Semantic Concept Testing in Autonomous Driving by Extraction of Object-Level Annotations from CARLA

Sujan Gannamaneni
sujan.sai.gannamaneni@iais.fraunhofer.de

Sebastian Houben
sebastian.houben@iais.fraunhofer.de

Maram Akila
maram.akila@iais.fraunhofer.de

Fraunhofer IAIS

Abstract

With the growing use of Deep Neural Networks (DNNs) in various safety-critical applications comes an increasing need for Verification and Validation (V&V) of these DNNs. Unlike testing in software engineering, where several established methods exist for V&V, DNN testing is still at an early stage. The data-driven nature of DNNs adds to the complexity of testing them. In the scope of autonomous driving, we showcase our validation method by leveraging object-level annotations (object metadata) to test DNNs on a more granular level using human-understandable semantic concepts like gender, shirt colour, age, and illumination. Such an enhanced granularity, as we detail, can prove useful in the construction of closed-loop testing or the investigation of dataset coverage/completeness. Our add-on sensor to the CARLA simulator enables us to generate datasets with this granular metadata. For the task of semantic segmentation for pedestrian detection using DeepLabv3+, we highlight potential insights and challenges that become apparent on this level of granularity. For instance, imbalances within a CARLA generated dataset w.r.t. the pedestrian distribution do not directly carry over into weak spots of the DNN performances and vice versa.

1. Introduction

Verification and Validation (V&V) of Deep Neural Networks (DNNs) remains an elusive but important criterion for the deployment of such modules in safety-critical applications. Particularly for applications in computer vision (CV), or more precisely in automated driving (AD) and medical diagnostics, a lack of rigorous V&V before deployment can lead to catastrophic failures. Standard measures like mean intersection-over-Union (mIoU) or mean average precision (mAP) on test datasets are typically used as performance metrics for DNNs. However, they are not reliable Machine Learning (ML) validation methods due to the limited information they provide to the developer about the DNN behaviour. In addition to developing new metrics [17, 20, 21] to evaluate, *e.g.* uncertainty and robustness, ML developers can also be supported by providing techniques to evaluate their DNNs at a more granular level. In our work, we show that such systematic evaluation is possible using synthetic data generated from computer simulators such as CARLA [4].

In automated driving, typical computer vision tasks are object detection and semantic segmentation. For the most part, DNNs performing these tasks operate on the data from a front-facing camera of the vehicle. They are trained with real-world data that is both expensive and time-consuming to collect and label [3]. Considerations regarding the V&V of such DNNs should take into account the following two points: (i) The image space on which these DNNs operate is enormous, and there are currently no methods to estimate if any training dataset is sufficient to cover the operational domain of the task. (ii) While large-scale datasets are available for training and testing [3, 6], these datasets only contain ground-truth information for the given task without any information about the semantics present in the image. This semantic information about the scene or its objects, like the scene's brightness, weather conditions or the gender, shirt and age of a pedestrian in the scene, could help to evaluate DNNs at a more granular level. While extracting these semantics out of real-world data is time-consuming and expensive, simulations offer the flexibility to include such object-level annotations (object metadata) along with raw image and ground-truth during data generation. The goal of this work is to show that above points can be addressed by generating synthetic data plus metadata using the

CARLA simulator to enhance the validation of the DNNs. Since it is not possible to generate metadata with the current version of CARLA, we add a new sensor to the simulator to make our validation possible.

We propose to evaluate the performance of DNNs along semantic dimensions as shown in Figure 1. Note that this is only a simplistic representational sketch of our concept. We can consider the intersection-over-union (IoU) of a semantic segmentation network or, in practice, any other DNN metric on a per-pedestrian or per-image basis along one axis. The other two axes contain semantic and non-semantic dimensions. For example, consider the skin colour of the pedestrians as the semantic dimension. Intuitively, we expect to observe higher IoU in pedestrians with skin colour highly distributed in the training data. If the DNN indeed shows such behaviour, the developer could, *e.g.*, augment the training data with pedestrians of other skin colours to compensate. Similarly, we can also consider the shirt colour of a pedestrian as the semantic dimension under test. Suppose pedestrians with specific shirt colours tend to have higher IoU than others. In that case, the developer can explore the training distribution to uncover whether this is a weakness learnt from the training data. While this is a simplifying view, testing DNNs w.r.t. specific semantic dimensions can still help to detect weaknesses and provide information about coverage of training data.

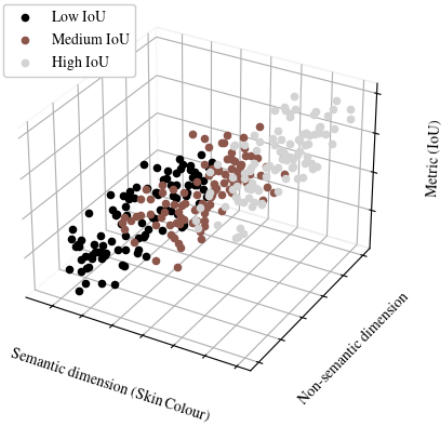


Figure 1. This is a representational sketch for semantic concept testing. One axis represents the performance metric, while the other two axes represent the semantic and non-semantic dimensions under test. The test reveals when systematic weaknesses learnt by a DNN occur along semantic dimensions (Best seen in colour).

The remainder of this paper is structured as follows: In the next section, we discuss the related work regarding testing for DNNs. In section 3, we describe the sensor add-on we developed to obtain the necessary metadata from the CARLA simulator. In section 4, we propose a potential closed-loop validation pipeline for DNNs. We describe the validation experiments and provide the results in section 5. Lastly, we provide a brief summary and conclude our work in 6.

2. Related works

For real-world validation of AD functions, Kalra and Paddock [8] estimated that the vehicles would need to be driven hundreds of millions to hundreds of billions of miles in order to demonstrate reliability with 95% confidence. In addition, with every major software update or modification, the validation step would have to be repeated. Kalra and Paddock [8] conclude that "Automated Vehicle developers cannot drive their way to safety", and alternative validation methods would be required.

Several works, see [10, 11, 13, 15, 16, 22], have proposed using simulators for training and testing of DNNs. Among these works, [16] propose a label-to-image transfer method to remove the domain gap problem by generating paired datasets to test the performance of networks trained on real-world data. Paranjape *et al*. [11] propose a simulation environment to generate scenarios for testing autonomous vehicles. [10] study the effect of combining real and synthetic datasets as a way to augment real-world data. [13, 15, 22] released synthetic benchmark datasets for DNN research.

In addition to CARLA simulator [4], several other proprietary and open-source simulators [1, 14, 18] with real-time engines are available for autonomous driving research. We use the CARLA simulator in our work for the flexibility it provides for modifying source code to add new sensors.

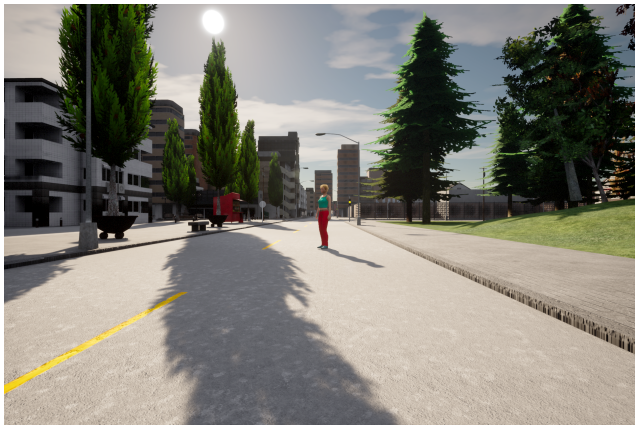
Syed *et al*. [19] also propose a validation engine approach in which parameters in the scene can be varied. Our proposed test method is similar to their approach. However, our approach grants the developer greater flexibility to control the training distribution. There is also no issue of domain gap as training, and test data are from the same domain. Parallel to our work, Lyssenko *et al*. [9] present a CARLA-based extension to perform validation of DNNs by considering the pedestrian distance from the ego vehicle as a safety relevance metric. However, they mention that their bounding box retrieval method for getting instances has some issues. Our proposed method does not make use of the bounding boxes for extracting instances.

3. Metadata generation in CARLA

CARLA simulator [4] is an open-source project developed using the UNREAL Engine [5] for the simulation of urban driving to further autonomous driving research. The simulator is set up as a server-client model in which the server is initialized as a 3D world filled with dynamic and static assets. By default, several different "maps" with pre-

positioned buildings, roads and scenery (*e.g*. hills or vegetation) are available for use. Using a Python API, multiple clients can interact with the server, create dynamic or static objects in the world, extract sensor output, and modify global parameters like weather or traffic conditions. Generating semantic segmentation datasets with CARLA is relatively straightforward by designing a client that spawns an ego vehicle with an RGB camera sensor and a predefined semantic segmentation sensor. These sensors record and save the output as the ego vehicle drives around the town. Additional clients can be used to fill the world with other participants like pedestrians and other vehicles and to change the weather dynamically. All these steps are possible out-of-the-box with the latest CARLA version (v0.9.11). However, the latest version does not include instance segmentation or metadata of the pedestrians in the scene. Although we know which pixels in an image belong to a pedestrian, we cannot link the pedestrians to their digital asset, *i.e*. we cannot generate pedestrian-specific metadata. As also discussed in Lyssenko *et al*. [9] this lacking separability and identifiability of individual pedestrians can be challenging for some more advanced analysis (depth-based in their case) and requires a workaround.

In this work, we introduce a simplistic pedestrian instance detection sensor to the CARLA simulator. This sensor allows us to link the pedestrians in the RGB camera to their digital assets. Our pedestrian detection sensor builds upon the radar sensor in CARLA. Since new sensors cannot be added using the Python API, we modified the CARLA source code to make the addition. The code for our sensor extension is publicly available[1]. Since the sensor has to detect pedestrians in the field of view (FOV) of the RGB camera sensor, we set the horizontal and vertical FOVs similar to the RGB camera. The UNREAL Engine provides a function `UWorld::LineTraceSingleByChannel` that traces a ray from the point of origin to the first blocking hit. While the radar sensor uses this function to simulate radar behaviour, we use it to get the instances of the first blocking pedestrian in front of the ego vehicle. An `instance id` in CARLA is a unique identification number assigned to an asset when the asset is spawned in the world for that specific run. These `instance ids` provide the link to the digital asset that the pedestrian belongs to. We save pedestrians instances detected per run of the sensor. In Figure 2, a sample RGB camera output and its corresponding semantic segmentation sensor output are shown. While these two could be generated easily, with our sensor, we can also extract metadata of the pedestrian and the scene as shown in Listing 1.



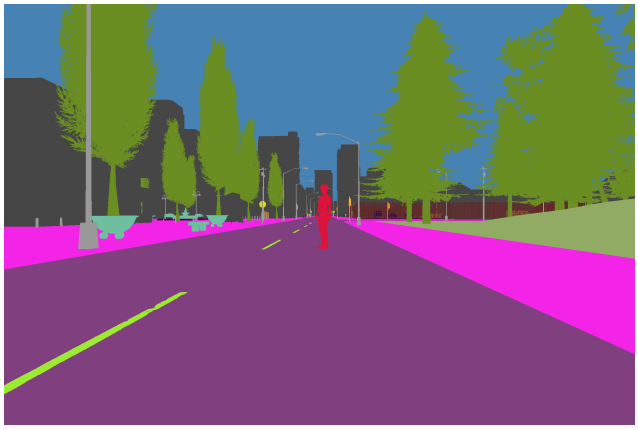


Figure 2. Here we have a sample RGB and semantic segmentation image from our generated CARLA dataset. While these are generated with minimal effort using the simulator, the corresponding metadata shown in Listing 1 was extracted using our extension.

```
{"Pedestrian_data": {
  "instance_id": 220,
  "pedestrian_asset_id": 0005,
  "world_x_coordinate": 190.0",
  "world_y_coordinate": 147.0,
  "gender": "female",
  "shirt_colour": "green",
  "pant_colour": "red",
  "skin_colour": "white",
  "age": "adult",},
  "Global_data": {
    "sun_angle":30.0,
    "sun_azimuth_angle":250.0,
    "fog_density":10.0,}}
```

Listing 1. Example of image level JSON file describing pedestrian and global metadata.

[1]https://github.com/sujan-sai-g/Semantic-Concept-Testing-using-CARLA

4. Validation Pipeline

As shown in Figure 1, our goal is to evaluate DNN behaviour along semantic dimensions. With our sensor extension to CARLA, such granular testing is made possible. A potential application could be closed-loop testing or similar approaches which aim at an iterative improvement of a given DNN. Figure 3 shows a high-level concept of such a closed-loop validation approach that builds on semantic testing and includes a feedback loop where the DNN behaviour is both studied and modified with the aim of improvement. The training and test data are initially generated by following some Operational Design Domain (ODD) parameters defined for the task. The DNN is trained on that preliminary training data and tested with the preliminary test datasets.

The test scenarios can be of two types: (i) a holdout test dataset with similar distribution to training data. (ii) specially designed test datasets where only a specific semantic concept is varied, *e.g*. the pedestrian asset changes, but everything else remains constant. The trained DNN is tested on these test datasets, and per-pedestrian or per-image performance metrics are extracted from the DNN. For our experiments, the IoU of a pedestrian is the metric under consideration.

A semantic testing setup as proposed should ideally uncover systematic weaknesses in the DNN. Since the DNNs learn from the training data, the goal is to fix systematic weaknesses by modifying/augmenting the training data. The DNN developer can query more data from a simulator, *e.g*. CARLA, with the required semantic properties to augment the overall training dataset and improve the DNN. After retraining, the overall process would be repeated to check if the DNN has indeed overcome the systematic weaknesses uncovered in the last run. By generating a large dataset of scenarios that are critical for safety, a DNN developer can iteratively test the DNN until performance reaches the required levels on all test scenarios. When a DNN fails in any given scenario, the developer can detect the possible weakness. While adding the failed scenarios directly to the training would overcome the weakness, it would also imply a direct (over-)optimization on the test invalidating its result in the next iteration. Therefore, identifying lacking training data is a crucial ingredient to any form of closed-loop approach. The granularity of the metadata and flexibility of the engine might play a vital role in this. For example, having the actual colour of the pedestrian shirt (like blue or black) as metadata instead of broader categories (such as bright or dark) could be much more beneficial for the validation and interpretation of results.

This work investigates how strongly results from tests are linked to the training data distribution when queried based on semantic categories. This could give insight into the feasibility and challenges of such a closed-loop or iterative testing and development approach. A strong connection between discovered weak spots and the original training distribution would afford an "easy" remedy by altering the latter.

5. Experiments

5.1. Dataset

The CARLA simulator has multiple maps available with different town architectures pre-built. We choose "Town02", a map containing a small town with narrower roads, for our experiments. These narrower roads of the town lead to more instances of pedestrians being closer to the ego vehicle and therefore larger in size, which in turn leads to better performance of the model on the pedestrian class compared to models trained on data generated from a larger town with relatively wider roads. The semantic segmentation sensor maps the image to 23 classes, the default mapping available in version 0.9.11 of the CARLA simulator, which is similar to the Cityscapes [3] class mapping. To generate the training data, we wrote a CARLA client using the Python API that spawns an ego vehicle at a random point in the town and drives around automatically. Furthermore, we make use of CARLA simulator example code `spawn_npc.py` to fill the world with 150 pedestrians and 30 vehicles. The pedestrians, for example, are spawned randomly on a set of predefined pavement locations.

The data is generated in different sequences. Each sequence is initialized with the same initial settings, like the number of AI actors and weather, to ensure that the data distribution remains the same. But, the spawn points of AI actors and the ego vehicle are random for each sequence.

The ego vehicle has an RGB sensor, a semantic segmentation sensor, and the pedestrian detection sensor we developed. The RGB and semantic segmentation sensors each have a resolution of 1920×1280 with 90° horizontal and 60° vertical field of view (FOV). The pedestrian detection sensor projects beams with horizontal FOV of 95° and vertical FOV of 50° using 200,000 points per scene. The maximum range of the beam is 200 metres, and the points are projected to form a cone with a rectangular base. These values were obtained heuristically to maximize pedestrian detection without increasing the points per scene, which is computationally expensive. If there are only a few pixels in a scene belonging to a pedestrian due to occlusion, our sensor could fail to detect and extra the metadata. While we can avoid this by increasing the points per scene, we neglect these corner cases for our preliminary experiments.

We generated 10,099 images, ground-truth and corresponding pedestrian and global metadata. From these generated data, we use 7394 for training, 1303 for validation and 1402 for testing. The validation data is used for hyper-parameter tuning and for choosing the best perform-

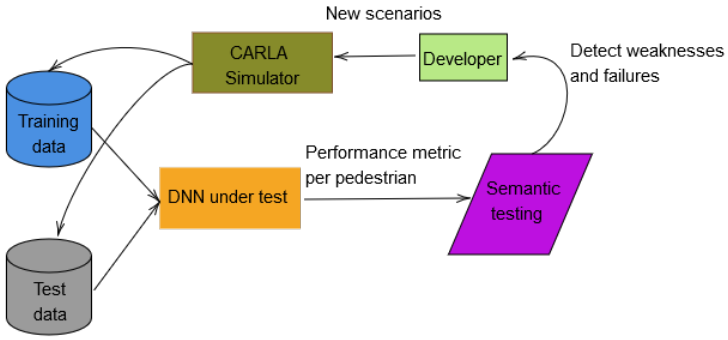


Figure 3. The proposed concept for a closed-loop validation method. The validation is performed by leveraging metadata generated from our CARLA sensor extension.

ing model. The test and validation data have a similar distribution to the training data. All three datasets have the default weather setting in CARLA. Furthermore, we generated the following separate datasets with specific scenarios for semantic testing.

- **Varying position:** For this experiment, we use a single isolated pedestrian (asset 0015) for each image and vary only its position and rotation while keeping all other semantic properties the same, compare Figure 2 for a sample image. We measure the pedestrian's position relative to the ego vehicle: x varies between +7 meters (right) to -6 (left) and depth y changes between 10 and 35 meters. The depth is deliberately limited to 35 as we consider these pedestrians as safety-relevant for our experiment. We chose -6 on the left side as obstacles on the pavement lead to collisions while spawning the pedestrian. Each position is sampled five times with varying orientations of the asset. In total, we generate 2046 images for this test dataset.

- **Varying digital asset:** The second experiment uses the same setup as the first but fixes the pedestrian's position at the image centre, again see Figure 2. Instead of the position, which remains fixed, we sample all 26 CARLA pedestrian assets (20 adults, six children). Again, we vary the orientation of each asset five times. We generate 181 images for this test dataset.

  The setup is used to generate a second dataset of identical size and making, except that the daytime is set to nighttime to perform an extrapolation experiment.

5.2. DNN Training

We use a DeepLabv3+ [2] with a randomly initialized Resnet-101 [7] backbone to perform the semantic segmentation task. The entire DNN is trained from scratch with a stochastic gradient descent (SGD) optimizer with momentum 0.9, weight decay of 5e-4 and a learning rate of 5e-3, along with a poly learning rate scheduler. We train for fifty epochs with a mini-batch size of two. We use PyTorch [12] for our implementation. We resize the images from 1920×1280 to 1080×1080. Further data augmentation techniques like random horizontal flipping (rate=50%) and random scaling are used. In addition, the images are normalized with mean=(0.485, 0.456, 0.406) and standard deviation=(0.229, 0.224, 0.225). Mean intersection-over-union (mIoU) is chosen as the performance metric.

For comparison, we train five models with different random seeds and perform our experiments on all of these five models. The best performing model for each seed is chosen based on the mIoU on the validation data. Each model achieved at least 70% mIoU over all classes on a holdout test dataset with a similar image distribution as the training set. However, for our experiments, we only focus on the pedestrian class and its corresponding IoU. To extract IoU at an instance level of individual pedestrians, we perform a post-processing step. First, using bounding-box information, we crop rectangular patches of each pedestrian from both the semantic segmentation ground-truth and the DNNs prediction image. We then calculate the IoU of classes inside this cropped patch. These steps are repeated for all pedestrians in an image to get the instance IoU. Note that this can lead to faulty IoU calculation when pedestrians overlap in the cropped images.

5.3. Results

5.3.1 Generalization gap

With the availability of metadata, we can explore the generalization gap of the DNN on training and test data at a more granular level. Instead of evaluating a DNN by averaged measures such as mean IoU, we can resolve the metric along semantic concepts like pedestrian asset type or pedestrian skin colour. In Figure 4, we see the IoU of the pedes-

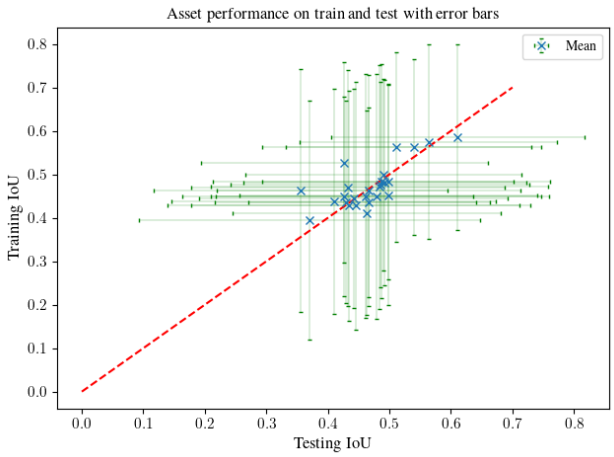


Figure 4. This figure shows the generalization gap between the train and test datasets for DeepLabv3+. The dashed red diagonal indicates an ideal case where train and test performance are equal. The points represent the average IoU for each pedestrian asset, along with the respective standard deviation given by the green lines.

trians grouped by pedestrian asset type in training and test datasets. A diagonal line is provided to show ideal DNN behaviour where train and test performance is identical. Points that lie above have a positive generalization gap as testing performance is lower than training performance. Points below the diagonal have a negative generalization gap as testing outperforms training performance at these points. Overall, as indicated by the bars representing the standard deviation for the IoU of each pedestrian asset, performance varies strongly even for a fixed asset, potentially due to other parameters such as distance to the camera. These fluctuations, paired with the comparatively small size of the (per asset) test dataset, might explain the otherwise uncommon negative generalization gaps.

5.3.2 Tests with special scenarios

With the flexibility of synthetic data simulators, we can also vary specific parameters in a scene to evaluate the performance of a DNN. We generated scenario-specific test datasets to test the DNN behaviour. We perform the experiments based on the scenarios mentioned in 5.1.

The results of the experiment with **varying position** are shown in Figure 5. This plot shows the top-down view of the scene in Figure 2, where each point represents the position of the pedestrian. The origin of the x-axis is the centre of the ego vehicle, and positive (negative) values indicate a shift of the asset to the right (left, respectively). The y-axis represents the depth of the pedestrian in the image, *i.e.* how far the pedestrian is away from the ego vehicle in the direction of travel. The pedestrian shown in Figure 2 is at position x=0, y=12. Red colour indicates higher IoU while blue indicates lower IoU, averaged over results from five models, each adopting over five orientations for each position. Intuitively, we expect pedestrians closer to the centre of the image, *i.e.* closer to the ego vehicle, to have higher IoU. While this is true, the IoU is not symmetric w.r.t. to x. To the right, performance decreases for x$\geq$ 6 while for -6 IoU is higher at lower values of y. One potential explanation for this could be that the pedestrian at +6 and +7 are positioned on the grass. As the training data is generated with dynamic actors where pedestrians movements are predefined, pedestrians are mostly situated on pavements and crossings. This difference between train and testing data distribution based on position could be a reason for the lower IoU.

The data is generated without placing pedestrians at position x=-7 as the obstacles on the pavement like the street lamp cause collisions when spawning pedestrians. Spawning pedestrians on the road can sometimes lead to unexpected behaviour in the CARLA simulator, as seen by the lack of data points at y=27. The documentation recommends to spawn at specific predefined points to avoid this situation. However, due to the nature of our experiment, we spawn the pedestrians on the road as well as the pavement and grass.

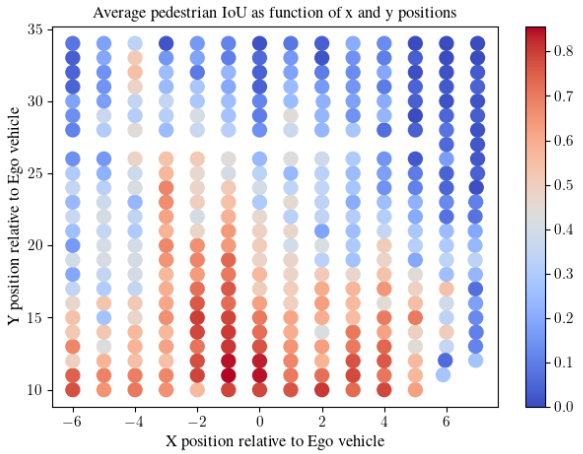


Figure 5. Top-down view of the scene, as seen in Figure 2, where each point represents one pedestrian position. The ego vehicle is at 0 on the x-axis. +6 on the x-axis means that the pedestrian is placed on the right side of the image, and -6 indicates the left side. The y-axis represents the depth in the image. The red colour indicates higher IoU, and blue indicates lower IoU (Best seen in colour).

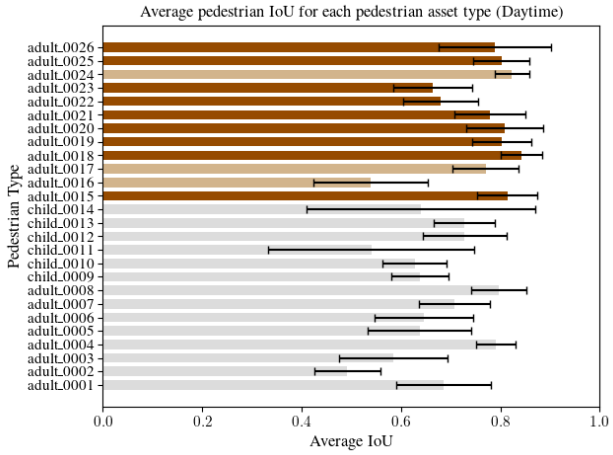
In the experiment with **varying digital asset**, we keep the position of the pedestrian fixed, and the weather remains the same as the training distribution (daytime). The pedestrian position relative to the ego vehicle is x=0, y=12. Figure 6 shows the average IoU grouped by asset type as bar

graph.

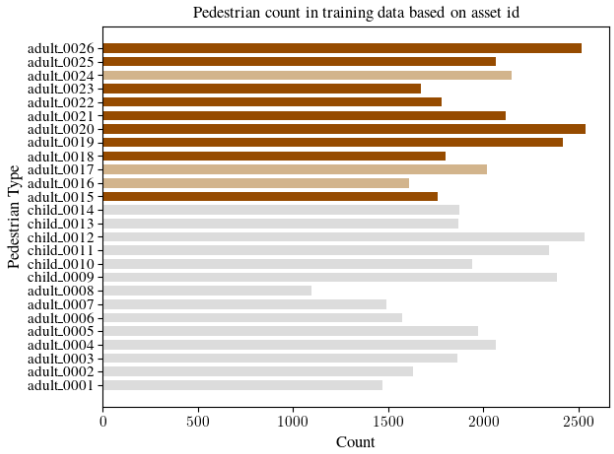
We order the asset types by asset id numbers from CARLA along with the age meta-information. In addition, we also include the skin colour of the pedestrian (brown, tanned, white) as the colour of the bars. We see that the performance among the assets is not the same, although all other semantic properties like scene and position are kept constant. We introduce variance within our five experiments by randomly varying the pedestrian orientation in each run. The standard deviation for some assets, *e.g.* assets 11 and 14, are almost double compared to other assets such as ids 4, 19, and 24. Most of the pedestrians with white skin colour have lower performance than brown and tanned. To study if these particular assets are underrepresented in the training data, we looked at the count of pedestrians in training data grouped by asset type, see Figure 6. We see that while asset id 26 is both strongly represented and has a high IoU, the same does not hold for asset 8, which despite significantly smaller representation, has a comparable IoU. This shows that the distribution in training data cannot be the only parameter impacting IoU. For instance, asset 12, a child whose distribution in training is equivalent to the adult 26, shows reduced performance compared to the adult. In CARLA, the child assets are around half the height of the adult assets. This leads to a relative reduction in pixel area for the child asset if both assets, such as here, are compared at the same distance, which might also impact the performance of the DNN. However, further experiments are needed to pinpoint more relevant semantic dimensions due to which there is a difference in IoUs for different assets.

5.3.3 Extrapolation to unseen data

The previous experiments followed a similar distribution of the test and train datasets as the overall simulation parameters such as the town map, daylight and weather were kept the same. To study the extrapolation capability of DeepLabv3+, we extended the **varying digital asset** experiment by introducing nighttime as a semantic change as shown in Figure 7. Looking at the IoU grouped by asset type as before, we see a significant reduction in performance for all assets in Figure 8. However, the drop in performance is not distributed homogeneously. For example, while asset 15 had high IoU in the daytime, the reduction in performance is more significant than other assets like 26 and 4. Similarly, asset 20 has a significant reduction compared to daytime performance. Especially for the child assets, the IoU is consistently less than 5%. We place the pedestrians under a streetlight in this experiment, which might lead to slightly better performance than other placements. For *e.g.*, this experiment could be extended to changing two semantic dimensions by placing pedestrians on the grass as an additionally challenging position while keeping nighttime to study DNN behaviour across two semantic changes.



(a) Pedestrian type vs IoU on test data



(b) Pedestrian type distribution in training data

Figure 6. These plots show the semantic dimension pedestrian type and corresponding average IoU for the test scenario **varying digital asset** and the distribution of pedestrian assets in the training data. The pedestrian asset id plus age are shown as y-axis ticks. The colour of the bar indicates the skin colour of the pedestrian (Best seen in colour).

6. Conclusion

We presented two contributions: At first, we showed how the CARLA simulator could be extended by an additional sensor that allows for logging of granular scene meta-information with limited effort. We demonstrated this by resolving where and which type of pedestrian asset, *i.e.* the corresponding 3D model, is situated in an obtained image. While this is a technical contribution, we also showcased

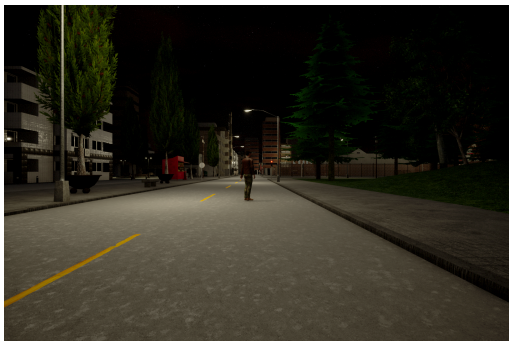


Figure 7. A sample image for **varying digital asset** experiment at nighttime.

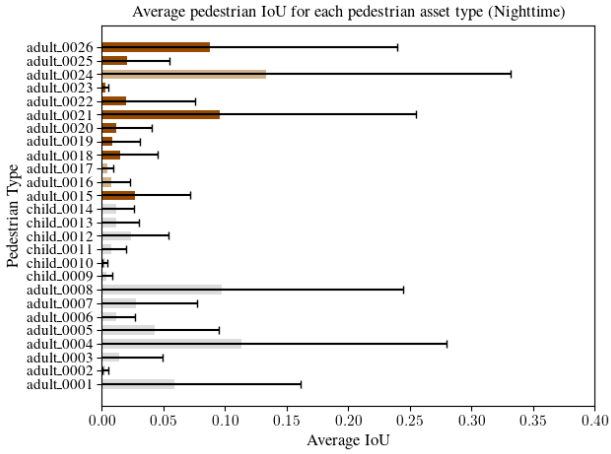


Figure 8. The average IoU is calculated by grouping by pedestrian asset type for the nighttime experiment. In comparison to Figure 6, the performance is consistently lower. The pedestrian asset id plus age are shown as y-axis ticks. The colour of the bar indicates the skin colour of the pedestrian (Best seen in colour).

how we can use such metadata to test DNNs along semantic dimensions. This provides an effective way to check for the presence of weaknesses or learnt shortcuts. For instance, we showed in Figure 4 that, given a trained DeepLabv3+ segmentation DNN, specific pedestrian assets are statistically better resolved (*i.e*. possess consistently higher IoU) than others when comparing performance on training and holdout test datasets. Using CARLA further to investigate specific test scenarios, we demonstrated in Figures 5 and 6 that performance differs even more significantly among assets and semantic dimensions such as position in a given scene. Similar statements hold for the case of extrapolation as shown in Figure 8, where observed performance losses were asset-specific, a property hard to resolve without good control over the metadata.

The results from Figure 6 also showcase the problems of identifying relevant semantic dimensions and ensuring that they are indeed correct. For example, while age in the simulation might be a good predictor for IoU performance, the result might be correlation rather than causality. Instead, majority of the effect is likely attributed to the physical size of the pedestrian, which is not fully determined by age. Therefore, choosing the appropriate semantic dimension is essential to perform validation. It is potentially impossible to find and test all potentially relevant semantic dimensions, *i.e*. to achieve completeness, due to their large number and difficulties to extract and generate them even in simulation. Nonetheless, even studying a limited set of metadata and resulting performance measures can point towards potential failure cases or weak spots.

Linking the experimental results to the training data distribution appears to be non-trivial. On the one hand, unseen scenarios such as nighttime lead to tremendous performance deterioration. On the other hand, shifts in relative distributions among different assets do not directly translate to observable performance differences. We highlight that we did not leverage the complete metadata available and provide our experiments as a proof-of-concept. Also, modifications to the training distribution, *e.g*. adding night scenes or further skewing the pedestrian distribution and retraining and re-evaluating the network based on such modifications, is left for future work.

Considerations in this direction are important as it is a common, and to some extent justified, assumption that DNNs as data-driven models suffer from weaknesses that stem from incomplete or lacking training data. Many improvements aim to compensate for or find such missing data, *e.g*. via closed-loop testing. In this regard, our CARLA sensor extension offers a principled way to study such concerns in the future.

Acknowledgment

The research leading to these results is funded by the German Federal Ministry for Economic Affairs and Energy within the project “Methoden und Maßnahmen zur Absicherung von KI basierten Wahrnehmungsfunktionen für das automatisierte Fahren (KI-Absicherung)”. The authors would like to thank the consortium for the successful cooperation. The work by author Sebastian Houben has been supported by the German Federal Ministry of Education and Research as part of the Competence Center Machine Learning Rhine-Ruhr ML2R (01—S18038ABC).

References

[1] Apollo: An open autonomous driving platform. `https://github.com/ApolloAuto/apollo`, 2017.

[2] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolu-

tion, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2017.

[3] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3213–3223, 2016.

[4] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Proceedings of the Conference on Robot Learning*, pages 1–16. PMLR, 2017.

[5] Epic Games. Unreal engine.

[6] Jakob Geyer, Yohannes Kassahun, Mentar Mahmudi, Xavier Ricou, Rupesh Durgesh, Andrew S Chung, Lorenz Hauswald, Viet Hoang Pham, Maximilian Mühlegg, Sebastian Dorn, et al. A2d2: Audi autonomous driving dataset. *arXiv preprint arXiv:2004.06320*, 2020.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[8] 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, 2016.

[9] Maria Lyssenko, Christoph Gladisch, Christian Heinzemann, Matthias Woehrle, and Rudolph Triebel. From evaluation to verification: Towards task-oriented relevance metrics for pedestrian detection in safety-critical domains. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 38–45, 2021.

[10] Farzan Erlik Nowruzi, Prince Kapoor, Dhanvin Kolhatkar, Fahed Al Hassanat, Robert Laganiere, and Julien Rebut. How much real data do we actually need: Analyzing object detection performance using synthetic and real data. *arXiv preprint arXiv:1907.07061*, 2019.

[11] Ishaan Paranjape, Abdul Jawad, Yanwen Xu, Asiiah Song, and Jim Whitehead. A modular architecture for procedural generation of towns, intersections and scenarios for testing autonomous vehicles. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, pages 162–168, 2020.

[12] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Proc. of NeurIPS*, pages 8024–8035, Vancouver, BC, Canada, Dec. 2019.

[13] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *Proceedings of the European Conference on Computer Vision*, pages 102–118. Springer, 2016.

[14] Guodong Rong, Byung Hyun Shin, Hadi Tabatabaee, Qiang Lu, Steve Lemke, Mārtiņš Možeiko, Eric Boise, Geehoon Uhm, Mark Gerow, Shalin Mehta, Eugene Agafonov, Tae Hyung Kim, Eric Sterner, Keunhae Ushiroda, Michael Reyes, Dmitry Zelenkovsky, and Seonman Kim. SVL Simulator: A high fidelity simulator for autonomous driving. *arXiv preprint arXiv:2005.03778*, 2020.

[15] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3234–3243, 2016.

[16] Julia Rosenzweig, Eduardo Brito, Hans-Ulrich Kobialka, Maram Akila, Nico M Schmidt, Peter Schlicht, Jan David Schneider, Fabian Hüger, Matthias Rottmann, Sebastian Houben, and Tim Wirtz. Validation of simulation-based testing: Bypassing domain shift with label-to-image synthesis. In *Proceedings of the IEEE Intelligent Vehicles Symposium, Workshop on Ensuring and Validating Safety for Automated Vehicles*, 2021.

[17] Timo Sämann, Peter Schlicht, and Fabian Hüger. Strategy to increase the safety of a DNN-based perception for HAD systems. *arXiv preprint arXiv:2002.08935*, 2020.

[18] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, 2017.

[19] Qutub Syed Sha, Oliver Grau, and Korbinian Hagn. DNN analysis through synthetic data variation. In *Proceedings of the ACM Computer Science in Cars Symposium*, pages 1–10, 2020.

[20] Georg Volk, Jörg Gamerdinger, Alexander von Bernuth, and Oliver Bringmann. A comprehensive safety metric to evaluate perception in autonomous systems. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8. IEEE, 2020.

[21] Oliver Willers, Sebastian Sudholt, Shervin Raafatnia, and Stephanie Abrecht. Safety concerns and mitigation approaches regarding the use of deep learning in safety-critical perception tasks. In *Proceedings of the International Conference on Computer Safety, Reliability, and Security*, pages 336–350. Springer, 2020.

[22] Magnus Wrenninge and Jonas Unger. Synscapes: A photorealistic synthetic dataset for street scene parsing. *arXiv preprint arXiv:1810.08705*, 2018.