On the Choice of Data for Efficient Training and Validation of End-to-End Driving Models

Marvin Klingner* Konstantin Müller* Mona Mirzaie
Jasmin Breitenstein Jan-Aike Termöhlen Tim Fingscheidt
{m.klingner, konstantin.mueller, mona.mirzaie
j.breitenstein, j.termoehlen, t.fingscheidt}@tu-bs.de

Technische Universität Braunschweig, Braunschweig, Germany

Abstract

The emergence of data-driven machine learning (ML) has facilitated significant progress in many complicated tasks such as highly-automated driving. While much effort is put into improving the ML models and learning algorithms in such applications, little focus is put into how the training data and/or validation setting should be designed. In this paper we investigate the influence of several data design choices regarding training and validation of deep driving models trainable in an end-to-end fashion. Specifically, (i) we investigate how the amount of training data influences the final driving performance, and which performance limitations are induced through currently used mechanisms to generate training data. (ii) Further, we show by correlation analysis, which validation design enables the driving performance measured during validation to generalize well to unknown test environments. (iii) Finally, we investigate the effect of random seeding and non-determinism, giving insights which reported improvements can be deemed significant. Our evaluations using the popular `CARLA` *simulator provide recommendations regarding data generation and driving route selection for an efficient future development of end-to-end driving models.*

1. Introduction

Towards End-to-End Deep Driving: In recent years there has been a steady trend towards higher automation levels in applications such as autonomous driving. Taking a look at the representative development in this area, past driving assistance systems have mainly been enabled by classical image processing techniques, such as using the Canny edge detector [8] with suitable post-processing to extract lanes and facilitate a lane keeping assistant [31]. Current systems [3] often still partially rely on such model-based algorithms, but also make use of data-driven machine learning models, e.g., for advanced environment perception tasks such as scene segmentation [4, 12, 28], or behavior prediction of traffic participants [25, 38]. Latest developments, however, envision end-to-end trainable deep driving algorithms [6, 36] driven solely by high amounts of data with high-dimensional sensor measurements as input and driving signals as output as shown in Fig. 1. As data is one of the main factors for such methods' success, it is essential to understand the influence of different data design choices.

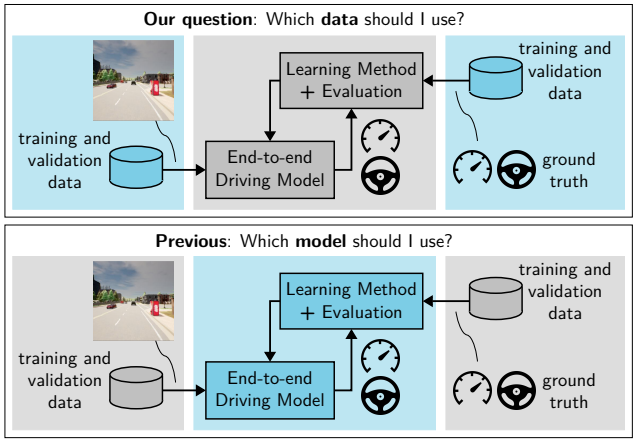


Figure 1: **General concept**: We investigate which data one should use to efficiently train and validate end-to-end driving models, while previous works often focused on improving models and learning methods. Blue parts in the figure identify components, optimized by many other works (bottom) and our work (top).

Training of Deep Driving Models: The core idea of end-to-end trainable driving algorithms [6] is to remove hand-crafted intermediate representations as it is nearly impossible to design a complete set of features that are important for the driving functionality. Accordingly, such end-to-end trainable models are supposed to learn an optimal representation on their own, where the learning success

*indicates equal contribution

is highly dependent on the chosen training data. Surprisingly, despite this fact current works often rather propose alternative datasets [17, 20, 43] instead of comparing the effect of choosing different training data. Moreover, the focus of current works is often rather on comparing different input/output representations [13, 41], learning methods [16, 26], or network architectures [24, 36], cf. Fig. 1, bottom part. In this work, we provide analysis and recommendations regarding training data and its limitations when keeping all other aspects of the deep driving model and learning method fixed, cf. Fig. 1, top part.

Evaluation of Deep Driving Models: Regarding evaluation of deep driving models, one can in general distinguish between open-loop evaluation (e.g., as in [5, 22]), where the model's predictions are compared to those of an expert in an offline fashion, and closed-loop evaluation (e.g., as in [20, 17, 36, 13]) where the driving policy is deployed and its driving quality is measured. While Codevilla et al. [15] have shown that offline metrics correlate badly to driving quality, it is still rarely investigated, which kind of closed-loop evaluation setting is well-suited to measure driving quality. In this work, we investigate a large variety of closed-loop evaluation settings using the CARLA simulator [20]. As a result, we provide guidance regarding a suitable validation design for end-to-end deep driving such that the validation performance generalizes well to the test performance, and a well-performing training checkpoint can be chosen. Furthermore, the CARLA simulator is subject to significant non-determinism such that evaluations cannot be carried out in a deterministic fashion. This actually reflects a real experimental setting quite well, where the same evaluation can also never be deterministically executed twice. As this phenomenon influences essentially all current works in the field, we investigate this effect to derive insights as to when a reported improvement is actually meaningful.

Contributions: To sum up, our contributions include the following. Firstly, we investigate the effect of varying amounts of training data on the final driving performance of end-to-end deep driving models. Secondly, we provide an analysis w.r.t the limitations of currently used training data in the domain of end-to-end deep driving. Thirdly, by correlation analysis, we provide recommendations for a well-generalizing validation of driving models. Finally, we investigate random seed dependency and non-determinism in current end-to-end deep driving models, which provides insights into the meaningfulness and comparability of reported improvements in end-to-end deep driving.

2. Related Work

In this section we discuss related work on training and evaluation of end-to-end deep driving models.

Training of End-to-End Deep Driving Models: The initial work of Bojarski et al. [6] facilitated much research in end-to-end deep driving. As their simple imitation learning approach could not handle challenging urban driving scenarios, subsequent works proposed several improvements: Firstly, more advanced learning methods were used. Some important examples are the incorporation of navigational commands by conditional imitation learning [16], the application of reinforcement learning techniques [26, 44, 10], and a two-stage training, where first a privileged expert is trained whose knowledge is subsequently transferred to the driving agent [11, 51]. Constrained highway scenarios were even approached by inverse reinforcement learning [39, 42], where the reward is not manually designed but optimized. Secondly, improved network architectures making use of, e.g., long short-term memory units [48] or self-attention [24] have been presented. Furthermore, multi-task networks with auxiliary tasks [46, 50], in particular with semantic segmentation [9] benefit the end-to-end driving task. The fusion of different input modalities such as camera and LiDAR has also been proposed [36]. Thirdly, different input and output representations have been proposed. Cai et al. [7] show that a driving model can also be trained based on LiDAR data, while other approaches replace the direct steering and speed output by affordances [41] or waypoints [13] and a subsequent PID controller, or even a probabilistic output [2]. Fourthly, the transfer of deep driving models to real data has been investigated as many deep driving models are trained and validated on simulated data and generalize poorly to real data [33]. For example GAN-based style transfer of real images to the virtual domain [32] or the segmentation domain [49] has been proposed. Finally, some works aim at improved interpretability of deep driving models by using the attention mechanism [47, 18] or an intermediate semantic representation [40].

While many aspects influencing the training of end-to-end deep driving have been thoroughly investigated, the influence of using different training sets has not been subject to a structured investigation so far, which we address with this work. Approaches introducing new datasets [17, 20, 43, 48] usually only compare different models on new data but do not show the influence induced by different amounts of training data. Other works investigate the usage of online data selection techniques [35, 19] or the adaptation to out-of-distribution scenes [21]. However, these works are limited in their improvement as more online adaptation also involves catastrophic forgetting [29] such that our recommendations for training set design are a vital component for well-performing driving policies. Moreover, we show that one of the main limitations of currently trained driving models is not the learning approach but the training data generated by a far-from-optimal "expert" driving policy.

Evaluation of End-to-End Deep Driving Models: Approaches to end-to-end deep driving usually train their models using recorded driving sequences with corresponding

driving actions from a human driver or an expert driving policy. Datasets such as BDD [48] or Waymo [43] collected in real environments often already provide a wide variety of situations. However, evaluation of models in a test car is usually not possible and neural simulators of "real" data [27] still lack performance and diversity, such that only the deviation between the predicted and the ground truth action can be measured at each time step [5, 22]. Notably, Codevilla et al. [15] show that such offline metrics correlate badly to actual driving quality. In consequence, current research focus has shifted to virtual data [37, 20], in particular to the CARLA simulator [20] providing flexible possibilities to test driving models in interaction with complex and configurable environments. Accordingly, the majority of current approaches report their performance on the CARLA benchmarks CoRL2017 [20], NoCrash [17], and the newly introduced Leaderboard [1], providing possibilities to upload an agent policy for evaluation on unknown test sequences. While many works present new datasets or benchmarks, we provide recommendations for an efficient validation design such that the measured performance generalizes well to an unknown driving test. As the driving performance usually varies strongly during different training epochs, a good validation design also enables the selection of a well-performing model checkpoint which easily improves the model's final driving quality.

3. End-to-End Deep Driving

In the following, we introduce our investigated problem setting of end-to-end deep driving as well as the TransFuser method [36], used to approach this problem.

3.1. Problem Definition

Task Description: We investigate the task of point-to-point navigation in an urban environment. The goal is to drive from a starting point $\boldsymbol{u}_1 \in \mathbb{R}^2$ along a pre-defined route $\boldsymbol{u}_1^G = (\boldsymbol{u}_1, ..., \boldsymbol{u}_g, ..., \boldsymbol{u}_G)$, defined by G 2D waypoints $\boldsymbol{u}_g \in \mathbb{R}^2$ towards an end point $\boldsymbol{u}_G$, while following traffic rules and avoiding hazardous incidents in the interaction with other traffic participants. These sparse locations are given by the route definition as global GPS coordinates, as is standard for the CARLA Leaderboard. We therefore employ this approach also in our used CARLA v0.9.13.

Input and Output Representation: At each discrete time instant t, the model has access to several input signals provided by sensor measurements $\mathcal{X}_t$. Firstly, an RGB front camera image $\boldsymbol{x}_t \in \mathbb{I}^{H \times W \times C}$ with height $H = 256$, width $W = 256$, number of channels $C = 3$, and $\mathbb{I} = [0, 1]$ is available. Note that the camera images are extracted at a resolution of 300×400 pixels, which we crop to the region of interest at resolution of 256×256, thereby also removing artifacts at the edges. Secondly, a LiDAR point cloud converted to a histogram pseudo-image $\boldsymbol{v}_t \in \mathbb{I}^{H \times W \times C'}$ with

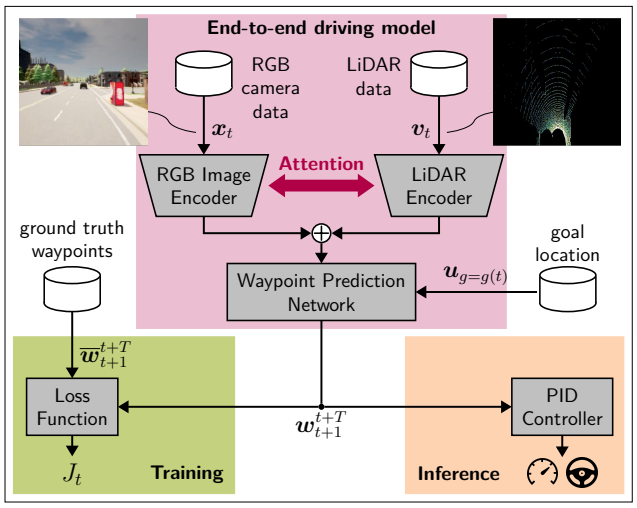


Figure 2: **End-to-end driving method**: The TransFuser model predicts steering, throttle, and brake signals (bottom right) from a camera image and a LiDAR bird's eye view image (top left and top right). Both inputs are encoded, the extracted features are fused, and finally waypoints are predicted from a GRU-based network, making use of the goal location. During training, a loss is applied minimizing the difference between predicted and ground truth waypoints (bottom left). During inference, the waypoints are converted to control signals via a PID controller (bottom right).

$C' = 2$ channels in bird's eye view (BEV) is available. To generate the histogram pseudo-image, the point cloud is divided along the ground plane such that the two channels represent the histogram over the number of points at each image location on/below and over the ground plane, respectively. The underlying LiDAR point cloud is considered in a region of $32\,\mathrm{m}$ in front of the vehicle and $16\,\mathrm{m}$ to each side. From the BEV perspective, the $32\,\mathrm{m} \times 32\,\mathrm{m}$ region is divided into 256×256 blocks of equal size. Finally, the next goal waypoint $\boldsymbol{u}_{g=g(t)}$ can be used as additional input, where $g(t)$ yields the index g which is the desired next goal waypoint at time instance t. Accordingly, the model input is defined by $\mathcal{X}_t = \left\{\boldsymbol{x}_t, \boldsymbol{v}_t, \boldsymbol{u}_{g=g(t)}\right\}$.

As output, the series of the next T waypoints $\boldsymbol{w}_{t+1}^{t+T} = (\boldsymbol{w}_{t+1}, ..., \boldsymbol{w}_{t+\tau}, ..., \boldsymbol{w}_{t+T})$ with $\boldsymbol{w}_{t+\tau} \in \mathbb{R}^2$ of the car's future trajectory in BEV space shall be predicted. The current position and orientation serve as reference coordinate frame to the waypoint's coordinates. The predicted waypoints are converted to steering, throttle, and brake signals via a PID controller [11], expecting $T = 4$ waypoints by default. Note that the model output, i.e., the series of waypoints $\boldsymbol{w}_{t+1}^{t+T}$, is in close proximity to the ego-vehicle, while the inputted goal waypoints $\boldsymbol{u}_1^G$ of the desired route are usually quite sparse and often further away from each other.

3.2. Method Description

End-to-end Driving Model: We choose the TransFuser architecture [36] depicted in Fig. 2 for

our experiments as it is one of the current state-of-the-art models for end-to-end deep driving. The architecture processes both camera image $\boldsymbol{x}_t$ and LiDAR BEV pseudo-image $\boldsymbol{v}_t$ by modality-specific `ResNet` encoders [23]. At each intermediate feature resolution global context information is exchanged between both encoders via attention modules [45]. Thereby, both images are compressed into 512-dimensional feature vectors, which are added element-wise and passed to the waypoint prediction network, cf. Fig. 2. This network first reduces the feature dimensionality from 512 to 64 by fully connected layers. Afterwards the network uses a GRU-based layer [14] taking the goal location $\boldsymbol{u}_{g=g(t)}$ as additional input and a subsequent linear layer to predict the differences $\Delta\boldsymbol{w}_\tau$ between two future waypoints such that $\boldsymbol{w}_{t+\tau} = \boldsymbol{w}_{t+\tau-1} + \Delta\boldsymbol{w}_\tau$. Note that the GRU-based layer uses its recurrent nature to predict each difference $\Delta\boldsymbol{w}_\tau$ by a separate forward pass from $\boldsymbol{w}_{t+\tau-1}$ (using $\boldsymbol{w}_t = (0, 0)$ for the first forward pass). The hidden state used in the first GRU layer forward pass is initialized by the previously extracted 64-dimensional feature vector. Subsequent GRU layer forward passes take the previous one's hidden state (=output) as initial hidden state. For additional details we refer to [36].

Training by Conditional Imitation Learning: Following many recent works [13, 15, 36], we employ conditional imitation learning (CIL), where we aim at obtaining a driving policy $\boldsymbol{\pi}$ that is trained in a supervised fashion to imitate the driving behavior of an expert policy $\overline{\boldsymbol{\pi}}$. The driving policy $\boldsymbol{\pi}$ takes the sensor measurements $\mathcal{X}_t$ as input and outputs the future waypoint trajectory $\boldsymbol{w}_{t+1}^{t+T}$ such that

$$\boldsymbol{w}_{t+1}^{t+T} = \boldsymbol{\pi}\left(\mathcal{X}_t\right). \tag{1}$$

For a certain time instance t, also the expert driving policy can be rolled out in the environment using the same initial conditions as for the trainable driving agent. Then, the expert's driving decisions can be obtained which we represent by a series of ground truth waypoints $\overline{\boldsymbol{w}}_{t+1}^{t+T} = (\overline{\boldsymbol{w}}_{t+1}, ..., \overline{\boldsymbol{w}}_{t+\tau}, ..., \overline{\boldsymbol{w}}_{t+T})$, $\overline{\boldsymbol{w}}_{t+\tau} \in \mathbb{R}^2$ in BEV space. To optimize the driving model, we minimize the distance between the driving model's output $\boldsymbol{w}_{t+1}^{t+T} = \boldsymbol{\pi}\left(\mathcal{X}_t\right)$ and the expert policy's output $\overline{\boldsymbol{w}}_{t+1}^{t+T}$ using the mean absolute error

$$J_t = J\left(\boldsymbol{\pi}\left(\mathcal{X}_t\right), \overline{\boldsymbol{w}}_{t+1}^{t+T}\right) = \sum_{\tau=1}^{T} ||\boldsymbol{w}_{t+\tau} - \overline{\boldsymbol{w}}_{t+\tau}||_1 \tag{2}$$

as shown in the bottom left of Fig. 2. If we now reinterpret t as a sample index such that we consider a whole dataset $\mathcal{D} = \cup_{r=1}^{R} \mathcal{D}_r$ consisting of R routes $\mathcal{D}_r = \left\{(\mathcal{X}_t, \overline{\boldsymbol{w}}_{t+1}^{t+T}), t \in \{1, ..., N_r\}\right\}$ of (possibly varying) length N_r with sensor measurements and corresponding expert driving decisions, we can optimize the model as

$$\boldsymbol{\pi}^* = \underset{\boldsymbol{\pi}}{\operatorname{argmin}}\, \mathbb{E}_{(\mathcal{X}_t, \overline{\boldsymbol{w}}_{t+1}^{t+T}) \sim \mathcal{D}} \left[J\left(\boldsymbol{\pi}\left(\mathcal{X}_t\right), \overline{\boldsymbol{w}}_{t+1}^{t+T}\right)\right], \tag{3}$$

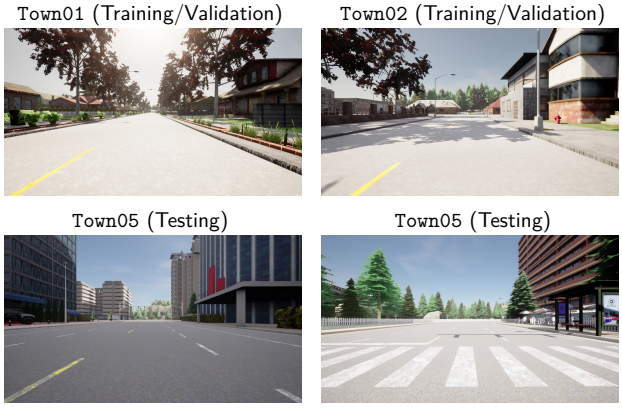


Figure 3: **CARLA simulation environment**: We show some exemplary images, collected in towns, used for training and validation (top) and from the town used for testing (bottom).

to obtain the optimal driving policy $\boldsymbol{\pi}^*$. In practice, we implement the driving model using `PyTorch` [34] and train it for 50 epochs using the AdamW optimizer [30] with a learning rate of 10^{-4} and weight decay of 0.01.

Inference using a PID Controller: While the model is trained to predict waypoints in BEV space, the final driving actions are determined by an inverse dynamics model [11] implemented as PID controller, cf. bottom right in Fig. 2. Specifically, there are two separate PID controllers for both lateral and longitudinal control, both taking the driving model's predicted future waypoints $\boldsymbol{w}_{t+1}^{t+T}$ as input. Accordingly, the longitudinal controller sets throttle and brake, while the lateral controller sets the steering. Specific implementation details can be found in [36].

4. Training and Evaluation Setup

We conduct our experiments using the latest `CARLA v0.9.13`. In the following, we describe our data generation process as well as our validation and test design.

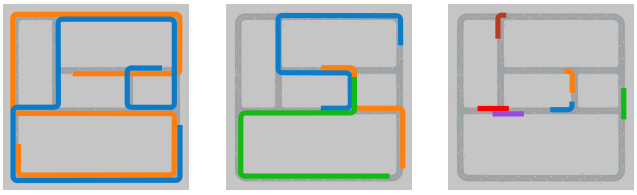
4.1. Training Dataset Generation

Data Generation Concept: For training data, we follow the protocol of [36] and roll out an expert policy in `CARLA`, recording the observations $\mathcal{X}_t$ and corresponding expert actions $\overline{\boldsymbol{w}}_{t+1}^{t+T}$ at a frame rate of 2 fps. RGB images are captured with a forward-facing camera at a resolution of 400×300 pixels and field of view (FOV) of $100°$. LiDAR point clouds are captured with a ray-cast-based `Velodyne 64` LiDAR at a rotation frequency of 10 fps at $10°$ upper FOV and $-30°$ lower FOV. Further, the handcrafted expert policy used to generate $\overline{\boldsymbol{w}}_{t+1}^{t+T}$ has access to privileged simulator information to avoid collisions and other infractions.

Route Design Considerations: The expert follows a set of predefined routes (uniquely defined by sparse waypoints $\boldsymbol{u}_1^G$), during which the expert encounters several

Table 1: **Training set design**: The datasets we use mainly differ in the number of collected images and the number of used routes. We also report the corresponding portions of the four most frequent driving maneuvers (last four columns) given in (%). All training data $\mathcal{D}^{\text{train}}$ has been collected in `CARLA Town[01-04, 06-07, 10]`, while `CARLA Town05` is kept for testing.

training set	# images	# routes	follow lane	go straight	turn left	turn right
$\mathcal{D}^{\text{train}}_{100\text{K}}$	99,806	1762	69.8	11.3	6.9	10.3
$\mathcal{D}^{\text{train}}_{160\text{K}}$	166,852	1903	71.5	10.2	6.9	9.5
$\mathcal{D}^{\text{train}}_{220\text{K}}$	228,023	2901	69.5	11.4	8.2	9.1



(a) Long routes (L) (b) Short routes (S) (c) Tiny routes (T)

Figure 4: **Validation and test route types**: We show examples of the different route types which the driving agent should drive along during validation and testing. Long routes (L) usually progress over many intersections and turns, while short routes (S) only cover a few of these urban traffic sections. A tiny route (T), on the other hand, only covers a single such section by design.

complex urban traffic scenarios. We collect training data in seven different `CARLA` towns (cf. Tab. 1) ranging from rural areas, residential districts to urban areas under simple `ClearNoon` weather conditions (cf. Fig. 3), as we do not focus on investigations regarding weather domain shift. Along a route, the expert is exposed to various predefined randomly picked traffic scenarios, even some scenarios where other traffic participants do not adhere to traffic rules. The expert navigates along two different route types (cf. Fig. 4) in each town: *Tiny routes* (T) involve a single traffic intersection or turn and are usually shorter than 100 meters. *Short routes* (S) cover two or more intersections, being typically 300 to 500 meters long. The training dataset does not include *long routes* (L), which involve complex routing with a total length of more than 1000 meters, as these are characterized by a large imbalance of driving maneuvers towards "follow lane". Accordingly, for each town a set of tiny and short routes is generated. As we investigate different amounts of training data, we ensured that the distribution of driving maneuvers is approximately the same for all collected datasets, cf. Tab. 1. Still, some more variety regarding traffic scenarios is to be expected in larger datasets, which can hardly be quantified.

4.2. Validation and Test Design

Validation and Test Design: For validation and testing, the driving quality of the trained model is measured when driving along pre-defined routes. Note that such closed-loop evaluation is only possible in simulation and differs significantly from many fields, where only the model predictions on a single-image basis are evaluated in open-loop fashion. We put emphasis on comparing the effect of using long, short, or tiny routes depicted in Figs. 4a-4c, respectively, for validation. We aim at a validation whose performance generalizes well to the test performance. For testing, we employ 10 long routes from `CARLA Town05` as in [36]. Note that `CARLA Town05` has not been seen during training and validation.

Table 2: **Validation and test design**: The validation and test setups mainly differ in the number and length of their routes. We also report the corresponding portions of the four most frequent driving maneuvers (last four columns) given in (%). Moreover, test routes $\mathcal{R}^{\text{test}}$ are located in `CARLA Town05`, while validation routes $\mathcal{R}^{\text{val}}$ are located in `CARLA Town[01-04, 06-07]`.

evaluation routes	# routes	route type	follow lane	go straight	turn left	turn right
$\mathcal{R}^{\text{val}}_{160\text{T}}$	160	Tiny	45.3	19.2	17.7	16.7
$\mathcal{R}^{\text{val}}_{80\text{T}}$	80	Tiny	45.1	15.2	20.9	17.5
$\mathcal{R}^{\text{val}}_{22\text{S},1}$	22	Short	75.4	7.7	9.4	6.9
$\mathcal{R}^{\text{val}}_{22\text{S},2}$	22	Short	72.4	10.7	9.1	7.3
$\mathcal{R}^{\text{val}}_{11\text{S}}$	11	Short	80.1	4.5	8.7	6.2
$\mathcal{R}^{\text{val}}_{12\text{L}}$	12	Long	78.3	10.3	4.2	6.9
$\mathcal{R}^{\text{val}}_{6\text{L}}$	6	Long	77.5	10.3	3.5	8.4
$\mathcal{R}^{\text{test}}$	10	Long	77.9	11.2	4.9	5.1

Validation Route Generation: For validation, we generate new routes as reported in Tab. 2 based on a method proposed by [36]. First, intersections are located on the map of a `CARLA` town based on the position of traffic lights. Then for each route a start waypoint $\boldsymbol{u}_1$ and an end waypoint $\boldsymbol{u}_G$ is sampled from the vertices of a square of size $100\,\text{m} \times 100\,\text{m}$ centered around an identified intersection. Based on these two waypoints, a trajectory is generated by a global route planning algorithm as in [1] forming a sparse sequence of route waypoints $\boldsymbol{u}_1^G$. This technique typically produces short or long routes with two or more intersections. Each passing of an intersection or turn in these routes can be converted into a tiny route by selecting the start point before the respective location and the end point afterwards. A more detailed description of the route generation process is given in [36]. After route generation, we remove duplicates in the generated routes as the used algorithm does not ensure a unique route design.

5. Experiments

In the following, we explain our evaluation methodology for analyzing data design choices. Afterwards, we investigate validation route and training data design.

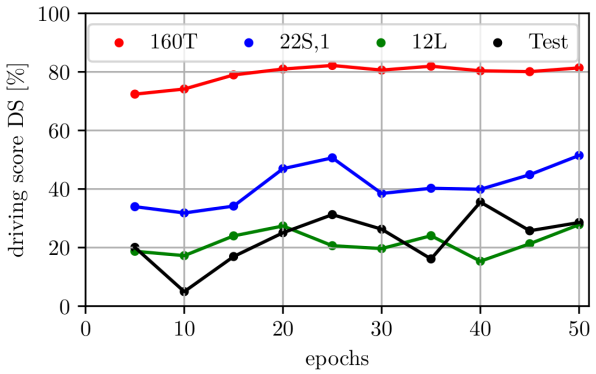


Figure 5: **Driving scores** DS measured at different epochs on **different validation sets** $\mathcal{R}^{\text{val}}_{(\cdot)}$, e.g., 12L represents $\mathcal{R}^{\text{val}}_{\text{12L}}$, and on the test set $\mathcal{R}^{\text{test}}$ (identified by "Test"). Training is performed on $\mathcal{D}^{\text{train}}_{160K}$. We observe high variance in performance over the course of training, making it difficult to identify the best driving model.

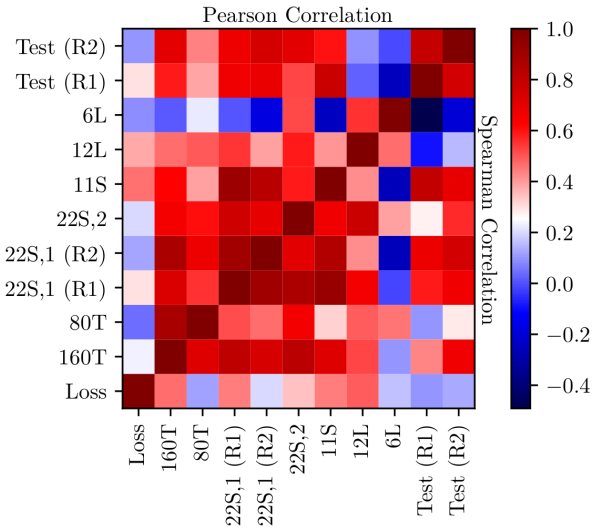


Figure 6: Pearson correlation (upper left part) and Spearman correlation (lower right part) between the driving scores measured on various validation routes $\mathcal{R}^{\text{val}}_{(\cdot)}$, e.g., 12L represents $\mathcal{R}^{\text{val}}_{\text{12L}}$, and the test routes $\mathcal{R}^{\text{test}}$ (identified by "Test"). R1 and R2 represent two different random seeds. We also report correlations between the validation loss "Loss" obtained as in [36] and the driving scores.

5.1. Evaluation Methodology

Driving Performance: We follow recent works [10, 13, 36, 44] in using the driving score DS $\in [0, 1]$ as the main metric to measure the driving performance of a model. As outlined in Sec. 4.2, the validation of driving models is conducted on a set of pre-defined routes. The driving score considers two aspects regarding driving quality along these routes: First, the route completion percentage RC $\in [0, 1]$ is calculated. Possible error cases lowering RC are, deviations from the pre-defined route (i.e., route deviation), an agent not taking any decisions (i.e., blocked agent), a route not finished in time (i.e., route timeout), or off-road driving. Second, an infraction score IS $\in [0, 1]$ considering various traffic incidents is computed as defined in [1]. Considered infractions lowering IS are collisions with pedestrians, other vehicles, and static elements, as well as running red lights or stop signs. For our investigations regarding a suitable validation design, we report the driving score as it reflects the overall driving quality, which we aim at optimizing. For an in-depth analysis regarding limitations of currently used training data for driving models, we additionally report statistics on all considered error cases.

Correlation and Generalization: During the course of training, the driving score obtained on the test set may vary significantly (black line in Fig. 5). As a consequence, we would desire our validation to reflect these performance changes well. As the optimal test driving score is usually reached before the final training epoch 50, we train our driving models for 50 epochs, validate and test it every 5 epochs, and compute the Pearson and/or Spearman correlation between the obtained driving scores. Moreover, we compute the driving score on the test set using the optimal model selected during validation to investigate the generalizability of our validation to the test set. Note that we use this methodology being aware of the test set performance only to find out, which validation design has a good predictive power for the performance obtained during testing.

5.2. Validation Route Design

Varying Driving Performance During Training: Starting point for our data design investigations was the experiment shown in Fig. 5. The driving score DS obtained by the driving model on the test routes $\mathcal{R}^{\text{test}}$ (black line) varies strongly between $5 \ldots 35$ for different training epochs. Even towards the end of training there is no stable convergence, which was a typical behavior in all of our experiments. *Our conclusion is that a good checkpoint selection is essential*, as the checkpoint after 50 epochs often turned out to perform poorly (cf. first row in Tab. 4).

Validation Performance Correlation: After this initial observation, our goal was to find a good generalizing validation, as the test set performance is usually unknown. To get an initial overview, we investigated the correlation between the performance measured on several differently designed validation routes (inducing similar computation complexity) and test routes for a single training run in Fig. 6. Several interesting insights are observable in this figure: First, the offline validation loss (computed as in [36]) correlates badly with driving performance DS on all validation and test routes, which is expected, considering the results for offline metrics from Codevilla et al. [15]. Second, the driving performance DS on test routes ($\mathcal{R}^{\text{test}}$) consisting of 10

Table 3: **Pearson correlation** between the **validation set performance** and the **test set performance** (given by the driving score). We show results for different models trained on $\mathcal{D}^{\text{train}}_{(\cdot)}$ (cf. Tab. 1) and for different validation sets, i.e., all models were validated on $\mathcal{R}^{\text{val}}_{(\cdot)}$ (cf. Tab. 2), where R1 and R2 represent two different random seeds. Best results in boldface, second-best underlined.

		time per checkpoint	trained on $\mathcal{D}^{\text{train}}_{100\text{K}}$	$\mathcal{D}^{\text{train}}_{160\text{K}}$ (R1)	$\mathcal{D}^{\text{train}}_{160\text{K}}$ (R2)	$\mathcal{D}^{\text{train}}_{220\text{K}}$
validated on	$\mathcal{R}^{\text{val}}_{160\text{T}}$	~ 5 h	<u>0.26</u>	0.68	0.30	-0.02
	$\mathcal{R}^{\text{val}}_{80\text{T}}$	~ 2.5 h	0.13	0.43	0.40	-0.01
	$\mathcal{R}^{\text{val}}_{22\text{S},1}$ (R1)	~ 5 h	0.22	0.71	0.73	**0.65**
	$\mathcal{R}^{\text{val}}_{22\text{S},1}$ (R2)	~ 5 h	0.01	**0.76**	0.73	<u>0.53</u>
	$\mathcal{R}^{\text{val}}_{22\text{S},2}$	~ 5 h	0.15	0.65	**0.82**	0.47
	$\mathcal{R}^{\text{val}}_{11\text{S}}$	~ 2.5 h	0.03	<u>0.73</u>	0.76	0.51
	$\mathcal{R}^{\text{val}}_{12\text{L}}$	~ 5 h	**0.79**	0.06	<u>0.79</u>	0.39
	$\mathcal{R}^{\text{val}}_{6\text{L}}$	~ 2.5 h	0.22	-0.14	0.70	0.33

long routes correlates rather poorly to the validation performance DS on 12 or 6 other long routes ($\mathcal{R}^{\text{val}}_{12\text{L}}$ and $\mathcal{R}^{\text{val}}_{6\text{L}}$). As a reason we observed that for long routes, single events such as a blocked agent have a comparably large influence on the overall driving score as only few routes are considered. Moreover, such events happen with different frequency on different routes due to different difficulty level. Even on the same route due to the non-determinism of the validation and test simulations in CARLA, long routes are particularly volatile in their performance variations across different validations as can be seen from the Test or 12L curve in Fig. 5. Short routes ($\mathcal{R}^{\text{val}}_{22\text{S},1}$) and in particular tiny routes ($\mathcal{R}^{\text{val}}_{160\text{T}}$), on the other hand, are usually (a bit) less volatile due to the averaging over more routes. Third, in this initial experiment we observe a high correlation between the validation performance on a medium sized set of short routes ($\mathcal{R}^{\text{val}}_{22\text{S},1}$, $\mathcal{R}^{\text{val}}_{22\text{S},2}$, and $\mathcal{R}^{\text{val}}_{11\text{S}}$) and the test routes or the set containing 12 long routes ($\mathcal{R}^{\text{val}}_{12\text{L}}$). Similar observations can be made for tiny routes (160T and 80T), although they correlate a bit worse when looking at the Spearman correlation. The performance on the set of 6 long routes ($\mathcal{R}^{\text{val}}_{6\text{L}}$) has no high correlation with any other validation performance, again underlining the high performance variability of long routes.

Which validation routes should be used? To get more conclusive evidence, which validation route design provides the best predictive power towards test set performance DS, we compare the correlation of the driving model's performance DS on different validation routes for models trained on different training data in Tab. 3. We choose three sets of routes, i.e., $\mathcal{R}^{\text{val}}_{160\text{T}}$, $\mathcal{R}^{\text{val}}_{22\text{S},1}$, and $\mathcal{R}^{\text{val}}_{12\text{L}}$, inducing similar computational complexity of 5 hours validation time per checkpoint. We observe that tiny routes $\mathcal{R}^{\text{val}}_{160\text{T}}$ lead to a rather poor correlation to the test performance (cf. Tab. 3, first row). Comparing with Fig. 5, we suspect that tiny routes are less informative for measuring the driving quality as only a single traffic section needs to be solved per route, which is often quite easy such that the driving score is consistently high in this validation setting. In contrast, driving requires good driving behavior across many sections, which is better captured by short or long routes shown by the higher correlation values when using $\mathcal{R}^{\text{val}}_{22\text{S},1}$ or $\mathcal{R}^{\text{val}}_{12\text{L}}$ for validation. Halving the validation time per epoch by using just half the amount of routes only works reasonably well for short routes, cf. results for $\mathcal{R}^{\text{val}}_{11\text{S}}$ in Tab. 3.

Table 4: **Test driving scores** given in (%) obtained on $\mathcal{R}^{\text{test}}$, **having used different validation sets**. We show results for four different models trained on $\mathcal{D}^{\text{train}}_{(\cdot)}$ (cf. Tab. 1). The model checkpoint used to obtain the driving score has been selected using the validation set $\mathcal{R}^{\text{val}}_{(\cdot)}$ (cf. Tab. 2) of the respective row. R1 and R2 represent two different random seeds. We additionally show results when testing the model after 50 epochs of training ("naive approach"), using the model obtaining the lowest validation loss ("validation loss") and the result of the expert driving policy, which can be interpreted as an upper performance bound ("expert perf."). Best results in boldface, second-best underlined.

		trained on $\mathcal{D}^{\text{train}}_{100\text{K}}$	$\mathcal{D}^{\text{train}}_{160\text{K}}$ (R1)	$\mathcal{D}^{\text{train}}_{160\text{K}}$ (R2)	$\mathcal{D}^{\text{train}}_{220\text{K}}$
	naive approach	8.9	<u>26.3</u>	17.5	15.8
	validation loss	<u>13.8</u>	16.9	19.8	19.6
validated on	$\mathcal{R}^{\text{val}}_{160\text{T}}$	13.4	**32.7**	13.7	26.3
	$\mathcal{R}^{\text{val}}_{80\text{T}}$	13.4	17.7	26.5	15.8
	$\mathcal{R}^{\text{val}}_{22\text{S},1}$ (R1)	12.6	<u>26.3</u>	26.5	<u>26.7</u>
	$\mathcal{R}^{\text{val}}_{22\text{S},1}$ (R2)	12.6	<u>26.3</u>	<u>28.4</u>	20.5
	$\mathcal{R}^{\text{val}}_{22\text{S},2}$	12.6	**32.7**	<u>28.4</u>	26.3
	$\mathcal{R}^{\text{val}}_{11\text{S}}$	12.6	<u>26.3</u>	26.5	26.3
	$\mathcal{R}^{\text{val}}_{12\text{L}}$	**22.3**	<u>26.3</u>	**28.9**	**30.8**
	$\mathcal{R}^{\text{val}}_{6\text{L}}$	12.6	17.7	16.1	15.8
	expert perf.	46.8	46.8	46.8	46.8

Looking at the obtained driving scores in Tab. 4, this fact is confirmed when validating the model every 5 epochs and choosing the best-performing checkpoint for testing. First of all, we observe that the performance of the naive approach in Tab. 4, which simply tests the driving model obtained after 50 epochs of training is often quite low. Using the validation loss for model selection improves slightly, however, independent of the used validation routes, the test performance is almost always better or on par when using the checkpoint selection. Again we observe that tiny routes $\mathcal{R}^{\text{val}}_{160\text{T}}$ tend to lead to poor driving scores due to the rather weak correlation leading to a bad checkpoint selection. Well performing model checkpoints are usually selected by short routes $\mathcal{R}^{\text{val}}_{22\text{S},1}$ and long routes $\mathcal{R}^{\text{val}}_{12\text{L}}$, with results much closer to the expert's performance. However, when halving the number of routes, the set $\mathcal{R}^{\text{val}}_{11\text{S}}$ still yields high test performance, while $\mathcal{R}^{\text{val}}_{6\text{L}}$ produces rather poor results due to the aforementioned high volatility when using few long routes. *As conclusion, we would choose a large number of long routes when having access to a vast amount of computation resources. If one aims at efficient model development, a medium-sized set of short routes (such as, e.g., $\mathcal{R}^{\text{val}}_{11\text{S}}$) provides a good trade-off between validation*

Table 5: **Performance** on **different training sets**: We report driving score DS, route completion RC, and infraction score IS for various training sets containing a different number of samples. Values are given in (%) and higher is better. We also show the test performance of the expert ("expert perf."), used during generation of training data $\mathcal{D}^{\text{train}}$. Results are averaged over three test runs. As we observed quite some differences between different test runs, we also report respective standard deviations. We additionally report metrics regarding route completion failures and infraction types in number of events per kilometer where lower is better.

training set	time per epoch	driving score	route completion	route deviation	agent blocked	route timeout	off-road driving	infraction score	collisions with pedestrians	collisions with other vehicles	collisions with static elements	running a red light	running a stop sign
$\mathcal{D}^{\text{train}}_{100\text{K}}$	~ 0.5 h	16.8±4.6	91.0±5.3	0.0±0.0	2.4±1.9	0.5±0.4	3.3±0.1	18±6	3.6±0.7	9.6±0.3	0.3±0.5	40.2±3.8	5.6±2.6
$\mathcal{D}^{\text{train}}_{160\text{K}}$	~ 1.0 h	28.8±2.6	80.7±3.3	0.0±0.0	6.7±2.1	0.3±0.5	2.1±1.0	37±2	2.3±0.5	6.3±1.6	1.2±1.0	24.6±1.4	3.5±3.0
$\mathcal{D}^{\text{train}}_{220\text{K}}$	~ 2.0 h	32.2±4.8	78.6±4.8	0.0±0.0	11.8±6.9	0.0±0.0	2.8±0.9	44±2	1.0±0.1	5.5±1.8	1.5±2.6	21.6±1.8	4.2±2.3
expert perf.	-	46.8±7.2	83.4±4.2	0.0±0.0	5.6±2.9	0.0±0.0	0.0±0.0	60±8	1.0±0.9	8.2±2.9	0.0±0.0	8.6±3.2	0.0±0.0

time and driving performance as shown in Tab. 4. Moreover, when noting that the driving performance does not improve anymore through further training, the training process can be stopped, thereby reducing training time by well-generalizing validation during training.

Effect of Non-Determinism: As we observed quite some variance in the obtained driving performance, we want to give insights into the effect of different validation design choices. We provide results in Tabs. 3 and 4. First, we simply repeated the validation on $\mathcal{R}^{\text{val}}_{22\text{S},1}$ using a different random seed (R1,R2). The results regarding correlation and driving score are similarly high but differ in quite a bit in some cases. Similar observations are made when running the same training with a different random seed ($\mathcal{D}^{\text{train}}_{160\text{K}}$ (R1) and $\mathcal{D}^{\text{train}}_{160\text{K}}$ (R2)) or when varying the chosen validation routes ($\mathcal{R}^{\text{val}}_{22\text{S},1}$ and $\mathcal{R}^{\text{val}}_{22\text{S},2}$). However, we observed that simulations in CARLA currently cannot be run in completely deterministic fashion such that the same evaluation result can never be reproduced completely. This is actually similar to an experimental setting in reality, where this is also the case. Therefore, *the standard deviation of results gives important insights into the reproducibility of driving models and the meaningfulness of reported results. Accordingly, it should always be reported.* For example, highly overlapping standard deviation intervals might indicate a low probability that a reported improvement is actually significant.

5.3. Training Data Design

Which training data amount should be used? According to our results on a suitable validation, we now use the validation on $\mathcal{R}^{\text{val}}_{22\text{S},1}$ to select a suitable checkpoint from driving model trainings making use of different amounts of training data. After a checkpoint has been selected the reported results in Tab. 5 are obtained by averaging over three test set evaluations. We also report corresponding standard deviations for each result. We observe that increasing the amount of training data samples $\mathcal{X}_t$ from $100,000$ ($\mathcal{D}^{\text{train}}_{100\text{K}}$) to $220,000$ ($\mathcal{D}^{\text{train}}_{220\text{K}}$) significantly increases the driving score probably due to the larger and more diverse data basis, cf. the respective number of routes in Tab. 1. Interestingly, with less training data, the driving model has a very high route completion score but a rather bad infraction score. With increasing data amount the driving agent apparently learns to avoid hazardous traffic incidents (maybe there have been more diverse examples in the training data available), which, however, also makes the route completion more difficult. *Training on approximately* $160,000$ *images already seems to provide a good trade-off between performance and complexity, while best results are obtained using larger but computationally more expensive amounts of data.*

Expert Performance Bound: Finally, we compare our best results to the results of the CARLA expert. We note that the route completion result is already similar to the expert's result for all trained models. Interestingly, even the expert gets blocked ("agent blocked" in Tab. 5) quite often. As the driving model is trained on data generated using the expert, this behavior seems to transfer to some degree. We observe a similar behavior for collision infractions. Not adhering to rules imposed by signs or traffic lights shows a slightly different behavior. The performance on these infractions tends to improve with more data but is still much worse than the expert result. Here, additional training signals might be necessary. Still, overall *we conclude that for a further improvement of the driving model better data not generated by a far-from-perfect expert is essential.*

6. Conclusions

In this work we present recommendations regarding training and validation data for end-to-end deep driving models. Our results show that in the range of currently employed amounts of data, the driving performance still scales with more data, but seems to be strongly limited by the performance of the expert driving policy used to generate the data. Further, we find that a medium-sized set of short validation routes provides an efficient and (mostly) well-suited validation w.r.t. generalization to unseen test data. Finally, we observe that non-determinism still influences all currently reported results on CARLA evaluations, showing the need to report standard deviations in reported improvements, in particular in the domain of end-to-end deep driving. We believe that our investigations will help researchers to choose efficient setups for their training data and validation design, allowing to find better models for end-to-end deep driving and to reach their goals in shorter time.

References

[1] CARLA Autonomous Driving Leaderboard. https://leaderboard.carla.org, 2020. 3, 5, 6

[2] Alexander Amini, Guy Rosman, Sertac Karaman, and Daniela Rus. Variational End-to-End Navigation and Localization. In *Proc. of ICRA*, pages 8958–8964, Montréal, QC, Canada, May 2019. 2

[3] Claudine Badue, Rânik Guidolini, Raphael Vivacqua Carneiro, Pedro Azevedo, Vinicius B. Cardoso, Avelino Forechi, Luan Jesus, Rodrigo Berriel, Thiago M. Paixao, Filipe Mutz, Lucas Veronesea, Thiago Oliveira-Santosa, and Alberto Ferreira DeSouza. Self-Driving Cars: A Survey. *Expert Systems with Applications*, 165:113816, Mar. 2021. 1

[4] Andreas Bär, Marvin Klingner, Serin Varghese, Fabian Hüger, Peter Schlicht, and Tim Fingscheidt. Robust Semantic Segmentation by Redundant Networks With a Layer-Specific Loss Contribution and Majority Vote. In *Proc. of CVPR - Workshops*, pages 1348–1358, Seattle, WA, USA, June 2020. 1

[5] Alex Bewley, Jessica Rigley, Yuxuan Liu, Jeffrey Hawke, Richard Shen, Vinh-Dieu Lam, and Alex Kendall. Learning to Drive from Simulation without Real World Labels. In *Proc. of ICRA*, pages 4818–4824, Montréal, QC, Canada, May 2019. 2, 3

[6] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, and Jake Zhao. End-to-End Learning for Self-Driving Cars. *ArXiv*, (1604.07316):1–9, Apr. 2016. 1, 2

[7] Peide Cai, Sukai Wang, Hengli Wang, and Ming Liu. Carl-Lead: Lidar-based End-to-End Autonomous Driving withContrastive Deep Reinforcement Learning. *ArXiv*, (2109.08473):1–8, Sept. 2021. 2

[8] John Canny. A Computational Approach to Edge Detection. *IEEE Trans. on PAMI*, PAMI-8(6):679–698, Nov. 1986. 1

[9] Florence Carton, David Filliat, Jaonary Rabarisoa, and Quoc Cuong Pham. Using Semantic Information to Improve Generalization of Reinforcement Learning Policies for Autonomous Driving. In *Proc. of WACV*, pages 144–151, Virtual, Jan. 2021. 2

[10] Dian Chen, Vladlen Koltun, and Philipp Krähenbühl. Learning to Drive from a World on Rails. In *Proc. of ICCV*, pages 15590–15599, Virtual, Oct. 2021. 2, 6

[11] Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by Cheating. In *Proc. of CoRL*, pages 66–75, Virtual, Nov. 2020. 2, 3, 4

[12] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. DeepLab: Semantic Image Segmentation With Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Trans. on PAMI*, 40(4):834–848, Apr. 2017. 1

[13] Kashyap Chitta, Aditya Prakash, and Andreas Geiger. NEAT: Neural Attention Fields for End-to-End Autonomous Driving. In *Proc. of ICCV*, pages 15793–15803, Virtual, Oct. 2021. 2, 4, 6

[14] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proc. of EMNLP*, pages 1–15, Doha, Qatar, Oct. 2014. 4

[15] Felipe Codevilla, Antonio M. Lopez, Vladlen Koltun, and Alexey Dosovitskiy. On Offline Evaluation of Vision-based Driving Models. In *Proc. of ECCV*, pages 236–251, Munich, Germany, Aug. 2018. 2, 3, 4, 6

[16] Felipe Codevilla, Matthias Müller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end Driving via Conditional Imitation Learning. In *Proc. of ICRA*, pages 4693–4700, Brisbane, Australia, May 2018. 2

[17] Felipe Codevilla, Eder Santana, Antonio M. López, and Adrien Gaidon. Exploring the Limitations of Behavior Cloning for Autonomous Driving. In *Proc. of ICCV*, pages 9329–9338, Seoul, Korea, Oct. 2019. 2, 3

[18] Luca Cultrera, Lorenzo Seidenari, Federico Becattini, Pietro Pala, and Alberto Del Bimbo. Explaining Autonomous Driving by Learning End-to-End Visual Attention. In *Proc. of CVPR - Workshops*, pages 340–341, Virtual, June 2020. 2

[19] Soumi Das, Harikrishna Patibandla, Suparna Bhattacharya, Kshounis Bera, Niloy Ganguly, and Sourangshu Bhattacharya. TMCOSS: Thresholded Multi-Criteria Online Subset Selection forData-Efficient Autonomous Driving. In *Proc. of ICCV*, pages 6341–6350, Virtual, Oct. 2021. 2

[20] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An Open Urban Driving Simulator. In *Proc. of CoRL*, pages 1–16, Mountain View, CA, USA, Nov. 2017. 2, 3

[21] Angelos Filos, Panagiotis Tigkas, Rowan McAllister, Nicholas Rhinehart, Sergey Levine, and Yarin Gal. Can Autonomous Vehicles Identify, Recover From, and Adapt to Distribution Shifts? In *Proc. of ICML*, pages 3145–3153, Virtual, July 2020. 2

[22] Jeffrey Hawke, Richard Shen, Corina Gurau, Siddharth Sharma, Daniele Reda, Nikolay Nikolov, Przemysław Mazur, Sean Micklethwaite, Nicolas Griffiths, Amar Shah, and Alex Kendall. Urban Driving with Conditional Imitation Learning. In *Proc. of ICRA*, pages 251–257, Virtual, May 2020. 2, 3

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proc. of CVPR*, pages 770–778, Las Vegas, NV, USA, June 2016. 4

[24] Keishi Ishihara, Anssi Kanervisto, Jun Miura, and Ville Hautamaki. Multi-task Learning with Attention for End-to-end Autonomous Driving. In *Proc. of CVPR - Workshops*, pages 2902–2911, Virtual, June 2021. 2

[25] Ajay Jain, Sergio Casas, Renjie Liao, Yuwen Xiong, Song Feng, Sean Segal, and Raquel Urtasun. Discrete Residual Flow for Probabilistic Pedestrian Behavior Prediction. In *Proc. of CoRL*, pages 407–419, Virtual, Oct. 2020. 1

[26] Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. Learning to Drive in a Day. In *Proc. of ICRA*, pages 8248–8254, Montréal, Canada, May 2019. 2

[27] Seung Wook Kim, Jonah Philion, Antonio Torralba, and Sanja Fidler. DriveGAN: Towards a Controllable High-

Quality Neural Simulation. In *Proc. of ICCV*, pages 5820–5829, Virtual, Oct. 2021. 3

[28] Marvin Klingner, Jan-Aike Termöhlen, Jonas Mikolajczyk, and Tim Fingscheidt. Self-Supervised Monocular Depth Estimation: Solving the Dynamic Object Problem by Semantic Guidance. In *Proc. of ECCV*, pages 582–600, Glasgow, UK, Aug. 2020. 1

[29] Marvin Klingner, Jan-Aike Termöhlen, Jacob Ritterbach, and Tim Fingscheidt. Unsupervised BatchNorm Adaptation (UBNA): A Domain Adaptation Method for Semantic Segmentation Without Using Source Domain Representations. In *Proc. of WACV - Workshops*, pages 1–11, Waikoloa, HI, USA, Jan. 2022. 2

[30] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *Proc. of ICLR*, pages 1–18, New Orleans, LA, USA, May 2019. 4

[31] Abdelhamid Mammeri, Guangqian Lu, and Azzedine Boukerche. Design of Lane Keeping Assist System for Autonomous Vehicles. In *Proc. of NTMS*, pages 1–5, Paris, France, July 2015. 1

[32] Matthias Müller, Alexey Dosovitskiy, Bernard Ghanem, and Vladlen Koltun. Driving Policy Transfer via Modularity and Abstraction. In *Proc. of CoRL*, pages 1–15, Zürich, Switzerland, Oct. 2018. 2

[33] Blażej Osiński, Adam Jakubowski, Pawel Ziecina, Piotr Miloś, Christopher Galias, Silviu Homoceanu, and Henryk Michalewski. Simulation-Based Reinforcement Learningfor Real-World Autonomous Driving. In *Proc. of ICRA*, pages 6411–6418, Virtual, May 2020. 2

[34] 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. 4

[35] Aditya Prakash, Aseem Behl, Eshed Ohn-Bar, Kashyap Chitta, and Andreas Geiger. Exploring Data Aggregation in Policy Learning for Vision-based Urban Autonomous Driving. In *Proc. of CVPR*, pages 11763–11773, Virtual, June 2020. 2

[36] Aditya Prakash, Kashyap Chitta, and Andreas Geiger. Multi-Modal Fusion Transformer for End-to-End Autonomous Driving. In *Proc. of CVPR*, pages 7077–7087, Virtual, June 2021. 1, 2, 3, 4, 5, 6

[37] Stephan Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for Data: Ground Truth from Computer Games. In *Proc. of ECCV*, pages 102–118, Amsterdam, Netherlands, Oct. 2016. 3

[38] Daniela Ridel, Eike Rehder, Martin Lauer, Christoph Stiller, and Denis Wolf. A Literature Review on the Prediction of Pedestrian Behavior in Urban Scenarios. In *Proc. of ITSC*, pages 3105–3112, Maui, HI, USA, Nov. 2018. 1

[39] Sascha Rosbach, Vinit James, Simon Großjohann, Silviu Homoceanu, and Stefan Roth. Driving with Style: Inverse Reinforcement Learning in General-PurposePlanning for Automated Driving. In *Proc. of IROS*, pages 2658–2665, Macau, China, Nov. 2019. 2

[40] Abbas Sadat, Sergio Casas, Mengye Ren, Xinyu Wu, Pranaab Dhawan, and Raquel Urtasun. Perceive, Predict, and Plan: Safe Motion Planning Through Interpretable Semantic Representations. In *Proc. of ECCV*, pages 414–430, Virtual, Aug. 2020. 2

[41] Axel Sauer, Nikolay Savinov, and Andreas Geiger. Conditional Affordance Learning for Driving in Urban Environments. In *Proc. of CoRL*, pages 237–252, Zürich, Switzerland, Oct. 2018. 2

[42] Sahand Sharifzadeh, Ioannis Chiotellis, Rudolph Triebel, and Daniel Cremers. Learning to Drive using Inverse Reinforcement Learning and Deep Q-Networks. In *Proc. of NIPS - Workshops*, pages 1–7, Barcelona, Spain, Dec. 2016. 2

[43] 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, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in Perception for Autonomous Driving: Waymo Open Dataset. In *Proc. of CVPR*, Seattle, WA, USA, June 2020. 2, 3

[44] Marin Toromanoff, Emilie Wirbel, and Fabien Moutarde. End-to-End Model-Free Reinforcement Learningfor Urban Driving using Implicit Affordances. In *Proc. of CVPR*, pages 7153–7162, Virtual, June 2020. 2, 6

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All You Need. In *Proc. of NIPS*, pages 1–11, Long Beach, CA, USA, Dec. 2017. 4

[46] Dan Wang, Junjie Wen, Yuyong Wang, Xiangdong Huang, and Feng Pei. End-to-End Self-Driving Using Deep Neural Networks with Multi-auxiliary Tasks. *Automotive Innovation*, 2(2):127–136, May 2019. 2

[47] Bob Wei, Mengye Ren, Wenyuan Zeng, Ming Liang, Bin Yang, and Raquel Urtasun. Perceive, Attend, and Drive: Learning Spatial Attention for Safe Self-Driving. In *Proc. of ICRA*, pages 4875–4881, Virtual, May 2021. 2

[48] Huazhe Xu, Yang Gao, Fisher Yu, and Trevor Darrell. End-to-end Learning of Driving Models from Large-scale Video Datasets. In *Proc. of CVPR*, pages 2174–2182, Honolulu, HI, USA, July 2017. 2, 3

[49] Luona Yang, Xiaodan Liang, Tairui Wang, and Eric Xing. Real-to-Virtual Domain Unification for End-to-EndAutonomous Driving. In *Proc. of ECCV*, pages 530–545, Munich, Germany, Sept. 2018. 2

[50] Zhengyuan Yang, Yixuan Zhang, Jerry Yu, Junjie Cai, and Jiebo Luo. End-to-end Multi-Modal Multi-Task Vehicle Controlfor Self-Driving Cars with Visual Perceptions. In *Proc. of ICPR*, pages 2289–2294, Beijing, China, Aug. 2018. 2

[51] Jiaming Zhang, Kailun Yang, Angela Constantinescu, Kunyu Peng, Karin Müller, and Rainer Stiefelhagen. Trans4Trans: Efficient Transformer for Transparent Object Segmentation To Help Visually Impaired People Navigate in the Real World. In *Proc. of ICCV - Workshops*, pages 1760–1770, Virtual, Oct. 2021. 2