is done such that the relation between the agents and the environment is taken into account when modeling the relation between both separate parts and the ego vehicle.

Next, a GCN facilitated by A_{E2V} is applied between the ego vehicle and the features of other vehicles, obtained in the previous step. A_{E2V} is produced by connecting the ego vehicle to all other vehicles within $d = 30m$. These connections are unweighted such that the model can learn the importance weights of the connections themselves. This block consists of two GCN layers, fewer than in the environment representation, as the lower density of vertices (there are fewer agents than waypoints) makes the required receptive field achievable after just two layers.

4.2.3 Agent environment fusion

First, the features of the environment and the agents are merged. This is done in the block running in parallel with the agent representation block mentioned in Sec. 4.2.2. This parallel structure allows the model to learn features simultaneously using both environmental and agent information, while still preserving their distinctiveness. The relationship between the ego vehicle and environmental features from Sec. 4.2.1 is established using A_{E2E} (Ego to Environment). A_{E2E} encapsulates ego vehicle to waypoint unidirectional connections within $d = 30m$. These connections remain unweighted, enabling the model to learn their individual importance. The output of the GCN blocks is fed through a fully connected layer separately before the second stage of the fusion process. In the second part of the feature fusion process, the outputs are summed and fed through a fully connected layer to generate the final spatial encoding.

4.3. Temporal aggregation

As defined in Sec. 3.1, scenarios describe the temporal development of a scene. Thus scenario classification cannot depend solely on spatial data. Graph evolution over time must be considered. Since we are interested in short-term scenarios (8s on average), we use CNNs for temporal aggregation. All the spatial information is captured by the aforementioned blocks. The input of the CNN becomes of shape $H \in \mathbb{R}^{N \times F \times T}$. Here, F represents the number of channels of the features learned from the previous step, T denotes the number of frames, and N the number of vertices in the graph. A convolutional kernel with dimensions $1 \times F \times Q$ is applied to slide over this input along the T dimensions to learn the temporal dependencies. We use dilated convolution [43] in the temporal dimension for a larger receptive field without using too many layers. The input is padded to maintain the same output size. It is important to note that due to the kernel's convolution over multiple timeframes, which also include future timeframes, the model is restricted to performing offline predictions exclusively.

4.4. Classification head

The last stage of the model consists of a fully connected layer that outputs the class probability logits for every frame of the temporal window that is observed, in the shape $T \times n_{classes}$ ($= 8$ in our case). A softmax function is used to obtain class probabilities. The class with the highest probability is selected as the final prediction at frame t, which gives a set of predictions $Y = (c_1, ..., c_T)$, where T is the sequence length, as output.

4.5. Loss function

The network is trained by minimalizing the common Cross-Entropy Loss for n classes:

$$L_{CE} = -\sum_{i=1}^{n} y_i \log \hat{y}_i \tag{3}$$

Where $\hat{y}_i$ is the softmax probability for the i^{th} class and y_i is 1 if the class label i is the correct ground truth label or 0 if this is not the case.

5. Implementation details

The model primarily uses PyTorch and PyTorch Geometric (PYG) for GCN implementation and efficient graph handling. Training occurs on an NVIDIA Titan RTX GPU. To enhance computational efficiency, sparse form adjacency matrices are employed, using two indices for connections rather than dense matrices.

5.1. Spatial feature extractor

In Sec. 4.2.1 a block with 4 graph convolution layers is detailed, featuring 4 layers and outputs of 16, 64, 128, and 128 channels. The agent representation block contains two graph convolution layers with an output feature dimension of 128. Layer normalization and ReLU activation are applied GCN layer. The Agent-environment fusion block, with two graph convolution layers, retains 128 dimensions. In all three parts after every GCN layer, Layer Normalization [2] and Rectified Linear Unit (ReLU) [11] are applied.

5.2. Temporal Aggregation

The temporal feature extractor is composed of four CNN layers. The first layer reduces the feature dimensions from 128 to 16. The next two layers maintain 16 feature channels, and each uses a 1×3 kernel. In the first three layers, asymmetrical padding of 1, 2, and 4 is applied in the time dimension, respectively. Zero padding is used in the vertice dimension to preserve the same dimensions. The last convolutional layer uses a kernel size of 1×7 with a padding of 3 in the time dimension only. This is done for smoothing the predictions. After each convolutional layer a Scaled Exponential Linear Unit (SELU) [19] is applied.

5.3. Training process

The model is trained for 25 epochs using the Adam optimizer [17]. The learning rate is initiated at 1×10^{-4} and decays with a factor of 0.1 after epochs 8,14 and 18. Class weights are used to prevent the effects of class imbalance on the classification output. The class weights are as follows:

$$W_i = \frac{N_{samples}}{n_{classes} \times n_{samples,i}} \tag{4}$$

Where W_i is the weight for specific class i, $N_{samples}$ is the total amount of samples, $n_{classes}$ the total amount of classes, and $n_{samples,i}$ the amount of samples labeled as i.

6. Experiments

The proposed model's performance assessment is evaluated in three steps. First, we compare it to simpler scenario classification models to understand the impact of our model's elements. Then, we perform error analysis with an Error distribution diagram for the top-performing model. Finally, we assess per-class performance.

6.1. Ablation study

The metric used to compare different versions of the models is the area under the precision-recall curve (PR-AUC). This metric is advantageous because it focuses on identifying positives, rather than attempting to balance negatives, without the need to fine-tune a decision threshold. PR-AUC is also well-suited for use on imbalanced datasets. The average $\overline{PR-AUC}$ is calculated by finding the PR-AUC of every class first using a one-versus-all strategy. We conducted an ablation study to assess the significance of each component of the model. The findings are summarized in Tab. 2. The full model (as in Fig. 1) outperforms all other variations. Residual connections over the main blocks of Fig. 1 perform worse. Introducing weighted adjacency matrices, where connections within the adjacency matrices reflect the reciprocal of the distance (d^{-1}) such that the weight of closer vehicles is larger. This weighting leads to performance suppression.

The importance of map data becomes evident when examining the results of the experiment in which the map data is removed. Substituting the temporal aggregation method with an LSTM instead of a convolution results in poorer performance compared to the full model, but the inclusion of residual connections enhances performance in this case. A model with the same spatial encoding as the full model, but without any temporal aggregation is shown as "No temp. aggregation". The low PR-AUC shows the importance of temporal aggregation. Furthermore, the removal of ego convolution from Sec 4.2.2 results in a substantial performance drop, although it still performs better than the model lacking both ego convolution and map convolution. The worst model is the baseline, comprising of a single GCN applied across all available vertices, followed by a CNN in the temporal dimension. This architecture fails to capture the distinctions between waypoints and agents, leading to a significant performance decline.

6.2. Error analysis

Continuous sequences present various challenges, such as varying sequence lengths, potential merging or fragmentation of scenarios, and fuzzy scenario boundaries that are difficult to determine even for humans. To gain a better understanding of the model's predictions, we proceeded to conduct more comprehensive testing on the model that exhibited the best results in the previous section. The initial step involved generating the Error Distribution Diagram (EDD) [40]. See Fig 3. The EDD breaks down False-Positives (FP) and False-Negatives (FN) into multiple categories [39]. For FP, we consider three subcategories:

1. Overfill: The prediction extends beyond the ground truth boundary of the scenario.
2. Merge: The prediction combines two separate scenarios of the same class into one.
3. Insertion: The model predicts a scenario where no scenario is actually happening.

For FN, we have three subcategories:

1. Underfill: The prediction does not cover the entire ground truth of a scenario.
2. Fragmenting: The model splits the scenario into pieces.
3. Deletion: The prediction fails to detect a scenario.

For multi-class classification, we distinguish between different cases when an FN classification occurs [25]. Underfill is divided into two categories: substitute underfill, where the underfill error is replaced by another class, and normal underfill, where it is replaced by 0. Similarly, fragmentation has substitute fragmentation and normal fragmentation. When a boundary lies between two non-zero scenarios, the underfill-overfill error option also comes into play. The occurrence percentages of these error subcategories are visualized in the EDD. *Overfill*, *underfill* and *underfill-overfill* are placed above the serious error line because these are mistakes that are inevitable considering the fuzzy boundaries of the scenarios beginning and end.

6.3. Per-class performance

In addition to comparing different model setups using PR-AUC, we offer a more intuitive metric for each class. Tab. 3 shows class prediction accuracy per dataset and the class distribution in the training data. Classes 1 and 2 show

Test setup	Map conv. (Sec. 4.2.1)	Ego conv. (Sec. 4.2.2)	Residual conn.	Map data	Weighted adj.	Temp. agg.	$\overline{PR-AUC}$
Baseline				✓		Conv.	38.9
Baseline + ego conv.		✓		✓		Conv.	42.7
Baseline + map conv.	✓			✓		Conv.	47.2
No temp. aggregation	✓	✓		✓		None	33.0
LSTM	✓	✓		✓		LSTM	43.1
LSTM + res. conn.	✓	✓	✓	✓		LSTM	50.4
Full model - map data	✓	✓				Conv.	29.8
Full model + weight. adj.	✓	✓		✓	✓	Conv.	52.7
Full model + res. conn.	✓	✓	✓	✓		Conv.	56.0
Full model	✓	✓		✓		Conv.	**58.4**

Table 2. Overview of different trained models based on different ablations in the model, showing the importance of each part. The map convolution refers to the part described in Sec 4.2.1, and the Ego convolution refers to the block in Sec. 4.2.2 The differences between each row of the table are explained in Sec. 6.1.

#	Scenario	nuScenes		Argoverse 2		Total
		Acc.	Occ.	Acc.	Occ.	Acc.
0	No scenario	60.0	-	64.7	-	62.3
1	Cut-in	48.4	11	48.7	66	48.2
2	Stationary vehicle in lane	48.1	34	-	8	48.2
3	Ego lane change right	66.7	29	93.8	18	70.8
4	Ego lane change left	67.9	27	42.4	16	52.2
5	Right turn at crossing	60.0	93	50.0	43	56.8
6	Left turn at crossing	73.8	78	79.4	39	75.3
7	Straight ahead at crossing	54.5	74	58.4	101	56.5

Table 3. Validation accuracy of each class and their occurrence in the training split divided by dataset. No accuracy is given for class 2 in the Argoverse dataset because there is no occurrence in the validation set.

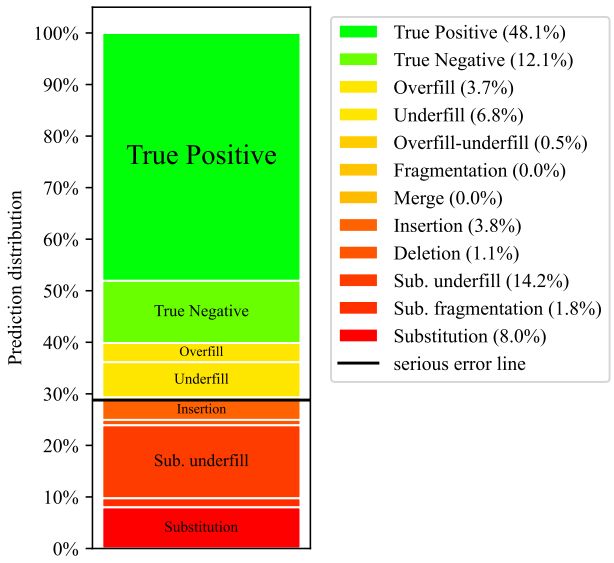

Figure 3. Error Distribution Diagram corresponding to the best-performing model from Tab. 2.

lower accuracy on both sets, which can be attributed to two reasons. Firstly, their occurrences are fewer compared to others. Secondly, these scenarios are more complex, they require information from the relation between agents and between agents and the environment. The accuracy on class 0 is also high because this is still present in every sequence and therefore dominant in the train set. Another positive insight is that overall accuracy is similar per class for both datasets. Despite some scenarios being underrepresented in one dataset (like class 2), the model generalizes well across traffic scenarios, not just specific datasets. In summary, the model is trained with relatively few instances of each class. This is particularly noticeable when compared to previous scenario classification studies. However, it continues to exhibit satisfactory performance.

6.4. State of the art comparison

The literature presents various works on scenario classification. However, these works often struggle with complex scenarios. [27] is capable of classifying more complex scenarios. To compare performance we conducted comparative tests using their model on our dataset. Implementation-wise, there are notable distinctions between the models. Firstly, our spatial aspect relies on Cartesian coordinates, theirs is based on the quadrant in which a vehicle or object is situated relative to another. secondly, their work performs classification per graph vertice instead of per timestep. This means that for a given observation window of length T with N detected objects it outputs N class predictions instead of T, as in our work. To make testing on our dataset possible, some alterations had to be made in the model of Mylavarapu [27]. Details on these alterations are described in Appendix A. The comparative results of these experiments can be found in Tab. 4. In the table, we differentiate between ego and non-ego actions. Ego actions are solely related to the ego vehicle's actions and their relationship

with the environment, indicated by a checkmark. Non-ego actions involve interactions between several agents and the ego vehicle and their relation to the environment. We can

#	Scenario	Ego action	Ours	[27]
0	No scenario	-	62.3	**64.1**
1	Cut-in	-	48.2	**58.6**
2	Stationary vehicle in lane	-	48.1	**59.9**
3	Ego lane change right	✓	**70.8**	39.6
4	Ego lane change left	✓	**52.2**	40.6
5	Right turn at crossing	✓	56.8	**82.0**
6	Left turn at crossing	✓	**75.3**	62.4
7	Straight ahead at crossing	✓	56.5	**59.0**
-	Average accuracy	-	**58.8**	58.3
-	Epoch training time	-	**46.9**	156.4
-	Ego action avg.	-	62.3	56.72
-	Non-ego action avg.	-	48.15	59.3

Table 4. Accuracy and training time per epoch comparison between our work and the model of Mylavarapu [27] altered to perform per frame scenario classification instead of per vertice classification.

conclude from Tab. 4 that the overall accuracy is very comparable. As we can see our model outperforms the ego actions specifically on the lane changes. The explanation for this lies in the fact that our model is ego-centered. This is because, in our GCN part, several layers are focused solely on the relation between the ego vehicle and the environment or actors. In their work, the GCN is based on the relation between all present vertices. Furthermore, our model's better performance in predicting ego actions is attributed to the quadrant-based approach's lower sensitivity to minor changes like during for example lane changes, as it only detects differences when a vertex moves to another quadrant. Next to this accuracy comparison, we compared the average required training time per epoch for both models. These results are also shown in Tab. 4. This is training only, so no validation. This shows that our model trains more than three times faster than theirs. Furthermore, in order to train their model on our dataset, we had to shrink the input size due to memory constraints. These factors collectively showcase the substantial computational efficiency advantage of our model. The explanation for this is that their temporal aggregation method, based on a combination of LSTMs and attention is significantly more complex than ours.

7. Discussion

To form a well-informed opinion about the results in previous sections, we must place them in context. To assess the PR-AUC's significance, we compare it to a random guessing model's PR-AUC. This value, calculated from positive occurrences in the validation set ($\frac{P}{N}$), averages 12.5. Contrasted with the top method in Tab. 2 (58.8), the model effectively identifies and detects scenarios. The per-class accuracy showcases the model's ability to classify all trained classes. Additionally, it holds promise for future implementations. The model's generalization extends beyond a single dataset, enabling further training with diverse sources for enhanced performance. This also brings possibilities to add scenarios that are not present in the currently used datasets. When comparing most methods in Sec. 2 it is trained on significantly less data and could have potential there as well. Our distinguishing aspect is its per-frame classification. This also presents a challenge in terms of performance. The EDD graph becomes insightful in this context. Even human annotators disagree about the precise beginning and end of a scenario. When we accept underfill and overfill errors to some extent, our model performs even better than at first sight. The work we compared our performance to currently shows comparable performance when modified for per-frame classification. But there are still other aspects where our model outperforms the state of the art. First of all in terms of flexibility. [27] only supports fixed sequence lengths. Our model does support varying sequence lengths. Another distinction lies in the substantial difference in training time among the models. Seeing this as a metric for computational efficiency, our model's selection of straightforward components like temporal convolutions over complex attention models demonstrates its advantages in this regard. In conclusion, we've developed a competitive method with scalability and computational efficiency advantages compared to the related works. Its potential is significant, especially when further refined, optimized, and supplemented with additional data.

8. Conclusion

In this work, we discussed that scenario-based testing of ADS is very time-efficient. Finding these scenarios streamlines this process more. We succeeded in designing a scenario classification method that is able to find the beginning and the end of diverse and complex scenarios. The model is based on GCNs for the spatial aspects and on CNNs for the temporal aspect. The combination makes it possible to learn to classify scenarios that are based on interactions of a vehicle between the environment as well as other vehicles. We showed that the model only classifies a serious error in less than 30% of the frames. This is achieved through training and validating the model on a dataset we developed, which extends nuScenes and Argoverse2, specifically designed for scenario classification. This model now forms a benchmark for future works on this dataset. Future works for this classifier will cover advanced network structures as attention models implemented in various parts of the model. This could be implemented for improved spatial aggregation as well as temporal aggregation.

Acknowledgment

Many thanks to everyone at Kognic for their support, especially Andreas Falkovén and Tommy Johansson for their daily guidance. I am also grateful to Dr. Holger Caesar for his valuable insights and guidance. All of Your contributions played a vital role in the success of this work. Thank you all.

Appendix

A. Comparison to Mylavarapu et al.

When we initially compare the model presented in [27] study with our own, it appears quite distinct. In broad terms, their approach takes camera footage as input and generates an activity label for each node within the graph. Specifically, these labels include: 1) moving away, 2) moving towards us, 3) parked, 4) left lane change, 5) right lane change, and 6) overtaking. Meaning that nodes that represent waypoints are labeled as parked.

Upon closer examination, the similarity between our model and theirs is more significant than anticipated. In both models, there is a clear distinction between spatial and temporal encoding. The difference in implementation is that the vertices in their model are of shape $V_i = (O)$. Where O is the object type $O = \{vehicle, waypoint\}$. This means that a waypoint is labeled as 1 and a car as 0. In our model, the nodes are given as $V_i = (x, y, \phi, v)$. Their model employs multi-relation GCNs over the quadrant in which two vertices are relative to each other, in ours, we use multi-relation GCNs based on the object types. In their model the temporal aspect is captured using LSTMs and attention networks in ours this is done with simple convolution over the time dimension. Also as discussed earlier our model focuses on the ego vehicle because several GCN layers perform convolution only on the relation between the ego vehicle and its environment or the surrounding agents. In their model, the GCN uses the relation between all the present vertices. This means that in the rare case that two scenarios happen simultaneously within the observation reach, their model not necessarily classify the correct scenario. Ours will be more likely to.

Because of these differences, certain adjustments were necessary prior to conducting a comparative test. As the published code lacks the semantic segmentation part, this is neglected. A graph made out of our dataset is used as input. We get adjacency in their format by computing angles between vertices. These angles are then replaced by $(0, 1, 3, 4, 5)$, which stand for top-left, bottom-left, top-right, bottom-right, and self-edge. This applies if the angle matches a quadrant or is a self-loop. The self-loops are on the diagonal. Furthermore, because the model is designed for fixed sequence lengths, the input sequences are padded to a fixed size. The padding is cut off after the GCN part. Since every timestep is handled separately inside the GCN, this padding does not affect the other timesteps. Converting their model to do classification over the timesteps instead of the vertices, the last AvgPool layers and fully connected layers are applied on the vertice dimension instead of the temporal dimension resulting in a $T \times n_{classes}$ output instead of $N \times n_{classes}$.

B. Data augmentation

To address the problem that our dataset is relatively small and imbalanced, we conducted some experiments with data augmentation. We did several experiments for the less-represented classes. The outcomes of these varied experiments are presented in Table 5. To see what the effect is on the other classes, the accuracy is given for every class as well as the average. The experiments are explained in more detail below.

Stretching In the experiments labeled as "stretching," we modified existing scenarios by adjusting the graph's x (the longitudinal ego driving direction) component by a factor of 1.02 and the y component by 0.98. This ensured that none of the values remained unchanged. The velocity and driving angle were adjusted accordingly to match the correct values for the newly stretched frames.

Mirroring Under the assumption that a scenario could happen from two sides, we conducted experiments with mirroring. This means the y components are multiplied by -1. This way it is mirrored along the longitudinal axis from the ego perspective. For the special cases of left and right lane change the label is also changed after mirroring the scenario. This means that "mirror class 4" in the table is obtained by mirroring the existing scenarios labeled as class 3. For the other classes, the label remains the same.

Results Based on the information presented in Table 5, it is evident that the outcomes of the data augmentation experiments have not resulted in the expected improvements in performance. Specifically, when we observe the impact of stretching on classes 1, 2, and 4, as well as mirroring classes 1, 3, and 4, we observed a decrease in the accuracy of these respective classes. In the other experiments, there was a notable enhancement in the performance of those individual classes. However, this improvement came at the cost of a reduction in the overall average performance. This suggests that while the augmented data may benefit specific classes, it affects the models' ability to generalize to other classes negatively. Interestingly, when we applied mirroring to class 3, we observed an increase in the average performance but a decrease in the accuracy of class 3 itself. When

combining all these data augmentation techniques the performance appeared to be similar to the original dataset. Due to the unpredictability of these outcomes and the relatively low impact on the overall results, no data augmentation was used in the design and validation phase of this network.

C. Output visualization

To gain a better understanding of the different model performances we visualized a specific scenario in Fig 5. To gain a better understanding of this specific scenario, a snapshot of the front camera is shown in Fig. 4 Where we see the agent cutting in from the right. In this visualization, only the relevant vehicles are plotted. Meaning that we have the ego vehicle and one agent. To improve clarity, we've represented the (x, y) coordinates at intervals of every 2 frames, effectively creating a 2Hz frequency display. The green sections denote drivable road segments. Looking closer at the visualization we notice that it is a cut-in scenario. The agent comes in from the side of the road. The decreasing gap between the ego points as time progresses implies braking, the widening gap between the locations of the agents indicates that there is acceleration happening within the lane. The outputs of the different models for this specific scenario are shown in Table 6. Compared to the ground truth the full model performs quite well. The scenario is detected too early so the predictions at frames 5 and 6 will be labeled as overfill for the EDD. The importance of temporal aggregation is shown in the model "No temporal agg". First of all, it shows a lot of wrong classifications, but secondly, it fails to learn the temporal aspect of any scenario. This can be seen because it switches between classes at high frequency and scenario occurrences of 1 frame exist. This is not the case in real-world scenarios. The LSTM learns that there is a cut in the present within this sequence since it is only predicting 1's and 0s, but it fails the timing. Note that this example is only for illustrational purposes and that it does not necessarily represent the performance of the separate models, this can be seen in Table 2.

In Table 6, you can see the results of various models applied to this specific scenario. The full model, when compared to the ground truth, demonstrates good performance. However, it appears to detect the scenario a bit early, leading to frames 5 and 6 being erroneously labeled as overfill for the EDD. The significance of temporal aggregation becomes apparent when examining the "No temporal agg" model. Firstly, it exhibits a high number of incorrect classifications. Moreover, it fails to capture the temporal aspect of the scenario correctly. This can be seen in the frequent switches between classes over time. These rapid switches in scenarios are never the case in real-world data. The LSTM model learns the correct scenario since there are only 0s and 1s present, but it fails to learn the timing of the scenario. It's important to note that this example serves purely illustrative purposes and may not accurately represent the performance of individual models. This performance of the models is detailed in Table 2.

Figure 4. Snapshot of the front camera during the scenario depicted in Fig. 5.

	Class								
Experiment	0	1	2	3	4	5	6	7	**Avg.**
No augmentation	62.3	48.2	48.2	70.8	52.2	56.8	73.3	56.5	58.8
Stretching class 1	60.0	39.9	67.9	64.2	31.9	43.6	65.6	44.5	52.2
Stretching class 2	62.2	58.0	64.7	63.2	55.1	49.2	64.0	41.1	57.2
Stretching class 3	65.7	44.4	62.6	57.6	52.9	50.2	65.3	47.7	55.7
stretching class 4	61.8	50.8	54.6	65.1	42.0	48.0	72.2	49.7	55.5
Mirror class 1	64.2	25.4	54.0	56.6	44.9	63.9	66.7	52.2	53.5
Mirror class 2	66.1	36.3	59.9	58.5	49.3	47.6	73.1	51.6	55.3
Mirror class 3	63.8	63.7	71.7	47.8	49.4	59.3	10	50.5	59.3
Mirror class 4	64.0	38.9	57.8	66.0	47.1	45.8	58.0	55.5	54.1
All of the above	61.3	39.9	57.7	65.1	50.7	57.5	78.5	56.9	58.3

Table 5. Different data augmentation experiments and their results on the model's accuracy

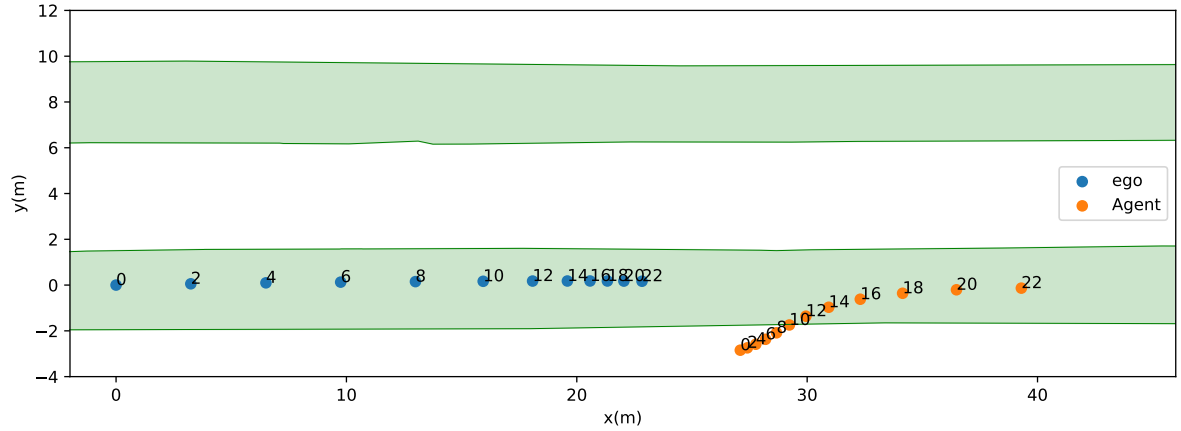

Figure 5. Bird's eye view visualization of a cut in (class 1) scenario.

	Frame																							
Model version	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Ground truth	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**0**	**0**
Full model	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
LSTM (with res. conn.)	1	1	1	1	1	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0
No temporal agg.	1	1	1	2	1	1	1	1	3	2	3	3	3	3	2	2	3	3	3	3	3	3	3	3
Baseline	0	1	1	1	1	1	1	1	1	0	0	0	2	2	2	2	2	2	2	2	2	2	2	0

Table 6. Per frame classification outputs of different model versions on the traffic scenario illustrated in Fig. 5.

References

[1] Matthias Althoff, Markus Koschi, and Stefanie Manzinger. CommonRoad: Composable benchmarks for motion planning on roads. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 719–726, Los Angeles, CA, USA, June 2017. IEEE. 3

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization, July 2016. arXiv:1607.06450 [cs, stat]. 6

[3] Halil Beglerovic, Jonas Ruebsam, Steffen Metzner, and Martin Horn. Polar Occupancy Map - A Compact Traffic Representation for Deep Learning Scenario Classification. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 4197–4203, Oct. 2019. 2, 3

[4] Halil Beglerovic, Thomas Schloemicher, Steffen Metzner, and Martin Horn. Deep Learning Applied to Scenario Classification for Lane-Keep-Assist Systems. *Applied Sciences*, 8(12):2590, Dec. 2018. Number: 12 Publisher: Multidisciplinary Digital Publishing Institute. 2, 3

[5] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A Multimodal Dataset for Autonomous Driving. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11618–11628, Seattle, WA, USA, June 2020. IEEE. 2, 3

[6] Holger Caesar, Juraj Kabzan, Kok Seang Tan, Whye Kit Fong, Eric Wolff, Alex Lang, Luke Fletcher, Oscar Beijbom, and Sammy Omari. NuPlan: A closed-loop ML-based planning benchmark for autonomous vehicles, Feb. 2022. arXiv:2106.11810 [cs]. 3

[7] Council of European Union. Annexes to the commision implementing regulation (EU) no 2019/2144, 2022. 3

[8] Council of European Union. Commision implementing regulation (EU) no 2019/2144, 2022. 1, 3

[9] Ahmetcan Erdogan, Burak Ugranli, Erkan Adali, Ali Sentas, Eren Mungan, Emre Kaplan, and Andrea Leitner. Real-World Maneuver Extraction for Autonomous Vehicle Validation: A Comparative Study. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 267–272, June 2019. ISSN: 2642-7214. 2, 3

[10] A Geiger, P Lenz, C Stiller, and R Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, Sept. 2013. Publisher: SAGE Publications Ltd STM. 3

[11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep Sparse Rectifier Neural Networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 315–323. JMLR Workshop and Conference Proceedings, June 2011. ISSN: 1938-7228. 5, 6

[12] Richard Gruner, Philip Henzler, Gereon Hinz, Corinna Eckstein, and Alois Knoll. Spatiotemporal representation of driving scenarios and classification using neural networks. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1782–1788, June 2017. 2, 3

[13] John Houston, Guido Zuidhof, and Luca Bergamini et al. One thousand and one hours: Self-driving motion prediction dataset. In *CoRL*, 2020. 3

[14] Ihab Kaddoura, Joschka Bischoff, and Kai Nagel. Towards welfare optimal operation of innovative mobility concepts: External cost pricing in a world of shared autonomous vehicles. *Transportation Research Part A: Policy and Practice*, 136:48–63, June 2020. 1

[15] Nidhi Kalra and Susan M. Paddock. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part A: Policy and Practice*, 94:182–193, Dec. 2016. 1

[16] Aida Khosroshahi, Eshed Ohn-Bar, and Mohan Manubhai Trivedi. Surround vehicles trajectory analysis with recurrent neural networks. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 2267–2272, Nov. 2016. ISSN: 2153-0017. 2

[17] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, Jan. 2017. arXiv:1412.6980 [cs]. 7

[18] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks, Feb. 2017. arXiv:1609.02907 [cs, stat]. 2, 4

[19] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-Normalizing Neural Networks, Sept. 2017. arXiv:1706.02515 [cs, stat]. 6

[20] Xin Li, Xiaowen Ying, and Mooi Choo Chuah. GRIP: Graph-based Interaction-aware Trajectory Prediction. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3960–3966, Oct. 2019. 2

[21] Xin Li, Xiaowen Ying, and Mooi Choo Chuah. GRIP++: Enhanced Graph-based Interaction-aware Trajectory Prediction for Autonomous Driving, May 2020. arXiv:1907.07792 [cs]. 2, 3

[22] Ming Liang, Bin Yang, Rui Hu, Yun Chen, Renjie Liao, Song Feng, and Raquel Urtasun. Learning Lane Graph Representations for Motion Forecasting, July 2020. arXiv:2007.13732 [cs]. 2, 3, 5

[23] W.J.R. Louwerse and S.P. Hoogendoorn. ADAS safety impacts on rural and urban highways. In *IEEE Intelligent Vehicles Symposium, 2004*, pages 887–890, June 2004. 1

[24] Moneim Massar, Imran Reza, Syed Masiur Rahman, Sheikh Muhammad Habib Abdullah, Arshad Jamal, and Fahad Saleh Al-Ismail. Impacts of Autonomous Vehicles on Greenhouse Gas Emissions—Positive or Negative? *International Journal of Environmental Research and Public Health*, 18(11):5567, Jan. 2021. Number: 11 Publisher: Multidisciplinary Digital Publishing Institute. 1

[25] David Minnen, Tracy Westeyn, Thad Starner, Jamie Ward, and Paul Lukowicz. Performance metrics and evaluation issues for continuous activity recognition," in Performance Metrics for Intelligent Systems. *Proceedings of Performance Metrics in Intelligent Systems Workshop*, Jan. 2006. 7

[26] Sajjad Mozaffari, Omar Y. Al-Jarrah, Mehrdad Dianati, Paul Jennings, and Alexandros Mouzakitis. Deep Learning-Based Vehicle Behavior Prediction for Autonomous Driving Applications: A Review. *IEEE Transactions on Intelligent Transportation Systems*, 23(1):33–47, Jan. 2022. Conference

Name: IEEE Transactions on Intelligent Transportation Systems. 3

[27] Sravan Mylavarapu, Mahtab Sandhu, Priyesh Vijayan, K Madhava Krishna, Balaraman Ravindran, and Anoop Namboodiri. Towards Accurate Vehicle Behaviour Classification With Multi-Relational Graph Convolutional Networks. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 321–327, Oct. 2020. ISSN: 2642-7214. 2, 3, 8, 9, 10

[28] Demin Nalic, Tomislav Mihalj, Maximilian Baeumler, Matthias Lehmann, Arno Eichberger, and Stefan Bernsteiner. *Scenario Based Testing of Automated Driving Systems: A Literature Survey*. Oct. 2020. 1

[29] Christian Neurohr, Lukas Westhofen, Tabea Henning, Thies de Graaff, Eike Möhlmann, and Eckard Böde. Fundamental Considerations around Scenario-Based Testing for Automated Driving. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 121–127, Oct. 2020. ISSN: 2642-7214. 1

[30] Julia Nilsson, Jonas Fredriksson, and Erik Coelingh. Rule-Based Highway Maneuver Intention Recognition. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 950–955, Sept. 2015. ISSN: 2153-0017. 2, 3

[31] Stefan Riedmaier, Thomas Ponn, Dieter Ludwig, Bernhard Schick, and Frank Diermeyer. Survey on Scenario-Based Safety Assessment of Automated Vehicles. *IEEE Access*, 8:87456–87477, 2020. Conference Name: IEEE Access. 1

[32] Zihao Sheng, Yunwen Xu, Shibei Xue, and Dewei Li. Graph-Based Spatial-Temporal Convolutional Network for Vehicle Trajectory Prediction in Autonomous Driving. *IEEE Transactions on Intelligent Transportation Systems*, 23(10):17654–17665, Oct. 2022. Conference Name: IEEE Transactions on Intelligent Transportation Systems. 2, 3

[33] Matteo Simoncini, Douglas Coimbra de Andrade, Leonardo Taccari, Samuele Salti, Luca Kubin, Fabio Schoen, and Francesco Sambo. Unsafe Maneuver Classification From Dashcam Video and GPS/IMU Sensors Using Spatio-Temporal Attention Selector. *IEEE Transactions on Intelligent Transportation Systems*, 23(9):15605–15615, Sept. 2022. Conference Name: IEEE Transactions on Intelligent Transportation Systems. 2

[34] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 06 2014. 2

[35] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Sheng Zhao, Shuyang Cheng, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in Perception for Autonomous Driving: Waymo Open Dataset, May 2020. arXiv:1912.04838 [cs, stat]. 3

[36] Simon Ulbrich, Till Menzel, Andreas Reschka, Fabian Schuldt, and Markus Maurer. Defining and Substantiating the Terms Scene, Situation, and Scenario for Automated Driving. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 982–988, Sept. 2015. ISSN: 2153-0017. 3

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need, Dec. 2017. arXiv:1706.03762 [cs]. 2

[38] Nannan Wang, Xi Wang, Paparao Palacharla, and Tadashi Ikeuchi. Cooperative autonomous driving for traffic congestion avoidance through vehicle-to-vehicle communications. In *2017 IEEE Vehicular Networking Conference (VNC)*, pages 327–330, Nov. 2017. ISSN: 2157-9865. 1

[39] Jamie Ward, Paul Lukowicz, and Gerhard Tröster. Evaluating Performance in Continuous Context Recognition Using Event-Driven Error Characterisation. pages 239–255, May 2006. 7

[40] Jamie A Ward, Nagendra B Bharatula, Gerhard Troster, and Paul Lukowicz. Continuous activity recognition in the kitchen using miniaturised sensor button. 7

[41] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, Deva Ramanan, Peter Carr, and James Hays. Argoverse 2: Next Generation Datasets for Self-Driving Perception and Forecasting. 3

[42] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, Deva Ramanan, Peter Carr, and James Hays. Argoverse 2: Next generation datasets for self-driving perception and forecasting. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS Datasets and Benchmarks 2021)*, 2021. 2, 3

[43] Fisher Yu and Vladlen Koltun. Multi-Scale Context Aggregation by Dilated Convolutions, Apr. 2016. arXiv:1511.07122 [cs]. 2, 6