A Graph-based Framework for Coverage Analysis in Autonomous Driving

Thomas Mühlenstädt and Marius Bause

No Institute Given

1 Abstract

Coverage analysis is essential for validating the safety of autonomous driving systems, yet existing approaches typically assess coverage factors individually or in limited combinations, struggling to capture the complex interactions inherent in traffic scenes. This paper proposes a graph-based framework for coverage analysis that represents traffic scenes as hierarchical graphs, combining map topology with actor relationships. The framework introduces a two-phase graph construction algorithm that systematically captures spatial relationships between traffic participants, including leading, following, neighboring, and opposing configurations.

Two complementary coverage analysis methods are presented. First, a subgraph isomorphism approach matches traffic scenes against a set of manually defined archetype graphs representing common driving scenarios. Second, a graph embedding approach utilizes Graph Isomorphism Networks with Edge features (GINE) trained via self-supervised contrastive learning to project traffic scenes into a vector space, enabling similarity-based coverage assessment.

The framework is validated on both real-world data from the Argoverse 2.0 dataset and synthetic data from the CARLA simulator. The subgraph isomorphism method is used to calculate node coverage percentages using predefined archetypes, while the embedding approach reveals meaningful structure in the latent space suitable for clustering and anomaly detection. The proposed approach offers significant advantages over traditional methods by scaling efficiently to diverse traffic scenarios without requiring scenario-specific handling, and by naturally accommodating varying numbers of actors in a scene.

2 Introduction

In autonomous driving, coverage analysis is a crucial step to ensure the safety and reliability of the system. In most situations, coverage arguments are collected either per coverage factor, or maybe up to 2 or 3 factor interactions. See for example [8] for a production-grade implementation of state-of-the-art coverage analysis. In contrast to existing approaches, this paper proposes a graph-based framework for coverage analysis. While graph-based representations of traffic scenes already exist, they have not been specifically designed for this purpose.

This work bridges that gap by utilizing graph-based traffic scene representations explicitly for coverage analysis.

This paper is structured as follows: In chapter 3, existing coverage and analysis approaches are discussed. The basis for the following chapters is layed in chapter 4. Afterwards, two coverage approaches using the traffic scene graphs are discussed in chapter 5 and chapter 6. Finally, the developed methods are applied to Carla and Argoverse 2.0 data in chapter 7 followed by a short summary and outlook.

3 Existing coverage and analysis approaches

Foundational terminology for automated driving is established by the SAE J3016 taxonomy [15], which defines six levels of driving automation and key concepts such as the Dynamic Driving Task (DDT) and Operational Design Domain (ODD). Building on this, [19] provide precise definitions for the commonly conflated terms *scene*, *situation*, and *scenario*, enabling a consistent vocabulary for test design and coverage arguments.

Scenario-based testing has emerged as the dominant paradigm for validating automated driving functions. The PEGASUS project [16] introduces a six-layer model to decompose the driving environment into structured logical scenarios, shifting validation from infeasible distance-based testing toward systematic scenario exploration across simulation and field tests. [13] refine this with three abstraction levels—functional, logical, and concrete scenarios—while noting that existing parameter-selection methods lack a systematic way to determine meaningful test coverage. Complementing these frameworks, [9] propose object-oriented ontologies for scenario description with explicit linkage to coverage arguments, and [17] present trajectory-based clustering of real-world driving data to structure the scenario space and reduce test-set redundancy.

A central challenge is the *approval trap* identified by [20]: statistically demonstrating superior safety through real-world driving alone would require billions of test kilometres, motivating the need for simulation-based tools and systematic coverage methods. General software testing theory [1] provides foundational coverage concepts (statement, branch, condition coverage), while [8] demonstrate their domain-specific application through coverage buckets, parameterised items, and performance metrics such as time-to-collision. At the standards level, ISO 21448 [18] and UL 4600 [12] both define frameworks for coverage criteria and measurement in the context of automated driving safety.

4 Defining a traffic scene graph

4.1 Map Graph Construction

The map graph is a directed multigraph $G_{\text{map}} = (V_{\text{map}}, E_{\text{map}})$ where each node $v \in V_{\text{map}}$ represents a lane segment, storing geometric information (boundaries, centerline, length) and semantic attributes (intersection status, road and lane

type). Edges encode three spatial relationships between lanes: **following edges** connect lanes forming a continuous path in the same direction, **neighbor edges** connect adjacent lanes traveling in the same direction, and **opposite edges** connect lanes traveling in opposite directions. The map graph is constructed by processing map data from Argoverse or CARLA to extract these relationships. Lanes with geometric overlap are marked as intersection lanes.

4.2 Actor Graph Construction

The actor graph $G_{\text{actor}} = (V_{\text{actor}}, E_{\text{actor}})$ represents dynamic relationships between actors at a specific timestep. Each node $v \in V_{\text{actor}}$ represents an actor and stores attributes including primary lane ID, all occupied lane IDs, longitudinal position s along the lane centerline, 3D position, longitudinal speed, actor type, and a lane change indicator. Each edge $e \in E_{\text{actor}}$ stores a relation type and a path length in meters along the lane network.

Four relationship types are defined between actors, ordered by semantic hierarchy:

1. **Following/Leading**: Longitudinal relationships where actors share the same lane or are connected entirely by following edges.
2. **Neighbor**: Lateral relationships via paths containing exactly one neighbor edge (actors need not be on immediately adjacent lanes).
3. **Opposite**: Relationships via paths containing exactly one opposite edge (actors need not be on immediately opposite lanes).

4.3 Hierarchical Graph Construction Algorithm

The actor graph is constructed in two phases that separate relation discovery from graph building, enabling hierarchical processing and preventing redundant edges. All parameters are listed in Table 1.

Phase 1: Relation Discovery: For each actor pair (A, B) at timestep t, the algorithm determines their primary lanes and checks if a connecting path exists in the map graph. The path structure determines the relationship type: paths consisting entirely of following edges yield following/leading relations, paths with exactly one neighbor (or opposite) edge yield neighbor (or opposite) relations. Both lane-based path length and Euclidean distance are checked against the thresholds in Table 1 to filter distant actors.

Phase 2: Hierarchical Graph Construction: Edges are added in hierarchical order—leading/following first (highest priority), then neighbor, then opposite—each sorted by path length (shortest first). Before adding an edge between actors A and B, a breadth-first search checks whether a path of length $\leq$ max_node_distance already exists in the current graph. If so, the direct edge is skipped, as the relationship is already implicitly encoded. The graph is updated after each edge addition so that subsequent checks reflect the current state.

This redundancy prevention significantly reduces edges while preserving connectivity. For example, three vehicles in a row need only two lead-follow edges;

Table 1: Input parameters for actor graph construction

Parameter	Type	Description	Value
Distance limits (discovery phase)			
`max_distance_lead_veh_m`	float	Max distance for leading/following	100
`max_distance_neighbor_fwd_m`	float	Max distance for forward neighbor	50
`max_distance_neighbor_bwd_m`	float	Max distance for backward neighbor	50
`max_distance_opposite_fwd_m`	float	Max distance for forward opposite	100
`max_distance_opposite_bwd_m`	float	Max distance for backward opposite	10
Node distance limits (construction phase)			
`max_node_dist_leading`	int	Max edge count in path for leading/following	3
`max_node_dist_neighbor`	int	Max edge count in path for neighbor	2
`max_node_dist_opposite`	int	Max edge count in path for opposite	2
Timestep configuration			
`delta_timestep_s`	float	Time step increment in seconds	1.0

the third pairwise relation is encoded through the intermediate vehicle. Similarly, for a row of vehicles adjacent to an opposite-direction vehicle, only the closest pairwise relation needs an explicit edge (see Figure 1).

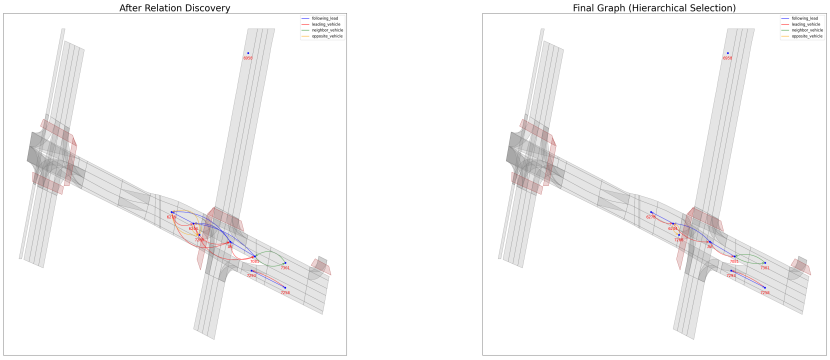
5 Create subgraphs for coverage analysis

Traffic scene archetypes from the literature (e.g., lead vehicle following, opposite traffic) can be expressed as small graphs. To determine whether such an archetype appears in a larger traffic scene graph G, we check if any subgraph of G is isomorphic to the archetype graph A. While subgraph isomorphism is NP-hard in general, the graphs considered here are small enough for practical computation using the VF2 algorithm [4].

Our strategy is as follows: (1) Define a set of archetype subgraphs S representing relevant traffic situations. (2) Specify which node and edge attributes are considered for the isomorphism check. (3) For each traffic scene graph G (e.g., from CARLA or Argoverse), check if any subgraph of G is isomorphic to any archetype in S and record the result in a coverage table C.

This bottom-up approach starts from individual situation archetypes and checks their presence across datasets. The resulting coverage table enables follow-up analyses such as visualizing attribute distributions (speed, distance) for matched scenes, cross-tabulating co-occurring archetypes, or computing AV performance metrics conditioned on specific archetypes.

We defined 18 subgraph archetypes covering common traffic scenarios: simple 2-actor patterns (following, opposite, neighbor—used only for isolated pairs), 3-actor patterns (lead vehicle with neighbor, platoons, opposite traffic, lane change/cut-in scenarios with intersection variants), 4-actor patterns (cut-out, multi-vehicle platoons, lead-follow with opposite traffic and intersection variants), and 5-actor patterns combining lead, neighbor, and opposite vehicles with intersection variants.



(a) After relation discovery (b) After hierarchical selection

Fig. 1: The discovery graph (left) contains all relations within distance limits. Hierarchical selection (right) removes redundant edges representable through existing paths.

The last step in the analysis of the subgraphs is to identify coverage holes. These provide an intuitive understanding of what traffic situations are missing from the simulated environment. The approach identifies scenario archetypes that are significantly underrepresented or absent in one dataset compared to another *REF* dataset.

These concepts will be applied to CARLA and Argoverse datasets in the application chapter 7.

6 Graph Embeddings for Traffic Scene Analysis

The subgraph isomorphism approach from Section 5 provides a bottom-up methodology for analyzing traffic scenarios through predefined archetypes. However, real-world traffic scene graphs exhibit significantly higher complexity than these patterns, motivating complementary top-down approaches such as graph embeddings.

Embeddings translate raw data into a vector space where distances measure similarity, analogous to the Word2Vec model for words ([14]). For traffic scene graphs, embeddings enable coverage analysis by comparing scenes: identifying similar simulation–real-world scenario pairs, detecting near duplicates, or visualizing structures that would be intractable in the original space of all possible traffic scenes.

A Graph Isomorphism Network with Edge features (GINE) [10] is used to generate embeddings, implemented in PyTorch Geometric ([7]). GINE is chosen because it directly incorporates edge features through its aggregation function:

edge type and path length between actors are encoded as edge features (see Section 2) and directly influence the message-passing process.

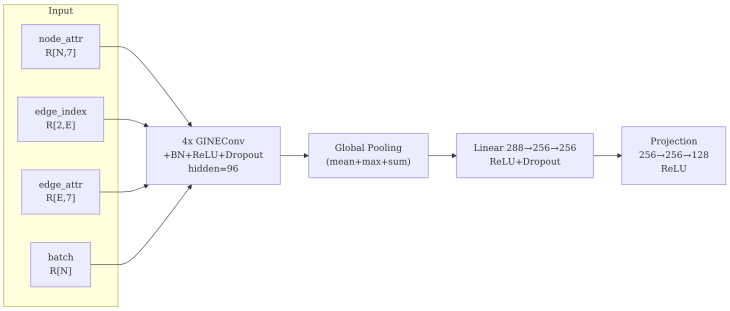


Fig. 2: Model architecture for the Graph Isomorphism Network with Edge features (GINE).

The architecture is shown in Figure 2. Node features are actor type (one-hot: vehicle, pedestrian, cyclist, motorcycle), speed, intersection presence, and lane change since the last timestep. Edge features are edge type (one-hot, six spatial relationship categories) and path length between nodes.

The model was trained on CARLA and Argoverse 2.0 using self-supervised contrastive learning [2], [10]. For each mini-batch of 384 graphs, two augmented views were created by perturbing continuous attributes with Gaussian noise ($\sigma = 0.08$) and randomly dropping edges (probability 0.1). A five-layer GINE encoder (hidden width 384) produced graph-level representations via concatenated mean, max, and sum pooling, followed by an embedding MLP (192-dimensional ℓ_2-normalized) and a projection head. The contrastive loss maximized agreement between views of the same graph while contrasting against other in-batch graphs (cosine similarity, $\tau = 0.07$). Optimization used AdamW (weight decay 5×10^{-6}), learning rate warmup over three epochs, and exponential decay (initial rate 0.0015, factor 0.85) across 15 training stages [10], [7].

7 Application: Analysing traffic scenes with coverage graphs

Having defined a graph-based traffic scene representation, we can now analyse the coverage of the system. Two methodologies are proposed for this purpose: One is to define archetypes of traffice scenes, and to compare graphs from observed traffic scenes to these archetypes. The second one is to translate graphs to graph embeddings, and then to compare the embeddings of different sets of traffic scenes.

7.1 Argoverse 2.0

Argoverse 2.0 [21] is a large-scale dataset for autonomous driving research, developed by Argo AI. It provides real-world sensor data collected from autonomous vehicle test fleets operating in six geographically diverse U.S. cities: Austin, Detroit, Miami, Palo Alto, Pittsburgh, and Washington D.C. The dataset includes high-definition LiDAR point clouds, ring camera imagery, and detailed vector map information. A key component is the Motion Forecasting Dataset, which contains 250,000 scenarios of 11 seconds each, featuring tracked object trajectories for vehicles, pedestrians, cyclists, and other road users.

The dataset has become a standard benchmark in the autonomous driving community, particularly for motion prediction and trajectory forecasting tasks. Its rich annotations include object classifications, track identities across frames, and semantic map information such as lane boundaries, crosswalks, and traffic signal locations. The diverse geographic coverage ensures exposure to varying road geometries, traffic patterns, and driving behaviors, making it suitable for developing and evaluating generalizable autonomous driving algorithms.

For this work, the Motion Forecasting Dataset is used, specifically focusing on the tracked object trajectories and their spatial relationships to construct traffic scene graphs. As the dataset is quite massive, a subset of 17000 scenes from the train partition is used for the analysis.

7.2 Carla

CARLA (Car Learning to Act, [5]) is an open-source simulator specifically designed for autonomous driving research and development. It provides a highly realistic urban driving environment with diverse road layouts, weather conditions, and traffic scenarios. The simulator features a comprehensive sensor suite simulation, flexible API for scenario creation, and supports both learning-based and traditional autonomous driving approaches. CARLA enables researchers to test and validate autonomous vehicle systems in a safe, controllable environment before real-world deployment.

The simulator has gained widespread adoption across both academic and industrial settings. In research, CARLA serves as a standard platform for developing and benchmarking autonomous driving algorithms, including reinforcement learning approaches for vehicle control and sensor fusion techniques [3]. Industry applications include virtual testing of production autonomous vehicle systems, scenario-based validation pipelines, and integration with hardware-in-the-loop testing frameworks [11]. CARLA is also extensively used in autonomous driving competitions and challenges, providing a common evaluation environment for comparing different approaches across research groups worldwide.

Here, Carla version 0.9.15 is used. The CARLA version 0.10.0 is not used, because it had only 2 maps and Mine_1 (which is not really normal roads) at the start of this project. Specifically, the following maps were used: Town01, Town02, Town03, Town04, Town05 and Town07.

A data generation script implements some behavior control mechanisms to create diverse and realistic traffic scenarios. Multiple vehicle types including trucks, motorcycles, and regular cars are spawned with varying probabilities, each exhibiting different behavioral characteristics such as speed preferences, following distances, and lane-changing tendencies. The script incorporates dynamic behavior modifications during simulation, including random slowdowns, periodic behavior changes, and adaptive responses to traffic conditions, resulting in rich and varied traffic scene data across multiple CARLA maps and simulation iterations. The simulation runs have between 20 and 60 vehicles each.

The resulting data consists of 2050 scenes with 11 seconds of simulation time each.

7.3 Subgraph Isomorphism Coverage Analysis

We apply the subgraph isomorphism approach to both the CARLA and the Argoverse data to identify coverage scenarios. The archetypes used here are defined manually and described in table 2. The chosen archetypes are typical traffic situations like e.g. lead vehicle situations, lane change situations or different combinations of opposite direction vehicles.

Table 2: Overview of all subgraph archetypes with their structural properties. Edge types: *fl* = following_lead, *lv* = leading_vehicle, *nv* = neighbor_vehicle, *ov* = opposite_vehicle.

Group	**Archetype**	**Actors**	**Edges**	**Edge Types**
Simple (2-actor)	Simple Following	2	2	fl, lv
	Simple Opposite	2	2	ov
	Simple Neighbor	2	2	nv
Complex (3-actor)	Lead + Neighbor (intersection)	3	4	fl, lv, nv
	Cut-in	3	4	fl, lv
	Cut-in (intersection)	3	4	fl, lv
	Platoon (intersection)	3	4	fl, lv
	Opposite Traffic (intersection)	3	4	fl, lv, ov
	Lead + Neighbor at Intersection	3	4	fl, lv, nv
	Triple Opposite (intersection)	3	4	ov
	Lead + Following in Back	3	4	fl, lv
Complex (4-actor)	Lead + Neighbor	3	4	fl, lv, nv
	Cut-out	4	6	fl, lv, nv
	Cut-out (intersection)	4	6	fl, lv, nv
	4-Vehicle Platoon (intersection)	4	6	fl, lv
	4-Vehicle Opposite (intersection)	4	6	fl, lv, ov
Complex (5-actor)	Lead + Neighbor + Opposite	5	8	fl, lv, nv, ov
	Lead + Neighbor + Opposite (inters.)	5	8	fl, lv, nv, ov

Also, information like which traffic actors are on an intersection are incorporated into the archetypes. Overall, the node coverage analysis across Argoverse and CARLA datasets achieved 62% overall coverage. The results are shown in Figure 3 to Figure 6. In Figure 3 there is a clear signal that for both datasets the coverage is not uniform per the different archetypes. Also, the distribution for the Carla dataset is not close to the Argoverse distribution, indicating that the CARLA data is not representative for the Argoverse data. Even worse, the Carla dataset is nearly completely missing out e.g. on the cut_out_intersection archetype, clearly indicating a rather randomly generated Carla dataset. A next step of analysis is to check, which archetypes are occuring simultaneously in a traffic scene. Figure 4 shows the agreement matrix for the manually defined coverage scenarios for CARLA and Argoverse. The heatmaps show the percentage of agreement between the manually defined coverage scenarios for CARLA and Argoverse.

A last example of how to use the assignment of archetypes to traffic scenes is to check the parameter distribution for the speed and path length of the traffic scenes. Figure 5 and Figure 6 show the parameter distribution for the speed and path length of the traffic scenes for CARLA and Argoverse. The plots show that the distribution for the CARLA data differs from the Argoverse distribution, indicating that the CARLA data is not representative for the Argoverse data. Given the assignment of archetypes to traffic scenes based on the actors can be used in further analysis not done here, e.g. to check more divers parameters and distributions.

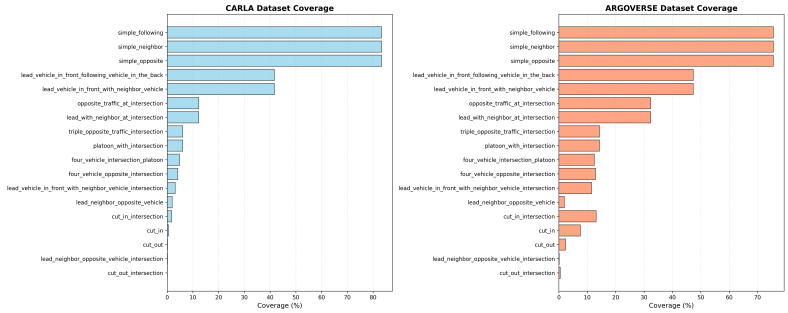
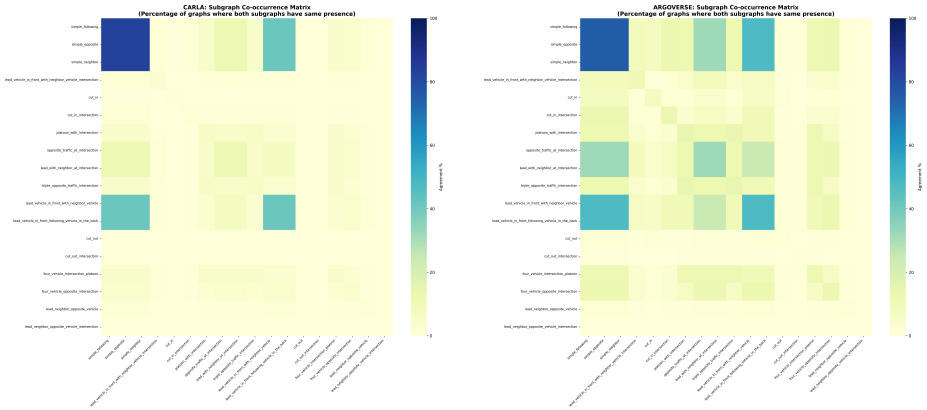


Fig. 3: Coverage barcharts for the manually defined coverage scenarios for CARLA and Argoverse.

7.4 Coverage gap analysis using subgraphs

Structural Coverage Analysis Figure 7 presents a comprehensive comparison of subgraph archetype coverage across the two datasets. The dual-axis visualization shows both the relative coverage percentages (left y-axis) and their



(a) Co-occurrence matrix for CARLA. (b) Co-occurrence matrix for Argoverse.

Fig. 4: Co-occurrence matrices showing the percentage of scenarios where pairs of subgraph patterns appear together in the same traffic scene for CARLA and Argoverse datasets.

differences (right x-axis, shown as diamond markers). Green markers indicate scenarios more prevalent in CARLA, while red markers highlight scenarios more common in Argoverse.

The analysis reveals several structural definition holes in the CARLA dataset. Most notably, complex multi-actor scenarios such as intersection-based patterns show significant underrepresentation. For instance, the `complex_4actor_cut_out_intersection` pattern appears in 8.2% of Argoverse scenes but only 0.3% of CARLA scenes, representing a coverage gap of 7.9 percentage points. Similarly, `complex_3actor_opposite_traffic_intersection` exhibits a gap of 6.5 percentage points (9.1% in Argoverse vs. 2.6% in CARLA).

Conversely, CARLA demonstrates higher coverage for simpler highway scenarios. The `simple_2actor_following` pattern appears in 42.3% of CARLA scenes compared to 31.8% in Argoverse, suggesting that the CARLA simulation emphasizes highway driving conditions over complex urban intersections.

The structured, bottom-up approach presented here demonstrates that subgraph isomorphism can effectively characterize traffic scene coverage for predefined archetypes. However, the manual definition of archetypes and the computational cost of isomorphism checking for large scenario collections motivate the graph embedding approach explored in the following chapter.

7.5 Graph Embeddings Coverage Analysis

The resulting embeddings have been analysed in a number of ways. For plausibility checks, for a number of randomly samples scenarios, the scenario with closest embedding vector (euclidean distance) has been visualized. This is shown

in Figure 10. This includes comparison between CARLA and Argoverse scenarios.

As a next step, the embedding space has been analysed using dimension reduction techniques, specifically PCA and t-SNE. This is shown in Figure 9. This can be done in a number of different flavors, like for example:

- distinguishing between CARLA and Argoverse scenarios by a color coding,
- visualizing just the CARLA scenarios and color code them by the map, in order to see if the maps deliver different types of scenarios,
- by calculating a density distribution of the embedding space for the Argoverse scenarios, and to check if for a regions with a density about a certain threshold, there exist CARLA scenarios, i.e. do a coverage analysis in the embedded space
- do the same vice versa, i.e. to check for relevance of CARLA scenarios.

As the embedding space is a normal metric space, representing scenarios across a large range of different driving situations, these embeddings can be used also for many other tasks, like for example clustering, anomaly detection, similarity search, etc.

And, as the main task considered here is coverage analysis, the embeddings can be used to check if a target distribution is met by a test distribution in the embedded space. Specifically, considering the Argoverse 2.0 scenarios as the target distribution, and the CARLA scenarios as the test distribution, the embeddings can be used to check if the CARLA scenarios cover the Argoverse 2.0 scenarios in the embedded space.

8 Summary

This paper presented a graph-based framework for coverage analysis in autonomous driving that represents traffic scenes as hierarchical graphs combining map topology with actor relationships. Two complementary analysis approaches were developed: subgraph isomorphism using manually defined archetype graphs for interpretable pattern-based coverage metrics, and graph embeddings via GINE networks trained with contrastive learning for similarity-based assessment, clustering, and anomaly detection.

Both approaches were validated on Argoverse 2.0 and CARLA data, demonstrating that the framework captures meaningful traffic scene structure and reveals distribution differences between real-world and simulated datasets (Section 7). Compared to approaches like TNO Streetwise [6], the framework scales efficiently without scenario-specific handling rules and naturally accommodates varying numbers of actors.

Future work will incorporate temporal information through multi-timestep graph structures and investigate automatic archetype extraction from real-world data.

The source code is publicly available at `https://github.com/tmuehlen80/graph_coverage`.

References

1. Ammann, P., Offutt, J.: Introduction to Software Testing. Cambridge University Press (2008)
2. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: International Conference on Machine Learning (ICML). pp. 1597–1607. PMLR (2020)
3. Codevilla, F., Santana, E., López, A.M., Gaidon, A.: Exploring the limitations of behavior cloning for autonomous driving (2019), https://arxiv.org/abs/1904.08980
4. Cordella, L.P., Foggia, P., Sansone, C., Vento, M.: A (sub)graph isomorphism algorithm for matching large graphs. IEEE Transactions on Pattern Analysis and Machine Intelligence **26**(10), 1367–1372 (oct 2004)
5. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: An open urban driving simulator. In: Proceedings of the 1st Annual Conference on Robot Learning. pp. 1–16 (2017)
6. Elrofai, H., Paardekooper, J.P., de Gelder, E., Kalisvaart, S., Op den Camp, O.: StreetWise: Scenario-based safety validation of connected and automated driving. Tech. rep., TNO, Helmond, The Netherlands (2018), https://publications.tno.nl/publication/34626550/AyT8Zc/TNO-2018-streetwise.pdf
7. Fey, M., Lenssen, J.E.: Fast graph representation learning with PyTorch Geometric. In: ICLR Workshop on Representation Learning on Graphs and Manifolds (2019)
8. Foretellix: Av coverage and performance metrics. https://blog.foretellix.com/2019/09/13/av-coverage-and-performance-metrics/ (September 2019), accessed: 2025-04-04
9. de Gelder, E., Paardekooper, J.P., Saberi, A.K., Elrofai, H., Camp, O.O.d., Kraines, S., Ploeg, J., De Schutter, B.: Towards an ontology for scenario definition for the assessment of automated vehicles: An object-oriented framework. IEEE Transactions on Intelligent Vehicles **7**(2), 300–314 (2022). https://doi.org/10.1109/TIV.2022.3144803
10. Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V., Leskovec, J.: Strategies for pre-training graph neural networks (2020), https://arxiv.org/abs/1905.12265
11. Jaeger, B., Chitta, K., Geiger, A.: Hidden biases of end-to-end driving models (2023), https://arxiv.org/abs/2306.07957
12. Laboratories, U.: UL 4600: Standard for Safety for the Evaluation of Autonomous Products. UL Standards (2020), standard for autonomous vehicle safety and validation
13. Menzel, T., Bagschik, G., Maurer, M.: Scenarios for development, test and validation of automated vehicles. CoRR **abs/1801.08598** (2018), http://arxiv.org/abs/1801.08598
14. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space (2013), https://arxiv.org/abs/1301.3781
15. On-Road Automated Driving (ORAD) Committee: Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. Standard J3016, SAE International (April 2021). https://doi.org/10.4271/J3016_202104, https://doi.org/10.4271/J3016_202104
16. PEGASUS: Pegasus method. Available online (2019), https://www.pegasusprojekt.de/files/tmpl/Pegasus-Abschlussveranstaltung/PEGASUS-Gesamtmethode.pdf, accessed: February 6, 2026

17. Ries, L., Rigoll, P., Braun, T., Schulik, T., Daube, J., Sax, E.: Trajectory-based clustering of real-world urban driving sequences with multiple traffic objects. In: 2021 IEEE International Intelligent Transportation Systems Conference (ITSC). pp. 1251–1258 (2021). https://doi.org/10.1109/ITSC48978.2021.9564636
18. for Standardization, I.O.: ISO 21448:2022 – Road Vehicles – Safety of the Intended Functionality (2022), standard document, ISO/TC 22/SC 32
19. Ulbrich, S., Menzel, T., Reschka, A., Schuldt, F., Maurer, M.: Defining and substantiating the terms scene, situation, and scenario for automated driving. In: 2015 IEEE 18th International Conference on Intelligent Transportation Systems. pp. 982–988 (2015). https://doi.org/10.1109/ITSC.2015.164
20. Wachenfeld, W., Winner, H.: The Release of Autonomous Vehicles, pp. 425–449. Springer Berlin Heidelberg, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-662-48847-8_21, https://doi.org/10.1007/978-3-662-48847-8_21
21. Wilson, B., Qi, W., Agarwal, T., Lambert, J., Singh, J., Khandelwal, S., Pan, B., Kumar, R., Hartnett, A., Pontes, J.K., Ramanan, D., Carr, P., Hays, J.: 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)

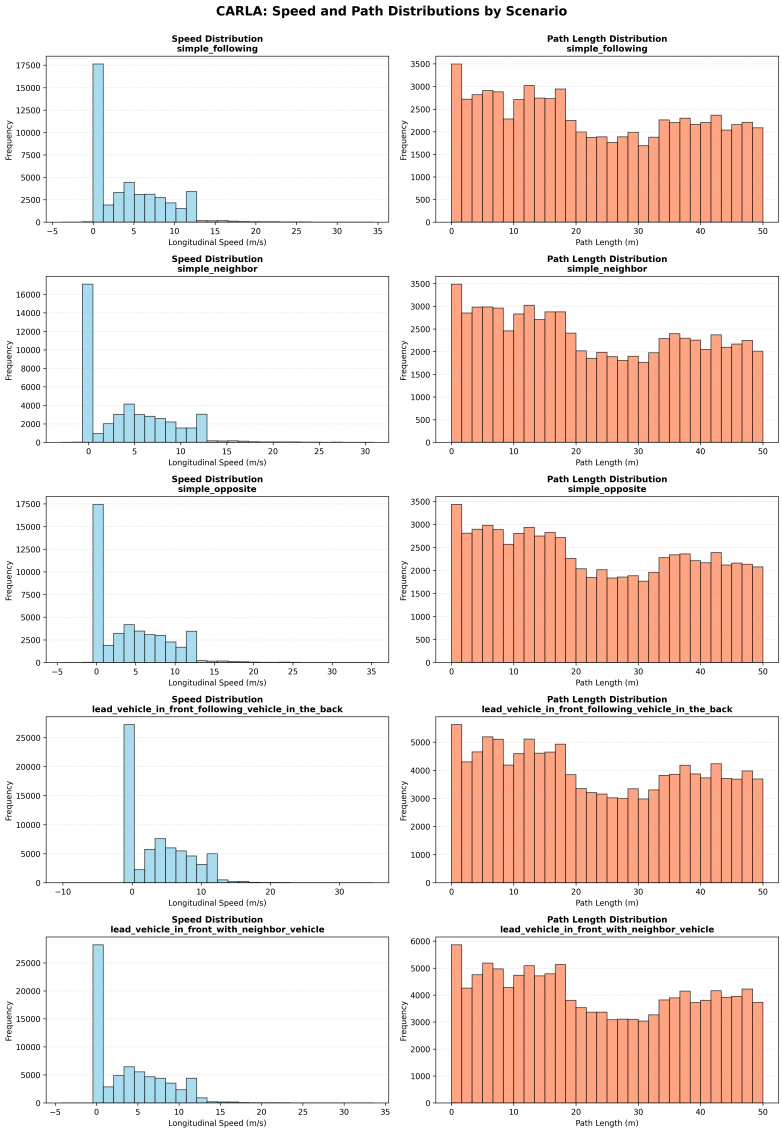


Fig. 5: Speed and path length distributions for the top 5 most common scenario archetypes in CARLA. Each row shows a different scenario with speed (left) and path length (right) distributions.

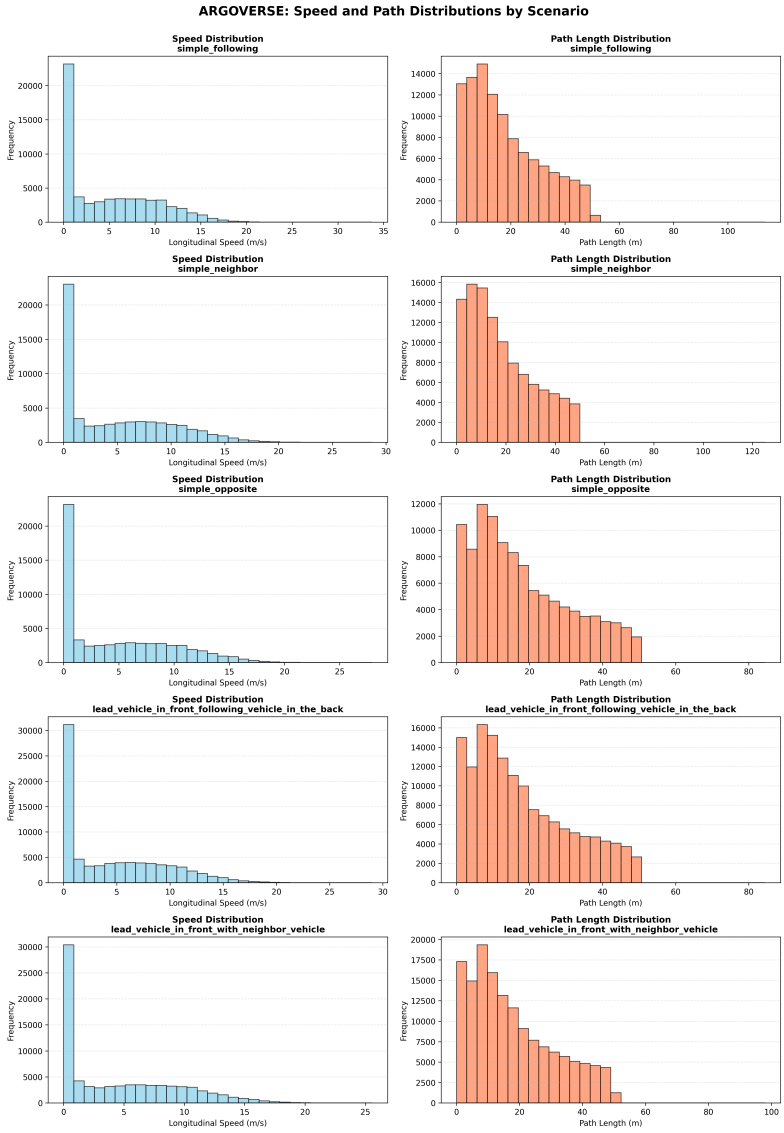


Fig. 6: Speed and path length distributions for the top 5 most common scenario archetypes in Argoverse. Each row shows a different scenario with speed (left) and path length (right) distributions.

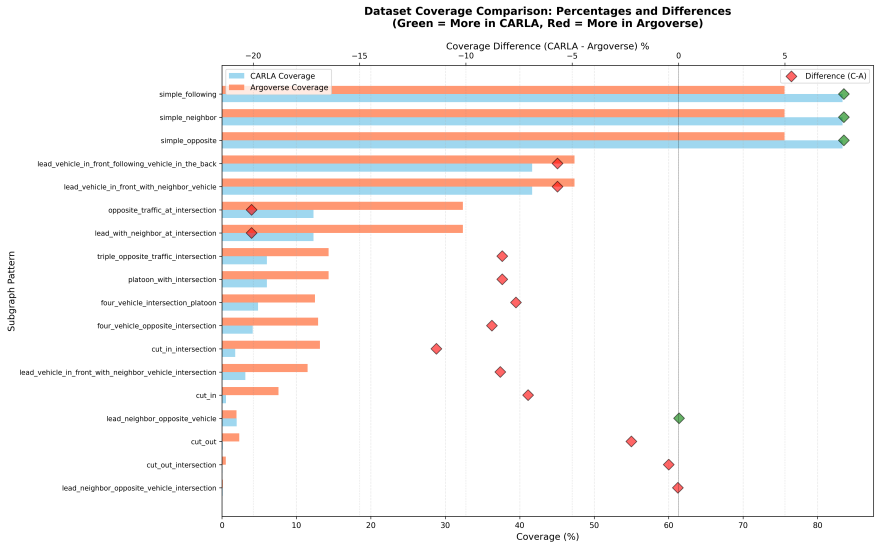


Fig. 7: Subgraph coverage comparison between CARLA and Argoverse datasets. The bar chart shows relative coverage percentages for each scenario archetype, while diamond markers indicate the magnitude and direction of coverage differences. Patterns sorted by average coverage across both datasets.

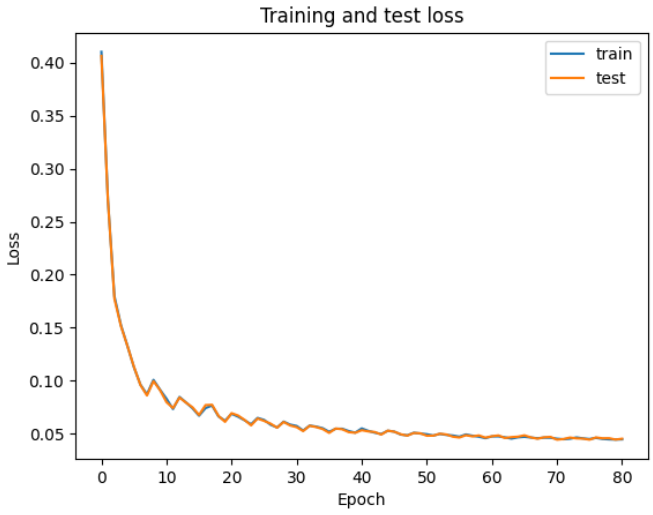


Fig. 8: Training and test loss for the graph embeddings model trained jointly on CARLA and Argoverse 2.0 data.

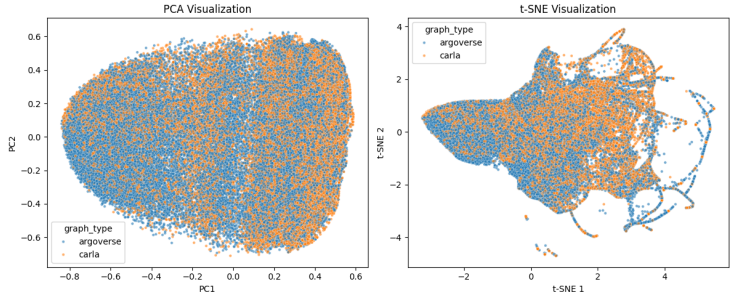


Fig. 9: PCA and t-SNE visualization of the embedding space for Carla and Argoverse 2.0 scenarios.

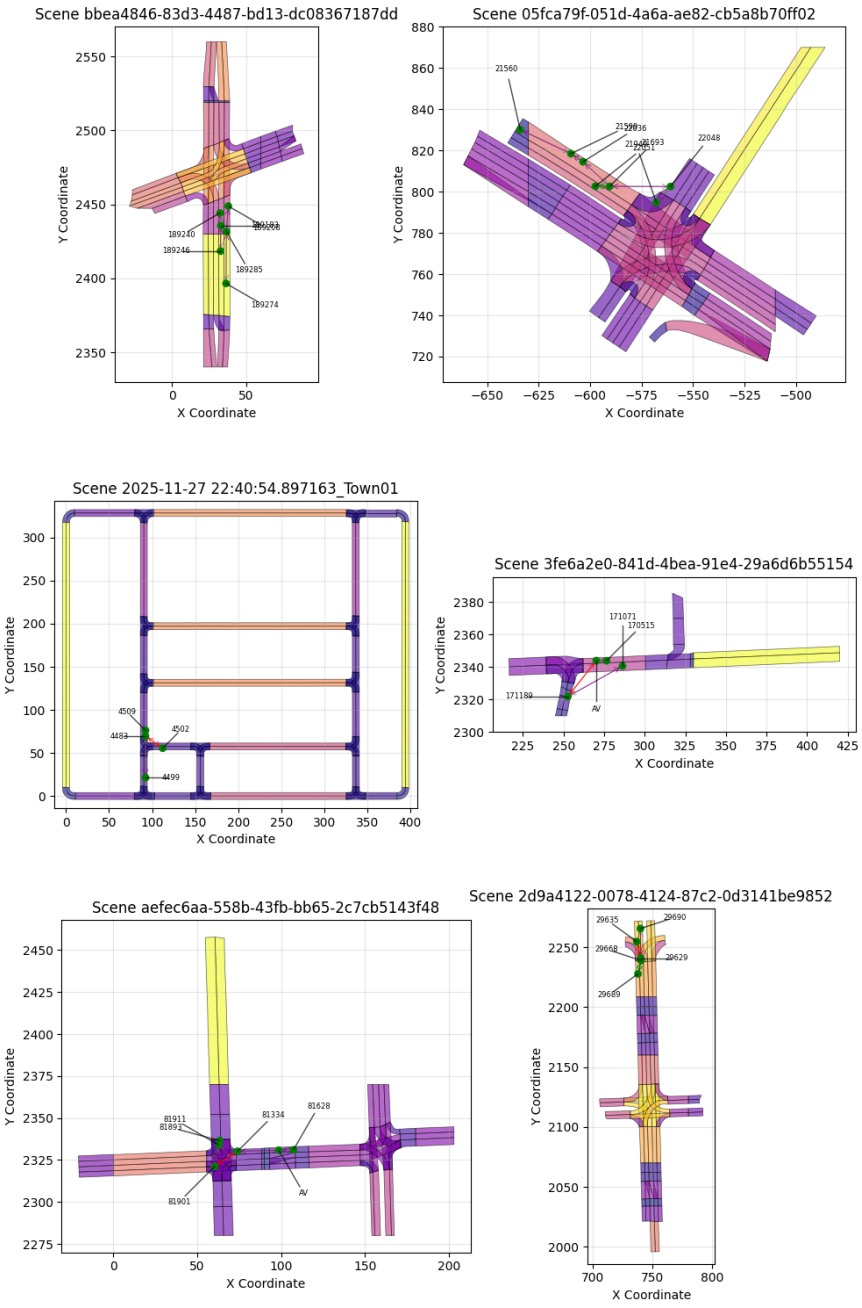


Fig. 10: Plausibility checks showing graph embedding comparisons. For randomly sampled scenarios, the most similar scenarios based on embedding distance (both Euclidean and cosine) are visualized, including comparisons between CARLA and Argoverse scenarios.