On Offline Evaluation of Vision-based Driving Models

Felipe Codevilla[1], Antonio M. López[1], Vladlen Koltun[2], and Alexey Dosovitskiy[2]

[1] Computer Vision Center, Universitat Autònoma de Barcelona
[2] Intel Labs

Abstract. Autonomous driving models should ideally be evaluated by deploying them on a fleet of physical vehicles in the real world. Unfortunately, this approach is not practical for the vast majority of researchers. An attractive alternative is to evaluate models offline, on a pre-collected validation dataset with ground truth annotation. In this paper, we investigate the relation between various online and offline metrics for evaluation of autonomous driving models. We find that offline prediction error is not necessarily correlated with driving quality, and two models with identical prediction error can differ dramatically in their driving performance. We show that the correlation of offline evaluation with driving quality can be significantly improved by selecting an appropriate validation dataset and suitable offline metrics.

Keywords: Autonomous driving, deep learning

1 Introduction

Camera-based autonomous driving can be viewed as a computer vision problem. It requires analyzing the input video stream and estimating certain high-level quantities, such as the desired future trajectory of the vehicle or the raw control signal to be executed. Standard methodology in computer vision is to evaluate an algorithm by collecting a dataset with ground-truth annotation and evaluating the results produced by the algorithm against this ground truth (Figure 1(a)). However, driving, in contrast with most computer vision tasks, is inherently active. That is, it involves interaction with the world and other agents. The end goal is to drive well: safely, comfortably, and in accordance with traffic rules. An ultimate evaluation would involve deploying a fleet of vehicles in the real world and executing the model on these (Figure 1(b)). The logistical difficulties associated with such an evaluation lead to the question: Is it possible to evaluate a driving model without actually letting it drive, but rather following the offline dataset-centric methodology?

One successful approach to evaluation of driving systems is via decomposition. It stems from the modular approach to driving where separate subsystems deal with subproblems, such as environment perception, mapping, and vehicle control. The perception stack provides high-level understanding of the scene in terms of semantics, 3D layout, and motion. These lead to standard computer vision tasks, such as object detection, semantic segmentation, depth estimation, 3D reconstruction, or optical flow estimation, which can be evaluated offline on benchmark datasets [10,5,19]. This approach has been extremely fruitful, but it only applies to modular driving systems.
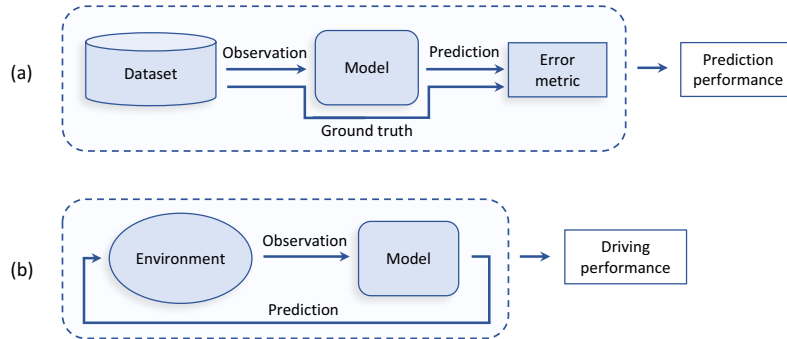
Fig. 1. Two approaches to evaluation of a sensorimotor control model. Top: offline (passive) evaluation on a fixed dataset with ground-truth annotation. Bottom: online (active) evaluation with an environment in the loop.

Recent deep learning approaches [1,27] aim to replace modular pipelines by end-to-end learning from images to control commands. The decomposed evaluation does not apply to models of this type. End-to-end methods are commonly evaluated by collecting a large dataset of expert driving [27] and measuring the average prediction error of the model on the dataset. This offline evaluation is convenient and is consistent with standard practice in computer vision, but how much information does it provide about the actual driving performance of the models?

In this paper, we empirically investigate the relation between (offline) prediction accuracy and (online) driving quality. We train a diverse set of models for urban driving in realistic simulation [6] and correlate their driving performance with various metrics of offline prediction accuracy. By doing so, we aim to find offline evaluation procedures that can be executed on a static dataset, but at the same time correlate well with driving quality. We empirically discover best practices both in terms of selection of a validation dataset and the design of an error metric. Additionally, we investigate the performance of several models on the real-world Berkeley DeepDrive Video (BDDV) urban driving dataset [27].

Our key finding is that offline prediction accuracy and actual driving quality are surprisingly weakly correlated. This correlation is especially low when prediction is evaluated on data collected by a single forward-facing camera on expert driving trajectories – the setup used in most existing works. A network with very low prediction error can be catastrophically bad at actual driving. Conversely, a model with relatively high prediction error may drive well.

We found two general approaches to increasing this poor correlation between prediction and driving. The first is to use more suitable validation data. We found that prediction error measured in lateral cameras (sometimes mounted to collect additional images for imitation learning) better correlates with driving performance than prediction in the forward-facing camera alone. The second approach is to design offline met-

rics that depart from simple mean squared error (MSE). We propose offline metrics that correlate with driving performance more than 60% better than MSE.

2 Related Work

Vision-based autonomous driving tasks have traditionally been evaluated on dedicated annotated real-world datasets. For instance, KITTI [10] is a comprehensive benchmarking suite with annotations for stereo depth estimation, odometry, optical flow estimation, object detection, semantic segmentation, instance segmentation, 3D bounding box prediction, etc. The Cityscapes dataset [5] provides annotations for semantic and instance segmentation. The BDDV dataset [27] includes semantic segmentation annotation. For some tasks, ground truth data acquisition is challenging or nearly impossible in the physical world (for instance, for optical flow estimation). This motivates the use of simulated data for training and evaluating vision models, as in the SYNTHIA [22], Virtual KITTI [9], and GTA5 datasets [20], and the VIPER benchmark [19]. These datasets and benchmarks are valuable for assessing the performance of different components of a vision pipeline, but they do not allow evaluation of a full driving system.

Recently, increased interest in end-to-end learning for driving has led to the emergence of datasets and benchmarks for the task of direct control signal prediction from observations (typically images). To collect such a dataset, a vehicle is equipped with one or several cameras and additional sensors recording the coordinates, velocity, sometimes the control signal being executed, etc. The Udacity dataset [25] contains recordings of lane following in highway and urban scenarios. The CommaAI dataset [23] includes 7 hours of highway driving. The Oxford RobotCar Dataset [16] includes over 1000 km of driving recoded under varying weather, lighting, and traffic conditions. The BDDV dataset [27] is the largest publicly available urban driving dataset to date, with 10,000 hours of driving recorded from forward-facing cameras together with smartphone sensor data such as GPS, IMU, gyroscope, and magnetometer readings. These datasets provide useful training data for end-to-end driving systems. However, due to their static nature (passive pre-recorded data rather than a living environment), they do not support evaluation of the actual driving performance of the learned models.

Online evaluation of driving models is technically challenging. In the physical world, tests are typically restricted to controlled simple environments [13,4] and qualitative results [18,1]. Large-scale real-world evaluations are impractical for the vast majority of researchers. One alternative is simulation. Due of its logistical feasibility, simulation have been commonly employed for driving research, especially in the context of machine learning. The TORCS simulator [26] focuses on racing, and has been applied to evaluating road following [3]. Rich active environments provided by computer games have been used for training and evaluation of driving models [7]; however, the available information and the controllability of the environment are typically limited in commercial games. The recent CARLA driving simulator [6] allows evaluating driving policies in living towns, populated with vehicles and pedestrians, under different weather and illumination conditions. In this work we use CARLA to perform an extensive study of offline performance metrics for driving.

Although the analysis we perform is applicable to any vision-based driving pipeline (including ones that comprise separate perception [21,24,28,2,12] and control modules [17]), in this paper we focus on end-to-end trained models. This line of work dates back to the ALVINN model of Pomerleau [18], capable of road following in simple environments. More recently, LeCun et al. [15] demonstrated collision avoidance with an end-to-end trained deep network. Chen et al. [3] learn road following in the TORCS simulator, by introducing an intermediate representation of "affordances" rather than going directly from pixels to actions. Bojarski et al. [1] train deep convolutional networks for lane following on a large real-world dataset and deploy the system on a physical vehicle. Fernando et al. [8] use neural memory networks combining visual inputs and steering wheel trajectories to perform long-term planning, and use the CommaAI dataset to validate the method. Hubschneider et al. [11] incorporate turning signals as additional inputs to their DriveNet. Codevilla et al. [4] propose conditional imitation learning, which allows imitation learning to scale to complex environments such as urban driving by conditioning action prediction on high-level navigation commands. The growing interest in end-to-end learning for driving motivates our investigation of the associated evaluation metrics.

3   Methodology

We aim to analyze the relation between offline prediction performance and online driving quality. To this end, we train models using conditional imitation learning [4] in a simulated urban environment [6]. We then evaluate the driving quality on goal-directed navigation and correlate the results with multiple offline prediction-based metrics. We now describe the methods used to train and evaluate the models.

3.1   Conditional Imitation Learning

For training the models we use conditional imitation learning – a variant of imitation learning that allows providing high-level commands to a model. When coupled with a high-level topological planner, the method can scale to complex navigation tasks such as driving in an urban environment. We briefly review the approach here and refer the reader to Codevilla et al. [4] for further details.

We start by collecting a training dataset of tuples $\{\langle \mathbf{o}_i, \mathbf{c}_i, \mathbf{a}_i \rangle\}$, each including an observation $\mathbf{o}_i$, a command $\mathbf{c}_i$, and an action $\mathbf{a}_i$. The observation $\mathbf{o}_i$ is an image recorded by a camera mounted on a vehicle. The command $\mathbf{c}_i$ is a high-level navigation instruction, such as "turn left at the next intersection". We use four commands – `continue`, `straight`, `left`, and `right` – encoded as one-hot vectors. Finally, $\mathbf{a}_i$ is a vector representing the action executed by the driver. It can be raw control signal – steering angle, throttle, and brake – or a more abstract action representation, such as a waypoint representing the intended trajectory of the vehicle. We focus on predicting the steering angle in this work.

Given the dataset, we train a convolutional network F with learnable parameters $\boldsymbol{\theta}$ to perform command-conditional action prediction, by minimizing the average predic-

tion loss:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_i \ell(F(\mathbf{o}_i, \mathbf{c}_i, \boldsymbol{\theta}),\, \mathbf{a}_i), \tag{1}$$

where ℓ is a per-sample loss. We experiment with several architectures of the network F, all based on the branched model of Codevilla et al. [4]. Training techniques and network architectures are reviewed in more detail in section 3.2. Further details of training are provided in the supplement.

3.2   Training

Data collection. We collect a training dataset by executing an automated navigation expert in the simulated environment. The expert makes use of privileged information about the environment, including the exact map of the environment and the exact positions of the ego-car, all other vehicles, and pedestrians. The expert keeps a constant speed of 35 km/h when driving straight and reduces the speed when making turns. We record the images from three cameras: a forward-facing one and two lateral cameras facing 30 degrees left and right. In 10% of the data we inject noise in the driving policy to generate examples of recovery from perturbations. In total we record 80 hours of driving data.

Action representation. The most straightforward approach to end-to-end learning for driving is to output the raw control command, such as the steering angle, directly [1,4]. We use this representation in most of our experiments. The action is then a vector $\mathbf{a} \in \mathbb{R}^3$, consisting of the steering angle, the throttle value, and the brake value. To simplify the analysis and preserve compatibility with prior work [1,27], we only predict the steering angle with a deep network. We use throttle and brake values provided by the expert policy described above.

Loss function. In most of our experiments we follow standard practice [1,4] and use mean squared error (MSE) as a per-sample loss:

$$\ell(F(\mathbf{o}_i, \mathbf{c}_i, \boldsymbol{\theta}),\, \mathbf{a}_i) = \|F(\mathbf{o}_i, \mathbf{c}_i, \boldsymbol{\theta}) - \mathbf{a}_i\|^2. \tag{2}$$

We have also experimented with the L1 loss. In most experiments we balance the data during training. We do this by dividing the data into 8 bins based on the ground-truth steering angle and sampling an equal number of datapoints from each bin in every mini-batch. As a result, the loss being optimized is not the average MSE over the dataset, but its weighted version with higher weight given to large steering angles.

Regularization. Even when evaluating in the environment used for collecting the training data, a driving policy needs to generalize to previously unseen views of this environment. Generalization is therefore crucial for a successful driving policy. We use dropout and data augmentation as regularization measures when training the networks.

   Dropout ratio is 0.2 in convolutional layers and 0.5 in fully-connected layers. For each image to be presented to the network, we apply a random subset of a set of transformations with randomly sampled magnitudes. Transformations include contrast change, brightness, and tone, as well as the addition of Gaussian blur, Gaussian noise, salt-and-pepper noise, and region dropout (masking out a random set of rectangles in the image,

each rectangle taking roughly 1% of image area). In order to ensure good convergence, we found it helpful to gradually increase the data augmentation magnitude in proportion to the training step. Further details are provided in the supplement.

Model architecture. We experiment with a feedforward convolutional network, which takes as input the current observation as well as an additional vector of measurements (in our experiments the only measurement is the current speed of the vehicle). This network implements a purely reactive driving policy, since by construction it cannot make use of temporal context. We experiment with three variants of this model. The architecture used by Codevilla et al. [4], with 8 convolutional layers, is denoted as "standard". We also experiment with a deeper architecture with 12 convolutional layers and a shallower architecture with 4 convolutional layers.

3.3   Performance metrics

Offline error metrics. Assume we are given a validation set $\mathcal{V}$ of tuples $\langle \mathbf{o}_i, \mathbf{c}_i, \mathbf{a}_i, v_i \rangle$, indexed by $i \in V$. Each tuple includes an observation, an input command, a ground-truth action vector, and the speed of the vehicle. We assume the validation set consists of one or more temporally ordered driving sequences. (For simplicity in what follows we assume it is a single sequence, but generalization to multiple sequences is trivial.) Denote the action predicted by the model by $\widehat{\mathbf{a}}_i = F(\mathbf{o}_i, \mathbf{c}_i, \boldsymbol{\theta})$. In our experiments, $\mathbf{a}_i$ and $\widehat{\mathbf{a}}_i$ will be scalars, representing the steering angle. Speed is also a scalar (in m/s).

Table 1. Offline metrics used in the evaluation. δ is the Kronecker delta function, θ is the Heaviside step function, Q is a quantization function (see text for details), $|V|$ is the number of samples in the validation dataset.

Metric name	Parameters	Metric definition
Squared error	–	$\frac{1}{|V|} \sum_{i \in V} \|\mathbf{a}_i - \widehat{\mathbf{a}}_i\|^2$
Absolute error	–	$\frac{1}{|V|} \sum_{i \in V} \|\mathbf{a}_i - \widehat{\mathbf{a}}_i\|_1$
Speed-weighted absolute error	–	$\frac{1}{|V|} \sum_{i \in V} \|\mathbf{a}_i - \widehat{\mathbf{a}}_i\|_1 v_i$
Cumulative speed-weighted absolute error	T	$\frac{1}{|V|} \sum_{i \in V} \left\| \sum_{t=0}^{T} (\mathbf{a}_{i+t} - \widehat{\mathbf{a}}_{i+t}) v_{i+t} \right\|_1$
Quantized classification error	σ	$\frac{1}{|V|} \sum_{i \in V} (1 - \delta\left(Q(\mathbf{a}_i, \sigma), Q(\widehat{\mathbf{a}}_i, \sigma)\right))$
Thresholded relative error	α	$\frac{1}{|V|} \sum_{i \in V} \theta\left(\|\widehat{\mathbf{a}}_i - \mathbf{a}_i\| - \alpha \|\mathbf{a}_i\|\right)$

Table 1 lists offline metrics we evaluate in this paper. The first two metrics are standard: mean squared error (which is typically the training loss) and absolute error. Absolute error gives relatively less weight to large mistakes than MSE.

The higher the speed of the car, the larger the impact a control mistake can have. To quantify this intuition, we evaluate speed-weighted absolute error. This metric approximately measures how quickly the vehicle is diverging from the ground-truth trajectory,

that is, the projection of the velocity vector onto the direction orthogonal to the heading direction.

We derive the next metric by accumulating speed-weighted errors over time. The intuition is that the average prediction error may not be characteristic of the driving quality, since it does not take into account the temporal correlations in the errors. Temporally uncorrelated noise may lead to slight oscillations around the expert trajectory, but can still result in successful driving. In contrast, a consistent bias in one direction for a prolonged period of time inevitably leads to a crash. We therefore accumulate the speed-weighted difference between the ground-truth action and the prediction over T time steps. This measure is a rough approximation of the divergence of the vehicle from the desired trajectory over T time steps.

Another intuition is that small noise may be irrelevant for the driving performance, and what matters is getting the general direction right. Similar to Xu et al. [27], we quantize the predicted actions and evaluate the classification error. For quantization, we explicitly make use of the fact that the actions are scalars (although a similar strategy can be applied to higher-dimensional actions). Given a threshold value σ, the quantization function $Q(x, \sigma)$ returns -1 if $x < -\sigma$, 0 if $-\sigma \leq x < \sigma$, and 1 if $x \geq \sigma$. For steering angle, these values correspond to going left, going straight, and going right. Given the quantized predictions and the ground truth, we compute the classification error.

Finally, the last metric is based on quantization and relative errors. Instead of quantizing with a fixed threshold as in the previous metric, here the threshold is adaptive, proportional to the ground truth steering signal. The idea is that for large action values, small discrepancies with the ground truth are not as important as for small action values. Therefore, we count the fraction of samples for which $\|\widehat{\mathbf{a}}_i - \mathbf{a}_i\| \geq \alpha \|\mathbf{a}_i\|$.

Online performance metrics. We measure the driving quality using three metrics. The first one is the success rate, or simply the fraction of successfully completed navigation trials. The second is the average fraction of distance traveled towards the goal per episode (this value can be negative is the agent moves away form the goal). The third metric measures the average number of kilometers traveled between two infractions. (Examples of infractions are collisions, driving on the sidewalk, or driving on the opposite lane.)

4   Experiments

We perform an extensive study of the relation between online and offline performance of driving models. Since conducting such experiments in the real world would be impractical, the bulk of the experiments are performed in the CARLA simulator [6]. We start by training a diverse set of driving models with varying architecture, training data, regularization, and other parameters. We then correlate online driving quality metrics with offline prediction-based metrics, aiming to find offline metrics that are most predictive of online driving performance. Finally, we perform an additional analysis on the real-world BDDV dataset. Supplementary materials can be found on the project page: `https://sites.google.com/view/evaluatedrivingmodels`.

4.1   Experimental setup

Simulation. We use the CARLA simulator to evaluate the performance of driving models in an urban environment. We follow the testing protocol of Codevilla et al. [4] and Dosovitskiy et al. [6]. We evaluate goal-directed navigation with dynamic obstacles. One evaluation includes 25 goal-directed navigation trials.

CARLA provides two towns (Town 1 and Town 2) and configurable weather and lighting conditions. We make use of this capability to evaluate generalization of driving methods. We use Town 1 in 4 weathers (Clear Noon, Heavy Rain Noon, Clear Sunset and Clear After Rain) for training data collection, and we use two test conditions: Town 1 in clear noon weather and Town 2 in Soft Rain Sunset weather. The first condition is present in the training data; yet, note that the specific images observed when evaluating the policies have almost certainly not been seen during training. Therefore even this condition requires generalization. The other condition – Town 2 and soft rain sunset weather – is completely new and requires strong generalization.

For validation we use 2 hours of driving data with action noise and 2 hours of data without action noise, in each of the conditions. With three cameras and a frame rate of 10 frames per second, one hour of data amounts to 108,000 validation images.

Real-world data. For real-world tests we use the validation set of the BDDV dataset [27], containing 1816 dashboard camera videos. We computed the offline metrics over the entire dataset using the pre-trained models and the data filtering procedures provided by Xu et al. [27].

Network training and evaluation. All models were trained using the Adam optimizer [14] with minibatches of 120 samples and an initial learning rate of 10^{-4}. We reduce the learning rate by a factor of 2 every $50K$ iterations. All models were trained up to $500K$ iterations. In order to track the evolution of models during the course of training, for each model we perform both online and offline evaluation after the following numbers of training mini-batches: 2K, 4K, 8K, 16K, 32K, 64K, 100K, 200K, 300K, 400K, and 500K.

4.2   Evaluated models

We train a total of 45 models. The parameters we vary can be broadly separated into three categories: properties of the training data, of the model architecture, and of the training procedure. We vary the amount and the distribution of the training data. The amount varies between 0.2 hours and 80 hours of driving. The distribution is one of the following four: all data collected from three cameras and with noise added to the control, only data from the central camera, only data without noise, and data from the central camera without noise. The model architecture variations amount to varying the depth between 4 and 12 layers. The variations in the training process are the use of data balancing, the loss function, and the regularization applied (dropout and the level of data augmentation). A complete list of parameters varied during the evaluation is provided in the supplement.

4.3   Correlation between offline and online metrics

We start by studying the correlation between online and offline performance metrics on the whole set of evaluated models. We represent the results by scatter plots and correlation coefficients. To generate a scatter plot, we select two metrics and plot each evaluated model as a circle, with the coordinates of the center of the circle equal to the values of these two metrics, and the radius of the circle proportional to the training iteration the model was evaluated at. To quantify the correlations, we use the standard sample Pearson correlation coefficient, computed over all points in the plot. In the figures below, we plot results in generalization conditions (Town 2, unseen weather). We focus our analysis on the well-performing models, by discarding the 50% worst models according to the offline metric. Results in training conditions, as well as scatter plots with all models, are shown in the supplement.

The effect of validation data. We first plot the (offline) average steering MSE versus the (online) success rate in goal-directed navigation, for different offline validation datasets. We vary the number of cameras used for validation (just a forward-facing camera or three cameras including two lateral ones) and the presence of action noise in the validation set. This experiment is inspired by the fact that the 3-camera setup and the addition of noise have been advocated for training end-to-end driving models [27,1,6,4].

The results are shown in Figure 2. The most striking observation is that the correlation between offline prediction and online performance is weak. For the basic setup – central camera and no action noise – the absolute value of the correlation coefficient is only 0.39. The addition of action noise improves the correlation to 0.54. Evaluating on data from three cameras brings the correlation up to 0.77. This shows that a successful policy must not only predict the actions of an expert on the expert's trajectories, but also for observations away from the expert's trajectories. Proper validation data should therefore include examples of recovery from perturbations.
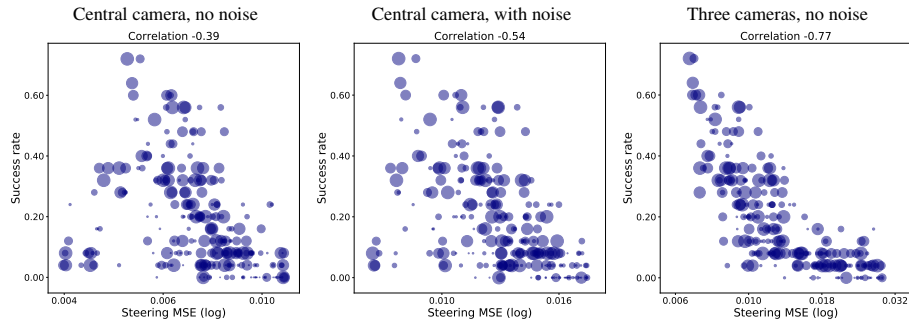

Fig. 2. Scatter plots of goal-directed navigation success rate vs. steering MSE when evaluated on data from different distributions. We evaluate the models in the generalization condition (Town 2) and we plot the 50% best-performing models according to the offline metric. Sizes of the circles denote the training iterations at which the models were evaluated. We additionally show the sample Pearson correlation coefficient for each plot. Note how the error on the basic dataset (single camera, no action noise) is the least informative of the driving performance.

Offline metrics. Offline validation data from three cameras or with action noise may not always be available. Therefore, we now aim to find offline metrics that are predictive of driving quality even when evaluated in the basic setup with a single forward-facing camera and no action noise.

Figure 3 shows scatter plots of offline metrics described in Section 3.3, versus the navigation success rate. MSE is the least correlated with the driving success rate: the absolute value of the correlation coefficient is only 0.39. Absolute steering error is more strongly correlated, at 0.61. Surprisingly, weighting the error by speed or accumulating the error over multiple subsequent steps does not improve the correlation. Finally, quantized classification error and thresholded relative error are also more strongly correlated, with the absolute value of the correlation coefficient equal to 0.65 and 0.64, respectively.
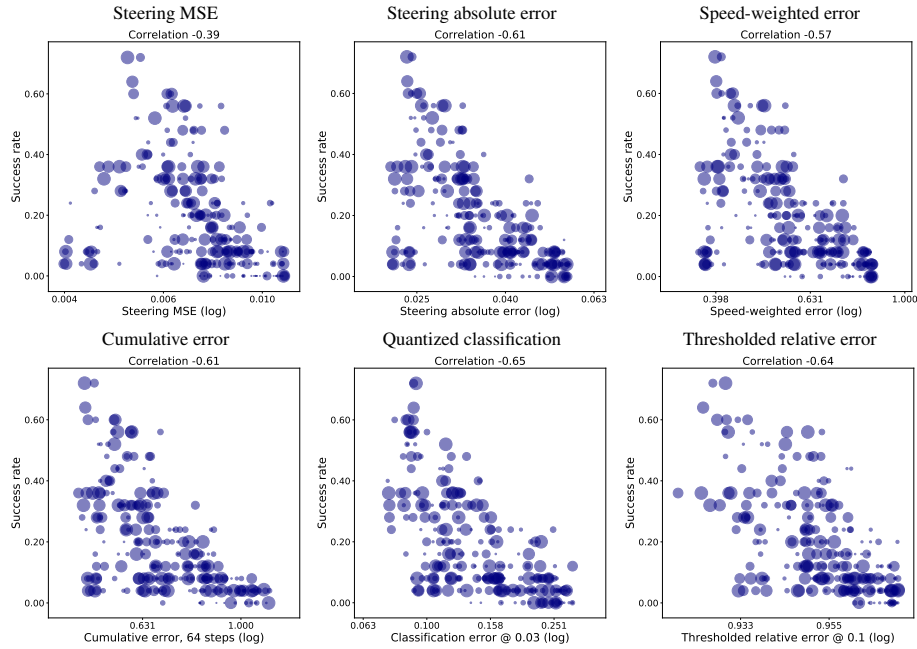

Fig. 3. Scatter plots of goal-directed navigation success rate vs. different offline metrics. We evaluate the models in the generalization condition (Town 2) and we plot the 50% best-performing models according to the offline metric. Note how correlation is generally weak, especially for mean squred error (MSE).

Online metrics. So far we have looked at the relation between offline metrics and a single online metric – success rate. Is success rate fully representative of actual driving quality? Here we compare the success rate with two other online metrics: average fraction of distance traveled towards the goal and average number of kilometers traveled between two infractions.

Figure 4 shows pairwise scatter plots of these three online metrics. Success rate and average completion are strongly correlated, with a correlation coefficient of 0.8. The number of kilometers traveled between two infractions is similarly correlated with the success rate (0.77), but much less correlated with the average completion (0.44). We conclude that online metrics are not perfectly correlated and it is therefore advisable to measure several online metrics when evaluating driving models. Success rate is well correlated with the other two metrics, which justifies its use as the main online metric in our analysis.
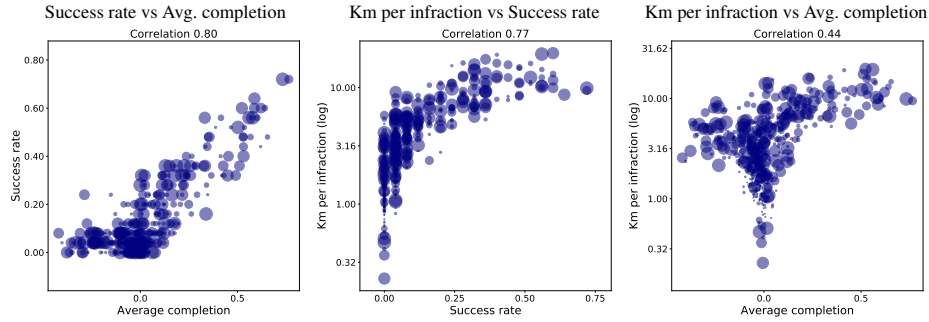

Fig. 4. Scatter plots of online driving quality metrics versus each other. The metrics are: success rate, average fraction of distance to the goal covered (average completion), and average distance (in km) driven between two infractions. Success rate is strongly correlated with the other two metrics, which justifies its use as the main online metric in our analysis.

Case study. We have seen that even the best-correlated offline and online metrics have a correlation coefficient of only 0.65. Aiming to understand the reason for this remaining discrepancy, here we take a closer look at two models which achieve similar prediction accuracy, but drastically different driving quality. The first model was trained with the MSE loss and forward-facing camera only. The second model used the L1 loss and three cameras. We refer to these models as Model 1 and Model 2, respectively.

Figure 5 (top left) shows the ground truth steering signal over time (blue), as well as the predictions of the models (red and green, respectively). There is no obvious qualitative difference in the predictions of the models: both often deviate from the ground truth. One difference is a large error in the steering signal predicted by Model 1 in a turn, as shown in Figure 5 (top right). Such a short-term discrepancy can lead to a crash, and it is difficult to detect based on average prediction error. The advanced offline metrics evaluated above are designed to be better at capturing such mistakes.

Figure 5 (bottom) shows several trajectories driven by both models. Model 1 is able to drive straight for some period of time, but eventually crashes in every single trial, typically because of wrong timing or direction of a turn. In contrast, Model 2 drives well and successfully completes most trials. This example illustrates the difficulty of using offline metrics for predicting online driving behavior.
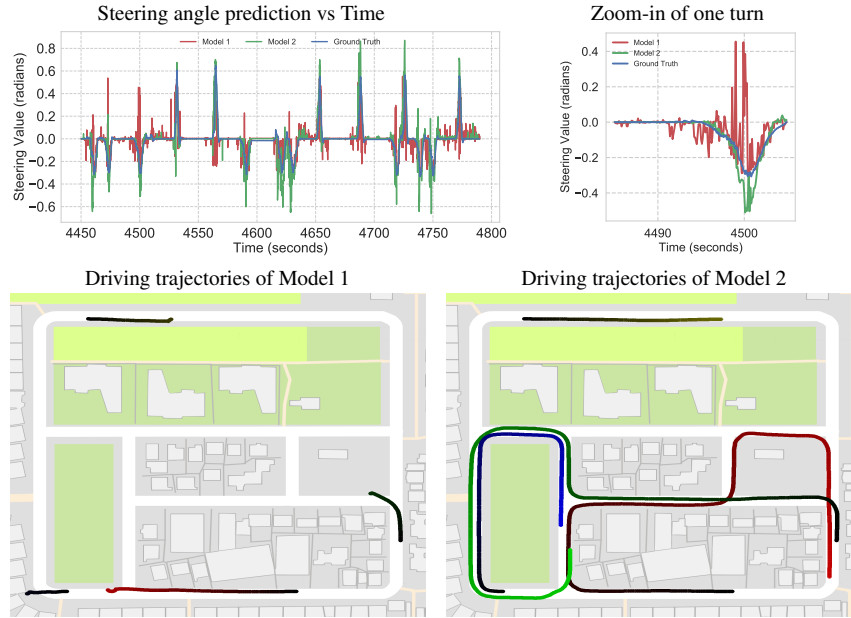
Fig. 5. Detailed evaluation of two driving models with similar offline prediction quality, but very different driving behavior. Top left: Ground-truth steering signal (blue) and predictions of two models (red and green) over time. Top right: a zoomed fragment of the steering time series, showing a large mistake made by Model 1 (red). Bottom: Several trajectories driven by the models in Town 1. Same scenarios indicated with the same color in both plots. Note how the driving performance of the models is dramatically different: Model 1 crashes in every trial, while Model 2 can drive successfully.

4.4   Real-world data

Evaluation of real-world urban driving is logistically complicated, therefore we restrict the experiments on real-world data to an offline evaluation. We use the BDDV dataset and the trained models provided by [27]. The models are trained to perform 4-way classification (accelerate, brake, left, right), and we measure their classification accuracy. We evaluate on the validation set of BDDV.

The offline metrics we presented above are designed for continuous values and cannot be directly applied to classification-based models. Yet, some of them can be adapted to this discrete setting. Table 2 shows the average accuracy, as well as several additional metrics. First, we provide a breakdown of classification accuracy by subsets of the data corresponding to different ground truth labels. The prediction error in the turns is most informative, yielding the largest separation between the best and the worst models. Second, we try weighting the errors with the ground-truth speed. We measure the resulting metric for the full validation dataset, as well as for turns only. These metrics reduce the gap between the feedforward and the LSTM models.

Table 2. Detailed accuracy evaluations on the BDDV dataset. We report the 4-way classification accuracy (in %) for various data subsets and varying speed.

Model	Average accuracy				Weighted with speed	
	All data	Straight	Stop	Turns	All data	Turns
Feedforward	78.0	90.0	72.0	32.4	80.7	27.7
CNN + LSTM	81.8	90.2	78.1	49.3	83.0	43.2
FCN + LSTM	83.3	90.4	80.7	50.7	83.6	44.4

4.5   Detailed evaluation of models

Scatter plots presented in the previous sections indicate general tendencies, but not the performance of specific models. Here we provide a more detailed evaluation of several driving models, with a focus on several parameters: the amount of training data, its distribution, the regularization being used, the network architecture, and the loss function. We evaluate two offline metrics – MSE and the thresholded relative error (TRE) – as well as the goal-directed navigation success rate. For TRE we use the parameter $\alpha = 0.1$.

The results are shown in Table 3. In each section of the table all parameters are fixed, except for the parameter of interest. (Parameters may vary between sections.) Driving performance is sensitive to all the variations. Larger amount of training data generally leads to better driving. Training with one or three cameras has a surprisingly minor effect. Data balancing helps in both towns. Regularization helps generalization to the previously unseen town and weather. Deeper networks generally perform better. Finally, the L1 loss leads to better driving than the usual MSE loss. This last result is in agreement with Figure 3, which shows that absolute error is better correlated with the driving quality than MSE.

Next, for each of the 6 parameters and each of the 2 towns we check if the best model chosen based on the offline metrics is also the best in terms of the driving quality. This simulates a realistic parameter tuning scenario a practitioner might face. We find that TRE is more predictive of the driving performance than MSE, correctly identifying the best-driving model in 10 cases out of 12, compared to 6 out of 12 for MSE. This demonstrates that TRE, although far from being perfectly correlated with the online driving quality, is much more indicative of well-driving models than MSE.

5   Conclusion

We investigated the performance of offline versus online evaluation metrics for autonomous driving. We have shown that the MSE prediction error of expert actions is not a good metric for evaluating the performance of autonomous driving systems, since it is very weakly correlated with actual driving quality. We explore two avenues for improving the offline metrics: modifying the validation data and modifying the metrics themselves. Both paths lead to improved correlation with driving quality.

Our work takes a step towards understanding the evaluation of driving models, but it has several limitations that can be addressed in future work. First, the evaluation is

Table 3. Detailed evaluation of models in CARLA. "TRE" stands for thresholded relative error, "Success rate" for the driving success rate. For MSE and TRE lower is better, for the success rate higher is better. We mark with bold the best result in each section. We highlight in green the cases where the best model according to an offline metric is also the best at driving, separately for each section and each town. Both MSE and TRE are not necessarily correlated with driving performance, but generally TRE is more predictive of driving quality, correctly identifying 10 best-driving models out of 12, compared to 6 out of 12 for MSE.

		MSE		TRE @ 0.1		Success rate	
Parameter	Value	Town 1	Town 2	Town 1	Town 2	Town 1	Town 2
Amount of training data	0.2 hours	0.0086	0.0481	0.970	0.985	0.44	0.00
	1 hour	0.0025	0.0217	0.945	0.972	0.44	0.04
	5 hours	**0.0005**	**0.0093**	0.928	0.961	0.60	**0.08**
	25 hours	0.0007	0.0166	**0.926**	**0.958**	**0.76**	0.04
Type of training data	1 cam., no noise	0.0007	**0.0066**	**0.922**	0.947	**0.84**	0.04
	1 cam., noise	0.0009	0.0077	0.926	**0.946**	0.80	**0.20**
	3 cam., no noise	**0.0004**	0.0086	0.928	0.953	**0.84**	0.08
	3 cam., noise	0.0007	0.0166	0.926	0.958	0.76	0.04
Data balancing	No balancing	0.0012	**0.0065**	0.907	**0.924**	0.88	0.36
	With balancing	**0.0011**	0.0066	**0.891**	0.930	**0.92**	**0.56**
Regularization	None	0.0014	0.0092	**0.911**	0.953	**0.92**	0.08
	Mild dropout	0.0010	0.0074	0.921	0.953	0.84	0.20
	High dropout	**0.0007**	0.0166	0.926	0.958	0.76	0.04
	High drop., data aug.	0.0013	**0.0051**	0.919	**0.931**	0.88	**0.36**
Network architecture	Shallow	**0.0005**	0.0111	0.936	0.963	0.68	0.12
	Standard	0.0007	0.0166	**0.926**	0.958	**0.76**	0.04
	Deep	0.0011	**0.0072**	0.928	**0.949**	**0.76**	**0.24**
Loss function	L2	**0.0010**	0.0074	0.921	0.953	0.84	0.20
	L1	0.0012	**0.0061**	**0.891**	**0.944**	**0.96**	**0.52**

almost entirely based on simulated data. We believe that the general conclusion about weak correlation of online and offline metrics is likely to transfer to the real world; however, it is not clear if the details of our correlation analysis will hold in the real world. Performing a similar study with physical vehicles operating in rich real-world environments would therefore be very valuable. Second, we focus on the correlation coefficient as the measure of relation between two quantities. Correlation coefficient estimates the connection between two variables to some degree, but a finer-grained analysis may be needed provide a more complete understanding of the dependencies between online and offline metrics. Third, even the best offline metric we found is far from being perfectly correlated with actual driving quality. Designing offline performance metrics that are more strongly correlated with driving performance remains an important challenge.

Acknowledgements

References

1. Bojarski, M., Testa, D.D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., Zieba, K.: End to end learning for self-driving cars. arXiv:1604.07316 (2016)
2. Bresson, G., Alsayed, Z., Yu, L., Glaser, S.: Simultaneous localization and mapping: A survey of current trends in autonomous driving. IEEE Trans. on Intelligent Vehicles (2017)
3. Chen, C., Seff, A., Kornhauser, A.L., Xiao, J.: DeepDriving: Learning affordance for direct perception in autonomous driving. In: International Conference on Computer Vision (ICCV) (2015)
4. Codevilla, F., Müller, M., López, A., Koltun, V., Dosovitskiy, A.: End-to-end driving via conditional imitation learning. In: International Conference on Robotics and Automation (ICRA) (2018)
5. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The Cityscapes dataset for semantic urban scene understanding. In: Computer Vision and Pattern Recognition (CVPR) (2016)
6. Dosovitskiy, A., Ros, G., Codevilla, F., López, A., Koltun, V.: CARLA: An open urban driving simulator. In: Conference on Robot Learning (CoRL) (2017)
7. Ebrahimi, S., Rohrbach, A., Darrell, T.: Gradient-free policy architecture search and adaptation. In: Conference on Robot Learning (CoRL) (2017)
8. Fernando, T., Denman, S., Sridharan, S., Fookes, C.: Going deeper: Autonomous steering with neural memory networks. In: International Conference on Computer Vision (ICCV) Workshops (2017)
9. Gaidon, A., Wang, Q., Cabon, Y., Vig, E.: Virtual worlds as proxy for multi-object tracking analysis. In: Computer Vision and Pattern Recognition (CVPR) (2016)
10. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? The KITTI vision benchmark suite. In: Computer Vision and Pattern Recognition (CVPR) (2012)
11. Hubschneider, C., Bauer, A., Weber, M., Zollner, J.M.: Adding navigation to the equation: Turning decisions for end-to-end vehicle control. In: Intelligent Transportation Systems Conference (ITSC) Workshops (2017)
12. Jin, X., Xiao, H., Shen, X., Yang, J., Lin, Z., Chen, Y., Jie, Z., Feng, J., Yan, S.: Predicting scene parsing and motion dynamics in the future. In: Neural Information Processing Systems (NIPS) (2017)
13. Kahn, G., Villaflor, A., Pong, V., Abbeel, P., Levine, S.: Uncertainty-aware reinforcement learning for collision avoidance. arXiv:1702.01182 (2017)
14. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: International Conference on Learning Representation (ICLR) (2015)
15. LeCun, Y., Muller, U., Ben, J., Cosatto, E., Flepp, B.: Off-road obstacle avoidance through end-to-end learning. In: Neural Information Processing Systems (NIPS) (2005)
16. Maddern, W., Pascoe, G., Linegar, C., Newman, P.: 1 Year, 1000km: The Oxford RobotCar Dataset. The International Journal of Robotics Research (IJRR) (2017)
17. Paden, B., Cáp, M., Yong, S.Z., Yershov, D.S., Frazzoli, E.: A survey of motion planning and control techniques for self-driving urban vehicles. IEEE Trans. on Intelligent Vehicles (2016)
18. Pomerleau, D.: ALVINN: An autonomous land vehicle in a neural network. In: Neural Information Processing Systems (NIPS) (1988)
19. Richter, S.R., Hayder, Z., Koltun, V.: Playing for benchmarks. In: International Conference on Computer Vision (ICCV) (2017)
20. Richter, S.R., Vineet, V., Roth, S., Koltun, V.: Playing for data: Ground truth from computer games. In: European Conference on Computer Vision (ECCV) (2016)

21. Ros, G., Ramos, S., Granados, M., Bakhtiary, A., Vázquez, D., López, A.M.: Vision-based offline-online perception paradigm for autonomous driving. In: Winter conf. on Applications of Computer Vision (WACV) (2015)
22. Ros, G., Sellart, L., Materzynska, J., Vázquez, D., López, A.M.: The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In: Computer Vision and Pattern Recognition (CVPR) (2016)
23. Santana, E., Hotz, G.: Learning a driving simulator. arXiv:1608.01230 (2016)
24. Schneider, L., Cordts, M., Rehfeld, T., Pfeiffer, D., Enzweiler, M., Franke, U., Pollefeys, M., Roth, S.: Semantic stixels: Depth is not enough. In: Intelligent Vehicles Symposium (IV) (2016)
25. Udacity: `https://github.com/udacity/self-driving-car`
26. Wymann, B., Espié, E., Guionneau, C., Dimitrakakis, C., Coulom, R., Sumner, A.: TORCS, The Open Racing Car Simulator. `http://www.torcs.org`
27. Xu, H., Gao, Y., Yu, F., Darrell, T.: End-to-end learning of driving models from large-scale video datasets. In: Computer Vision and Pattern Recognition (CVPR) (2017)
28. Zhu, Z., Liang, D., Zhang, S., Huang, X., Li, B., Hu, S.: Traffic-sign detection and classification in the wild. In: Computer Vision and Pattern Recognition (CVPR) (2016)

Supplementary material for "On Offline Evaluation of Vision-based Driving Models"

Felipe Codevilla[1], Antonio M. López[1], Vladlen Koltun[2], and Alexey Dosovitskiy[2]

[1] Computer Vision Center, Universitat Autònoma de Barcelona
[2] Intel Labs

1 Training Details

1.1 Network architecture

Table 1 provides the details of the standard architecture used in the experiments. We also experimented with a deeper architecture with 12 convolutional layers (an additional layer at each feature map resolution) and a shallower network with 4 convolutional layers (one layer at each resolution instead of two).

module	input dimension	channels	kernel	stride
Perception	$200 \times 88 \times 3$	32	5	2
	$98 \times 48 \times 32$	32	3	1
	$96 \times 46 \times 32$	64	3	2
	$47 \times 22 \times 64$	64	3	1
	$45 \times 20 \times 64$	128	3	2
	$22 \times 9 \times 128$	128	3	1
	$20 \times 7 \times 128$	256	3	2
	$9 \times 3 \times 256$	256	3	1
	$7 \cdot 1 \cdot 256$	512	–	–
	512	512	–	–
Measurement	1	128	–	–
	128	128	–	–
	128	128	–	–
Joint input	$512 + 128$	512	–	–
Control	512	256	–	–
	256	256	–	–
	256	1	–	–

Table 1. Architecture of the "standard" architecture used in the experiments.

1.2 Image Input

We record images from the simulator at the resolution of 800×600 pixels. Before feeding the image to the network, we cropped 171 pixels at the top and 45 pixels at

the bottom, resized the resulting 800×384 pixels image to 200×88 pixels resolution, subtracted the mean, and normalized.

1.3   Augmentation

We used the following transformations for data augmentation: Gaussian blur, additive Gaussian noise, pixel dropout, additive and multiplicative brightness variation, contrast variation, saturation variation and also region pixel dropout.

1.4   Noise Distribution

During training data collection, 10% of the time we inject noise into the expert's steering. Namely, at a random point in time we add a perturbation to the steering angle provided by the driver. The perturbation is a triangular impulse, please see Codevilla et al. [**?**] for details.

2   Evaluated Model Details

Table 2 lists the model parameters we varied in our experiments.

Training data	
Amount	The amount of training data (in hours of driving footage)
Distribution	Distribution of the data used for training
3 cam. + noise	Data from three cameras, with action noise in 10% of data
3 cam.	Data from three cameras, without action noise
1 cam. + noise	Data only from the central camera, with action noise
1 cam.	Data only from the central camera, without action noise
Model architecture	
Architecture	The architecture of the network
Shallow	4-layer convolutional network
Standard	8-layer convolutional network
Deep	12-layer convolutional network
Training procedure	
Regularization	Regularization techniques applied during training
None	No regularization
Mild Dropout	Dropout: 50% in the FC layers of the measurements and control modules
High Dropout	Dropout: 50% in all FC layers
Drop. + aug.	Dropout and random transformations applied to input images
Balancing	If data balancing w.r.t. steering angles is applied during training
Loss	Type of loss function used for training
MSE (L2)	Regression with MSE loss
L1	Regression with absolute error (L1)

Table 2. Parameters of driving models explored in the evaluation.

3   Additional results

In the main paper we evaluate the models in the generalization condition (Town 2) and we plot 50% best-performing models according to the offline metric. Here we show results in the training condition (Town 1) and show plots with all models, not only best-performing ones.

Figures 1 and 2 show scatter plots of online vs offline metrics with 50% best models, evaluated in Town 1. Figure 3 shows scatter plots of online driving quality metrics, evaluated in Town 1. Figures 4 and 5 show scatter plots of online vs offline metrics with all models, evaluated in Town 1. Figures 6 and 7 show scatter plots of online vs offline metrics with all models, evaluated in Town 2.

Fig. 1. Scatter plots of goal-directed navigation success rate vs steering absolute error when evaluated on data from different distributions. Town 1 (training conditions), best 50% of the models.
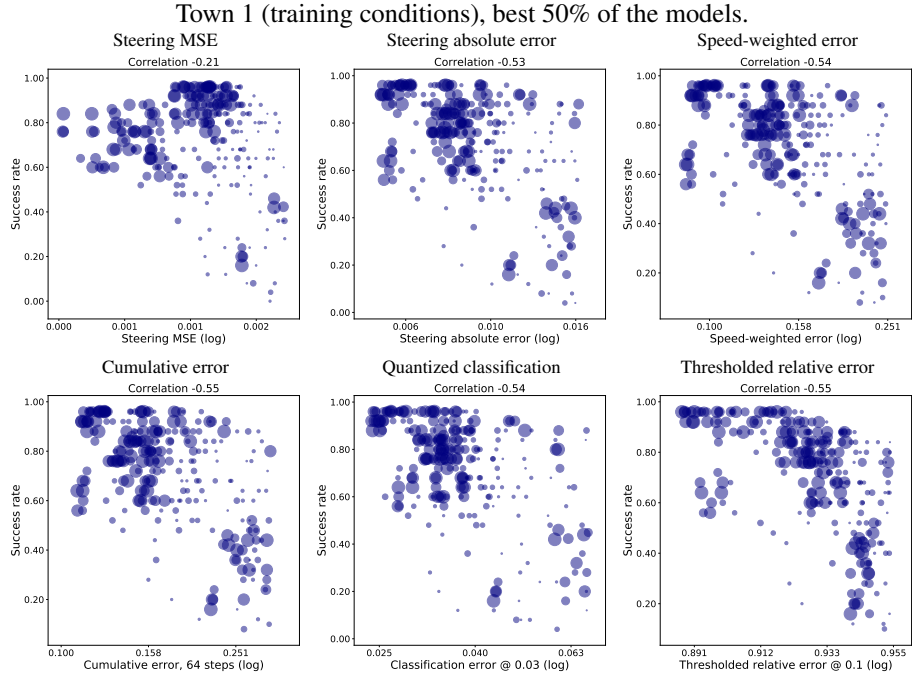
Town 1 (training conditions), best 50% of the models.

Fig. 2. Scatter plots of goal-directed navigation success rate vs different offline metrics. Town 1 (training conditions), best 50% of the models.
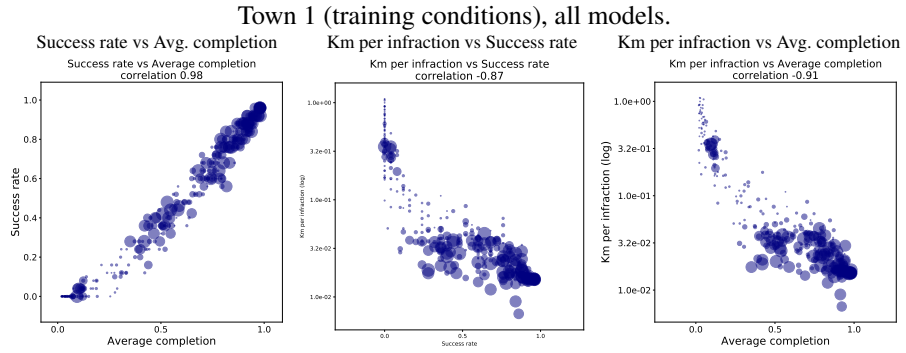
Town 1 (training conditions), all models.

Fig. 3. Scatter plots of online driving quality metrics versus each other. The metrics are: success rate, average fraction of distance to the goal covered (average completion), and average distance (in km) driven between two infractions. Town 1 (training conditions), all models.
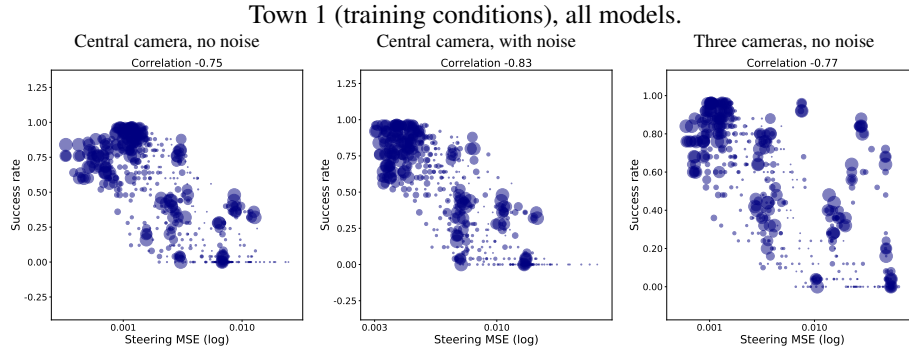
Town 1 (training conditions), all models.

Fig. 4. Scatter plots of goal-directed navigation success rate vs steering absolute error when evaluated on data from different distributions. Town 1 (training conditions), all models.
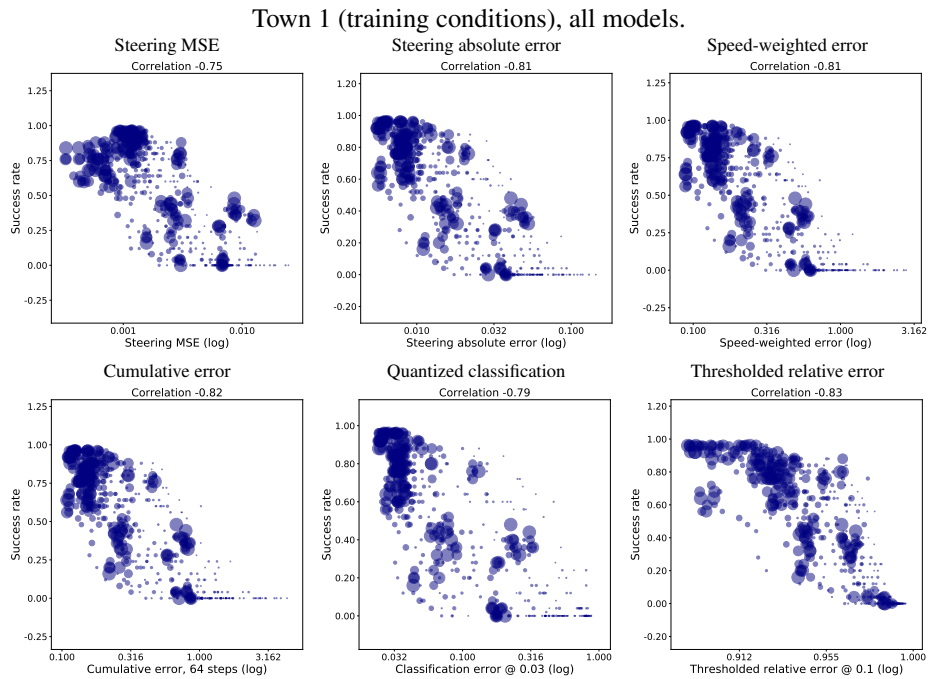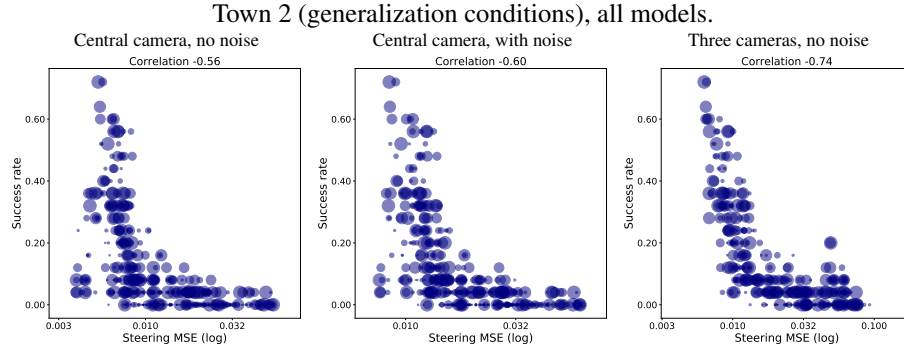
Town 1 (training conditions), all models.

Fig. 5. Scatter plots of goal-directed navigation success rate vs different offline metrics. Town 1 (training conditions), all models.

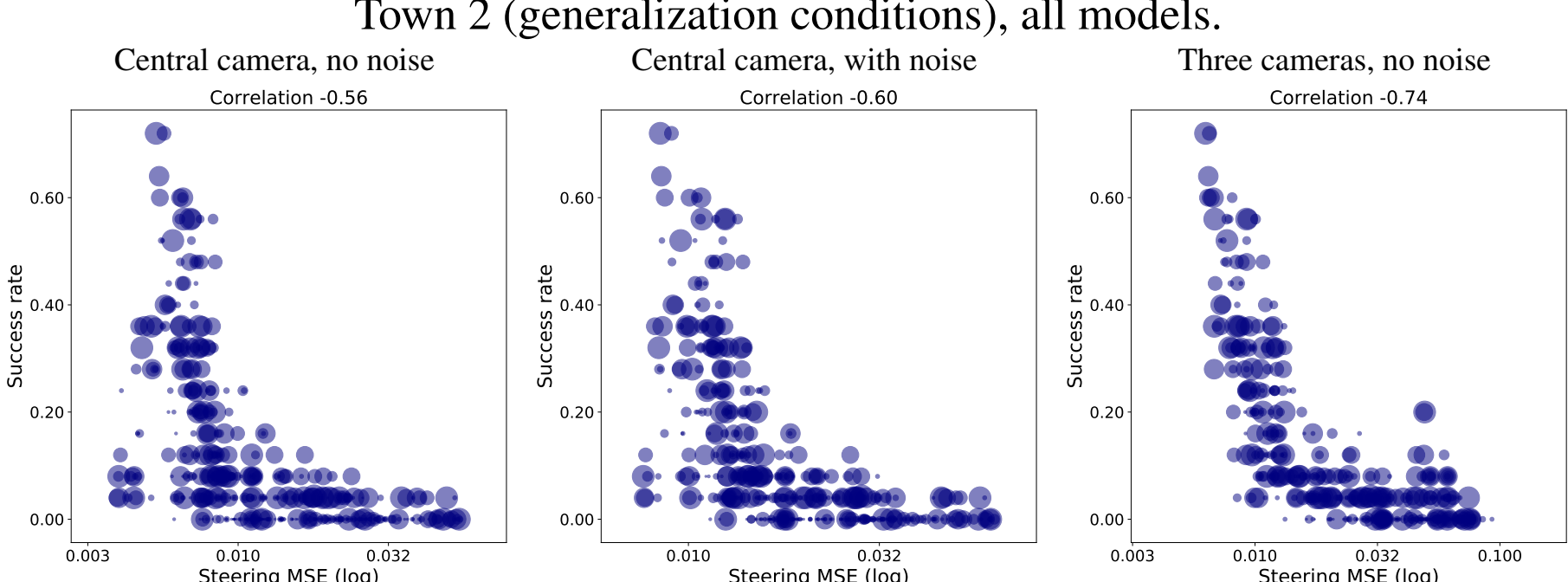

Fig. 6. Scatter plots of goal-directed navigation success rate vs steering absolute error when evaluated on data from different distributions. Town 2 (generalization conditions), all models.

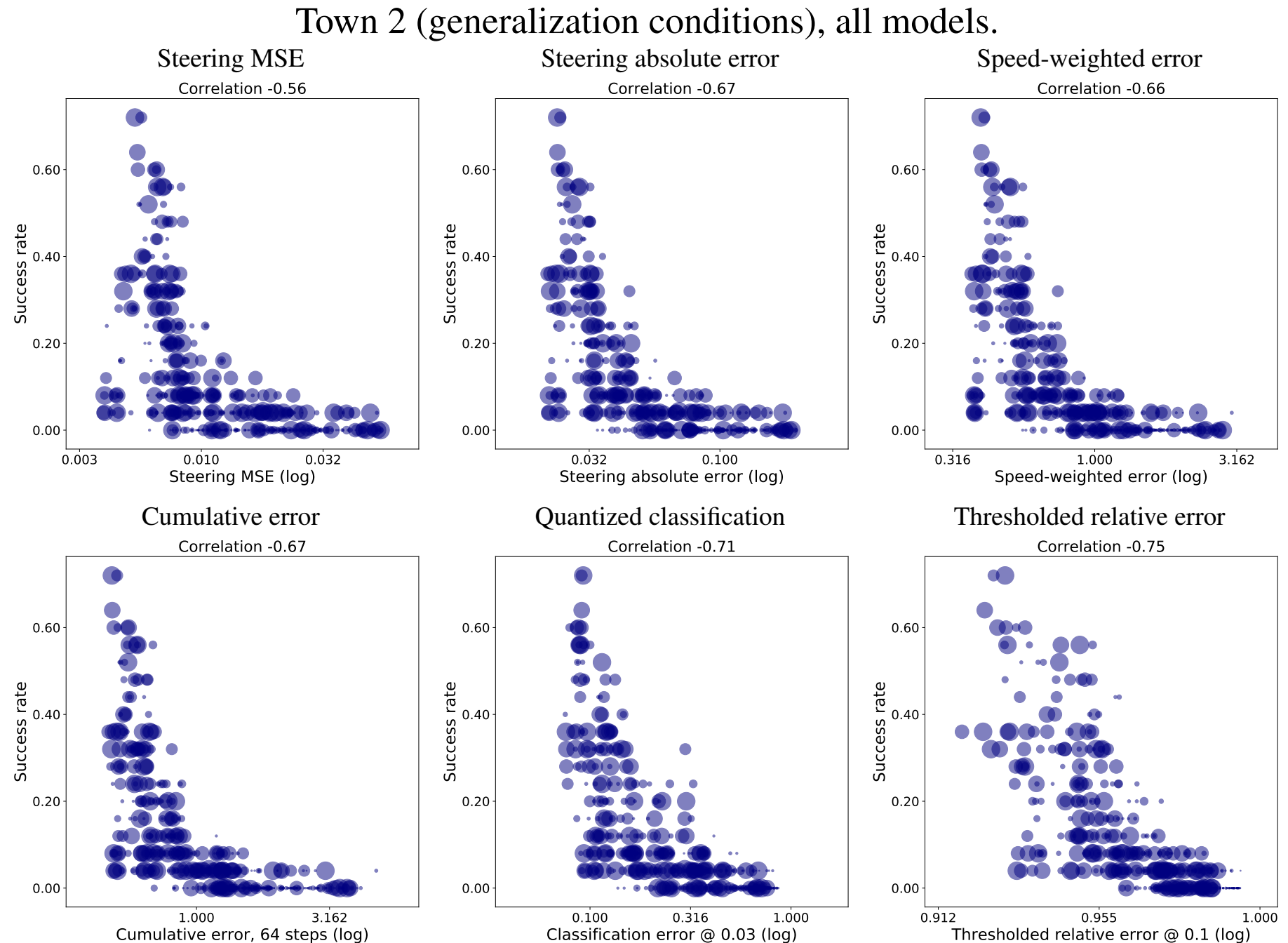

Fig. 7. Scatter plots of goal-directed navigation success rate vs different offline metrics. Town 2 (generalization conditions), all models.