

Supplementary Material for

“Behavioural Quality Is Not Learning Utility: Objective Mismatch in Human Behaviour Models for Robot Learning”

CONTENTS

I	Additional Experimental Details	2
I-A	Highway Driving	2
I-B	Object Handover	2
II	Highway Driving Model Details	2
II-A	Minimising Overall Braking Induced by Lane Changes (MOBIL) Model	2
II-B	Gipps’ Model	3
II-C	Krauß Model	4
II-D	Intelligent Driver Model (IDM)	4
II-E	Unif and Unif-W	5
II-F	MLP	5
II-G	MDN	6
II-H	Attention	7
II-I	Model Validation	8
III	Handover Giver Model Details	8
III-A	Minimum Jerk Model	8
III-B	Elliptical Model	8
III-C	Dynamic Movement Primitive (DMP)	9
III-D	MLP	11
III-E	MDN	11
III-F	Attention	11
III-G	Model Validation	11
IV	Trajectory Generation and Collection	12
IV-A	Highway Trajectory Generation	12
IV-B	Handover Trajectory Generation	12
V	Trajectory Representations	12
V-A	PCA Representation	13
V-B	Latent Representation	13
V-C	Manually Extracted Behavioural Features	14
VI	Evaluation Metrics	14
VI-A	Fréchet Distance (FD)	14
VI-B	Density	15
VI-C	Coverage	15
VI-D	Average Pair Distance (APD)	15
VI-E	Mean Maximum Similarity (MMS)	15
VI-F	KL Divergence	15
VII	Reinforcement Learning Details	15
VII-A	Highway Driving	15
VII-B	Object Handover	16
	References	16

I. ADDITIONAL EXPERIMENTAL DETAILS

A. Highway Driving

The highD dataset contains approximately 110,000 post-processed vehicle trajectories recorded by drones over six German highways near Cologne during 2017 and 2018. In total, the dataset comprises 60 recordings captured at 25 fps, each with an average duration of 17 minutes, covering highway segments of approximately 420 m in length [1].

To evaluate downstream policy performance, we train RL agents using Deep Q-Learning in the *highway-env* framework [2], a suite of environments designed for decision-making in autonomous driving. In the *Highway* task, the ego vehicle operates on a multi-lane highway populated with surrounding vehicles. At each time step, the agent observes its own position and velocity, together with the relative positions and velocities of nearby vehicles. Based on these observations, it selects one of five discrete actions: *Idle*, *Accelerate*, *Decelerate*, *Lane Left*, and *Lane Right*. The surrounding vehicles are controlled by one of the trained human behaviour models. The objective of the agent is to maintain a high speed while avoiding collisions, with additional rewards for driving in the rightmost lane.

We employ the *highway-fast-v0* environment, a computationally efficient variant of the standard *highway-v0*, characterised by a lower simulation frequency, fewer lanes (three), a reduced number of vehicles (20), and a shorter episode duration (30 seconds). This configuration facilitates faster RL training. All other environment and training hyperparameters are kept at their default settings, except that the behaviour model controlling the surrounding vehicles is replaced with one of our trained human behaviour models.

B. Object Handover

The handover@RPL dataset comprises 8,672 human-human handover trials collected from 13 participant pairs (26 participants in total). The experimental setup uses a custom 3D-printed baton equipped with three 6D force/torque sensors to measure the giver's grip force, the taker's grip force, and the central interaction wrench. These force measurements are synchronised at 120 Hz with an OptiTrack motion capture system that tracks a 13-segment upper-body skeletal representation of both participants. To investigate human motor adaptation under different conditions, the dataset contains three experimental settings: normal trials (2,999 handovers), a heavy-object setting in which external weights increase the baton mass from 0.8 kg to 1.8 kg (2,764 handovers), and a setting involving giver visual impairment (2,909 handovers). Individual handovers are post-processed and isolated into sequences of 6.675 s, centred around the temporal intersection of the participants' grip forces.

In this study, we focus on the giver's trajectory during the reverse handover task [3], specifically from the instant at which the giver establishes a stable grip on the baton until its release. This interval corresponds to the interaction phase of interest, during which the human behaviour model acts as the giver and the RL agent acts as the taker. Data

from Pair 13 are excluded because the participant is left-handed, whereas the remaining 12 participant pairs consist exclusively of right-handed participants, ensuring consistency in handedness throughout the dataset.

The raw trajectories are preprocessed in two stages. First, each trial is temporally clipped to retain only the interaction segment, with the grip force measured by the baton used to identify the start and end of the handover. Second, the resulting trajectories are filtered and denoised to reduce measurement noise and tracking artefacts. After preprocessing, 1,281 human giver trajectories remain for model training and evaluation.

We develop a customised environment for training the RL agent. The environment contains three entities: the giver's right hand, the taker's right hand, and a table. Both hands are modelled as point masses that move freely in three-dimensional space, without explicitly modelling human arm anatomy or kinematics.

Each hand is constrained to move within a sphere of radius 0.8 m, approximately corresponding to the average human arm length, centred at the respective shoulder position. The shoulder positions are determined by applying fixed offsets to the sampled initial hand positions. The table has dimensions of $1.2 \times 0.8 \times 0.8$ m [3] and is centred at the world origin. A safety margin of 0.1 m is imposed around the taker to prevent collisions with the table.

At the beginning of each episode, the initial positions of the giver and taker are sampled from the corresponding empirical distributions observed in the human data. An episode terminates when either the taker reaches the giver, defined as a giver-taker distance of less than 0.2 m, or when the maximum episode length of 60 steps is reached. The distance threshold is slightly shorter than the baton length reported in [3]. The reward function is defined as

$$r_t = 10(d_{t-1} - d_t) + 10\mathbb{I}(d_t < d_{\text{term}}), \quad (1)$$

where $d_t = \|\mathbf{p}_t^{\text{taker}} - \mathbf{p}_t^{\text{giver}}\|_2$ is the Euclidean distance between the taker's and giver's hands, $\mathbb{I}(\cdot)$ denotes the indicator function, and $d_{\text{term}} = 0.2$ m.

II. HIGHWAY DRIVING MODEL DETAILS

A. Minimising Overall Braking Induced by Lane Changes (MOBIL) Model

MOBIL is a general-purpose lane-changing framework that models both discretionary and mandatory lane changes by evaluating the impact of a potential lane change on the ego vehicle and surrounding vehicles. The highD dataset contains more than 16.5 hours of naturalistic highway driving recordings. However, lane-changing events are relatively rare in highway scenarios. Only around 14,000 lane-changing events are identified in the dataset, and only 5,600 of these are fully completed within the observed area [1]. This imbalance and limited effective sample size constrain the reliable identification and validation of lane-changing behaviour models. Furthermore, models such as MOBIL require an underlying longitudinal driving model to estimate the consequences of lane changes, such as resulting accelerations of surrounding vehicles. Since this longitudinal component must itself

be estimated from data, explicitly calibrating lane-changing behaviour would introduce additional modelling uncertainty. Therefore, to avoid compounding errors and overfitting to sparse lane-changing observations, we adopt standard MOBIL parameter values commonly used in the literature rather than estimating these parameters from the highD dataset.

In MOBIL, a lane change is executed only if two conditions are satisfied. First, the lane change must be safe for the new follower in the target lane:

$$\tilde{a}_n \geq -b_{\text{safe}}, \quad (2)$$

where $\tilde{a}_n$ denotes the post-lane-change acceleration of the new follower in the target lane, and b_{safe} is the maximum acceptable deceleration.

Second, the lane change must provide sufficient incentive:

$$\tilde{a}_c - a_c + p(\tilde{a}_n - a_n + \tilde{a}_o - a_o) \geq \Delta a_{\text{th}}, \quad (3)$$

where a_c and $\tilde{a}_c$ denote the accelerations of the centre (ego) vehicle before and after the lane change, respectively. Similarly, a_o and $\tilde{a}_o$ denote the accelerations of the old follower in the current lane before and after the lane change, and a_n and $\tilde{a}_n$ denote the corresponding accelerations of the new follower in the current and target lane. The parameter p is the politeness factor that weights the impact of the lane change on surrounding vehicles, and Δa_{th} is the incentive threshold governing the minimum required advantage for executing a lane change.

As discussed in the paper, we do not explicitly perform parameter estimation for the rule-based lane-changing models. The parameters of the MOBIL model are independently sampled for each vehicle from normal distributions, with means set to the typical values proposed in [4] for simulations:

$$\begin{aligned} p &\sim |X|, \quad X \sim \mathcal{N}(0, 0.25), \\ \Delta a_{\text{th}} &\sim \mathcal{N}(\mu_{\Delta a_{\text{th}}}, 0.5\mu_{\Delta a_{\text{th}}}), \\ b_{\text{safe}} &\sim \mathcal{N}(\mu_{b_{\text{safe}}}, 0.5\mu_{b_{\text{safe}}}), \end{aligned}$$

where

$$\begin{aligned} \mu_{\Delta a_{\text{th}}} &= 0.1 \text{ m/s}^2, \\ \mu_{b_{\text{safe}}} &= 4 \text{ m/s}^2. \end{aligned}$$

B. Gipps' Model

The Gipps model assumes that drivers maintain smooth and safe car-following behaviour subject to two constraints. First, the vehicle should not exceed its desired speed (i.e., the free-flow speed). Second, if the preceding vehicle performs an emergency stop, the following vehicle must still be able to decelerate safely and avoid a collision.

For the centre vehicle n and the leading vehicle $n-1$:

- $x_n(t)$: position of the vehicle
- $v_n(t)$: speed of the vehicle
- $x_{n-1}(t), v_{n-1}(t)$: position and speed of the leader
- $s_n(t) = x_{n-1}(t) - x_n(t) - l_{n-1} - L_{n-1}$: net gap
- l_{n-1} : physical length of a vehicle
- L_{n-1} : safe margin
- v_n^{max} : desired speed

- a_n : maximum acceleration
- b_n : comfortable deceleration
- b_{n-1} : assumed maximum deceleration of leader
- τ : apparent reaction time

The speed at the next time step is:

$$v_n(t + \tau) = \min \{v_n^{\text{free}}(t + \tau), v_n^{\text{safe}}(t + \tau)\}, \quad (4)$$

where the free-flow speed $v_n^{\text{free}}(t + \tau)$ of the Gipps' model is:

$$v_n^{\text{free}}(t + \tau) = v_n(t) + 2.5 a_n \tau \left(1 - \frac{v_n(t)}{v_n^{\text{max}}}\right) \sqrt{0.025 + \frac{v_n(t)}{v_n^{\text{max}}}}, \quad (5)$$

and the collision-avoidance speed $v_n^{\text{safe}}(t + \tau)$ is:

$$\begin{aligned} v_n^{\text{safe}}(t + \tau) &= b_n \tau \\ &+ \sqrt{(b_n \tau)^2 - b_n \left[2s_n(t) - v_n(t)\tau - \frac{v_{n-1}(t)^2}{b_{n-1}}\right]}. \end{aligned} \quad (6)$$

Finally, the position is updated as:

$$x_n(t + \tau) = x_n(t) + v_n(t + \tau) \tau. \quad (7)$$

In this work, we infer only the safe margin L_{n-1} and the desired speed v_n^{max} . All other model parameters are held fixed during inference and set to the typical values reported in [5]:

$$\begin{aligned} a_n &= 1.7 \text{ m/s}^2, \\ b_n &= -3.4 \text{ m/s}^2, \\ b_{n-1} &= -3 \text{ m/s}^2, \\ \tau &= 0.6 \text{ s}. \end{aligned}$$

Since the $\min\{\cdot\}$ operator is not differentiable, we decompose the likelihood into two separate components. Specifically, we infer the parameters associated with $v_n^{\text{free}}(t + \tau)$ and $v_n^{\text{safe}}(t + \tau)$ independently. To this end, the training data are partitioned based on the distance headway (DHW). Observations with a DHW shorter than 25 m are used to infer the safe margin L_{n-1} through the collision-avoidance component $v_n^{\text{safe}}(t + \tau)$, while observations with a DHW greater than or equal to 25 m are used to infer the desired speed v_n^{max} through the free-flow component $v_n^{\text{free}}(t + \tau)$. The threshold of 25 m is selected empirically. Both L_{n-1} and v_n^{max} are inferred using Automatic Differentiation Variational Inference (ADVI) [6]. For notational simplicity, we drop the subscripts $n-1$ and n from L_{n-1} and v_n^{max} in the following paragraphs, and simply refer to them as L and v_{max} , respectively.

To capture inter-driver variability while sharing statistical strength across vehicles, we assume a hierarchical prior structure. Specifically, the safe margin for vehicle i , denoted L_i , is drawn from a population-level distribution:

$$\begin{aligned} \mu_L &\sim \mathcal{N}(2.0, 1.0^2), \quad \sigma_L \sim \text{HalfNormal}(0.5), \\ L_i^* &\sim \mathcal{N}(\mu_L, \sigma_L^2), \quad L_i^* \in (0, \min(\text{DHW})), \end{aligned}$$

where the truncation enforces physical feasibility. To ensure strict positivity and numerical stability during inference, we apply a softplus transformation:

$$L_i = \log(1 + \exp(L_i^*)). \quad (8)$$

TABLE I
GIPPS' ADVI REPORT FOR L

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail
μ_L	2.486	0.068	2.358	2.610	0.002	0.001	1906.0	1931.0
σ_L	1.332	0.069	1.211	1.466	0.002	0.001	2030.0	2001.0
σ_{obs}	1.466	0.021	1.430	1.508	0.001	0.000	1793.0	1865.0

TABLE II
GIPPS' ADVI REPORT FOR v^{max}

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail
μ_v^{max}	32.336	0.216	31.926	32.734	0.005	0.003	2033.0	1882.0
σ_v^{max}	5.541	0.136	5.292	5.805	0.003	0.002	1914.0	1810.0
σ_{obs}	0.243	0.002	0.240	0.246	0.000	0.000	1975.0	1886.0

For each observation t associated with vehicle i , the predicted safe speed at time $t + \tau$ is given by Equation 6. Observed speeds $v_n(t + \tau)$ are assumed to be affected by Gaussian noise:

$$v(t + \tau) \sim \mathcal{N}(\mu_v(t + \tau), \sigma_{\text{obs}}^2), \quad \sigma_{\text{obs}} \sim \text{HalfNormal}(1.0). \quad (9)$$

Posterior inference is performed using ADVI, yielding approximate posterior distributions over both population-level hyperparameters (μ_L, σ_L) and vehicle-specific safe margins L_i . We employ a mean-field variational family and optimise the evidence lower bound (ELBO) for 30,000 iterations. After convergence, 2,000 posterior samples are drawn from the fitted variational approximation and used for all subsequent experiments.

Similarly, a hierarchical Bayesian formulation is employed to capture inter-driver variability for the desired speed v_i^{max} :

$$\mu_v^{\text{max}} \sim \mathcal{N}(20.0, 1.0^2), \quad \sigma_v^{\text{max}} \sim \text{HalfNormal}(0.5), \\ v_i^{\text{max}} = \log(1 + \exp(v_i^{* \text{max}})), \quad v_i^{* \text{max}} \sim \mathcal{N}(\mu_v^{\text{max}}, \sigma_v^{\text{max} 2}).$$

The predicted next speed under free-flow conditions is given by Equation 5 with Gaussian noise applied identical to Equation 9. Posterior inference is performed using mean-field ADVI for 100,000 iterations, after which 2,000 posterior samples are drawn from the variational approximation for downstream experiments.

We report the ADVI results for population-level hyperparameters $(\mu_L, \sigma_L, \sigma_{\text{obs}})$ and $(\mu_v^{\text{max}}, \sigma_v^{\text{max}}, \sigma_{\text{obs}})$ in Table I and II, respectively.

C. Krauß Model

The Krauß model determines vehicle motion by selecting a safe speed from dynamically feasible bounds, rather than explicitly solving a worst-case stopping constraint to obtain a single admissible next speed. Due to its computational efficiency and robustness, the model has been widely adopted in traffic simulators such as SUMO [7].

Using the same notation as the Gipps' model in Section II-B, the safe speed given by the Krauß model is:

$$v_{\text{safe}}(t) = v_{n-1}(t) + \frac{d_n(t) - v_{n-1}(t)\tau}{\frac{v_n(t) + v_{n-1}(t)}{2b_n} + \tau}, \quad (10)$$

where $d_n(t) = x_{n-1}(t) - x_n(t) - l_{n-1}$, which does not involve the safe margin. At each discrete time step, the vehicle chooses its speed according to:

$$v_n(t + \Delta t) = \min(v_{\text{safe}}(t), v_n(t) + a_n \Delta t, v_n^{\text{max}}) \quad (11)$$

Note that in the Krauß model, the numerical update time step Δt is distinct from the driver reaction time τ . In this study, we infer only the reaction time τ , while all other model parameters are fixed to the same values as those used for the Gipps' model. An exception is the maximum deceleration parameter b_n , which is specified as a positive quantity (e.g., $b_n = 3.4 \text{ m/s}^2$).

As with the Gipps' model, the Krauß model involves non-differentiable components. To address this issue, we adopt the same strategy employed for the Gipps' model. The likelihood of the safe speed is evaluated independently to infer the reaction time τ , using only training observations with a DHW shorter than 25 m.

To capture inter-driver variability in reaction time, we employ a hierarchical Bayesian formulation for τ :

$$\mu_\tau \sim \mathcal{N}(0.8, 0.3^2), \quad \sigma_\tau \sim \text{HalfNormal}(0.5), \\ \tau_i = \log(1 + \exp(\tau_i^*)), \quad \tau_i^* \sim \mathcal{N}(\mu_\tau, \sigma_\tau^2).$$

The predicted next speed is given by Equation 10 with Gaussian noise applied identical to Equation 9. Posterior inference is performed using mean-field ADVI for 20,000 iterations, after which 2,000 posterior samples are drawn from the variational approximation for downstream experiments. We report the ADVI results for population-level hyperparameters $(\mu_\tau, \sigma_\tau, \sigma_{\text{obs}})$ in Table III.

D. Intelligent Driver Model (IDM)

While both the Gipps and Krauß models are first-order formulations, the IDM is a second-order model that explicitly incorporates both vehicle velocity and acceleration when computing longitudinal dynamics.

The acceleration of an IDM vehicle is computed at each step according to:

$$\dot{v}_n(t) = a_n \left[1 - \left(\frac{v_n(t)}{v_{\text{max}}} \right)^\delta - \left(\frac{d_n^*(t)}{d_n} \right)^2 \right], \quad (12)$$

TABLE III
KRAUSS ADVI REPORT FOR τ

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail
μ_τ	0.087	0.051	-0.006	0.184	0.001	0.001	1695.0	1766.0
σ_τ	1.341	0.045	1.256	1.423	0.001	0.001	1930.0	1743.0
σ_{obs}	0.995	0.020	0.959	1.034	0.000	0.000	1896.0	1885.0

TABLE IV
IDM ADVI REPORT FOR v_n^{max} , d_0 AND T

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail
μ_v^{max}	32.671	0.096	32.491	32.848	0.002	0.002	1993.0	1961.0
σ_v^{max}	1.249	0.085	1.102	1.408	0.002	0.001	2148.0	2004.0
μ_{d_0}	0.371	0.088	0.210	0.540	0.002	0.001	2005.0	1985.0
σ_{d_0}	1.338	0.085	1.183	1.504	0.002	0.001	1981.0	1984.0
μ_T	-0.812	0.086	-0.979	-0.664	0.002	0.001	1996.0	1960.0
σ_T	1.258	0.084	1.110	1.427	0.002	0.001	2065.0	1698.0
σ_{obs}	2.094	0.137	1.863	2.371	0.003	0.002	2102.0	1876.0

where

$$d_n^*(t) = d_0 + T \cdot v_n(t) + \frac{v_n(t)(v_n(t) - v_{n-1}(t))}{2\sqrt{a_n \cdot b_n}}. \quad (13)$$

The IDM introduces three hyperparameters that do not appear in the Gipps or Krauß models, namely the acceleration exponent δ , the safe time headway T , and the jam distance d_0 . The acceleration exponent δ does not have a direct physical interpretation and is typically fixed to $\delta = 4$. In this study, we infer three IDM parameters: the desired velocity v^{max} , the safe time headway T , and the jam distance d_0 . All remaining parameters are fixed to the standard values proposed in [8]:

$$\begin{aligned} \delta &= 4; \\ a_n &= 0.73 \text{ m/s}^2; \\ b_n &= 1.67 \text{ m/s}^2. \end{aligned}$$

Unlike the Gipps' and Krauß models, the IDM is fully differentiable. Consequently, all available observations are used for training the IDM. A similar hierarchical Bayesian formulation is adopted:

$$\begin{aligned} \mu_v^{\text{max}} &\sim \mathcal{N}(30.0, 1.0^2), & \sigma_v^{\text{max}} &\sim \text{HalfNormal}(0.5), \\ v_i^{\text{max}} &= \log(1 + \exp(v_i^{\text{max}})), & v_i^{\text{max}} &\sim \mathcal{N}(\mu_v^{\text{max}}, \sigma_v^{\text{max}2}), \\ \mu_{d_0} &\sim \mathcal{N}(2.0, 1.0^2), & \sigma_{d_0} &\sim \text{HalfNormal}(1.0), \\ d_{0i} &= \log(1 + \exp(d_{0i}^*)), & d_{0i}^* &\sim \mathcal{N}(\mu_{d_0}, \sigma_{d_0}^2), \\ \mu_T &\sim \mathcal{N}(1.5, 0.5^2), & \sigma_T &\sim \text{HalfNormal}(0.5), \\ T_i &= \log(1 + \exp(T_i^*)), & T_i^* &\sim \mathcal{N}(\mu_T, \sigma_T^2). \end{aligned}$$

The likelihood is given by Equation 12 and 13. Posterior inference is performed using mean-field ADVI for 15,000 iterations, after which 2,000 posterior samples are drawn from the variational approximation for downstream experiments.

We report the ADVI results for population-level hyperparameters (μ_v^{max} , σ_v^{max} , μ_{d_0} , σ_{d_0} , μ_T , σ_T , σ_{obs}) in Table IV.

E. Unif and Unif-W

Unif samples longitudinal and lateral accelerations from $U(-4, 3)$ and $U(-2, 2)$, respectively, whereas *Unif-W* uses

the same longitudinal range but a wider lateral acceleration range of $U(-6, 6)$.

F. MLP

Behaviour cloning is an imitation learning approach in which an agent learns to reproduce expert behaviour by formulating the problem as supervised learning over state-action pairs. Given demonstrations from an expert policy, a model is trained to predict the expert action conditioned on the current state, without requiring an explicit reward function. In this study, expert demonstrations are obtained from the highD highway driving dataset.

For all three data-driven models, trajectories extracted from the highD dataset are first processed into structured feature representations comprising both ego-vehicle states and surrounding vehicle states before being provided as model inputs. Specifically, the input state is defined in an ego-centric coordinate frame and consists of the following components:

- Longitudinal velocity of the ego vehicle, $v_e^{\text{lon}} \in \mathbb{R}$;
- Lateral velocity of the ego vehicle, $v_e^{\text{lat}} \in \mathbb{R}$;
- Lateral offset from the lane centre, $\delta y_e = y_e - y_{\text{lane centre}}$;
- Physical length of the ego vehicle, l_e ;
- Binary indicators denoting the existence of an adjacent lane to the left and right, $(\mathbb{1}_{\text{left}}, \mathbb{1}_{\text{right}})$;
- Relative states of seven surrounding vehicles: $s_p, s_{la}, s_{lp}, s_{lf}, s_{ra}, s_{rp}$, and s_{rf} , corresponding to the preceding vehicle, left-along, left-preceding, left-following, right-along, right-preceding, and right-following vehicles, respectively.

A surrounding vehicle is classified as left-along or right-along if its longitudinal projection overlaps with that of the ego vehicle. The relative state of a surrounding vehicle i is defined as

$$s_i = [\mathbb{1}_i, l_i, \Delta x_i, \Delta y_i, \Delta v_i^{\text{lon}}, \Delta v_i^{\text{lat}}], \quad (14)$$

$$i \in \{p, la, lp, lf, ra, rp, rf\},$$

where $\mathbb{1}_i$ indicates the existence of vehicle i , l_i denotes its physical length, $\Delta x_i = x_i - x_e$ and $\Delta y_i = y_i - y_e$ are

the relative longitudinal and lateral distances, and Δv_i^{lon} and Δv_i^{lat} denote the relative longitudinal and lateral velocities with respect to the ego vehicle. If vehicle i does not exist, all values in s_i are set to zero.

The complete state vector is given by

$$\mathbf{s} = [v_e^{\text{lon}}, v_e^{\text{lat}}, l_e, \delta y_e, \mathbb{1}_{\text{left}}, \mathbb{1}_{\text{right}}, s_p, s_{la}, s_{lp}, s_{lf}, s_{ra}, s_{rp}, s_{rf}], \quad (15)$$

resulting in a 48-dimensional input representation.

The policy network outputs continuous control actions,

$$\mathbf{a} = [\hat{a}^{\text{lon}}, \hat{a}^{\text{lat}}], \quad (16)$$

where $\hat{a}^{\text{lon}} \in \mathbb{R}$ denotes the predicted longitudinal acceleration and $\hat{a}^{\text{lat}} \in \mathbb{R}$ denotes the predicted lateral acceleration.

The MLP model is trained using a mean squared error loss (MSE) over the continuous action space:

$$\mathcal{L}_{\text{MLP}} = \frac{1}{B} \sum_{i=1}^B \|\hat{\mathbf{a}}_i - \mathbf{a}_i^{\text{gt}}\|_2^2, \quad (17)$$

where B denotes the batch size and $\mathbf{a}_i^{\text{gt}}$ is the ground-truth action extracted from the dataset.

The raw highD dataset is recorded at 25 fps, resulting in only minor differences between consecutive samples. To reduce temporal redundancy during training, we downsample the dataset to 5 fps. Prior to training, the state-action pairs are standardised.

Specifically, all continuous state and action variables, including vehicle length, are standardised, while binary indicator features are retained as 0 or 1. For relative continuous variables such as Δx_i , a value of zero may correspond either to the non-existence of a vehicle or to zero relative distance. Standardising these variables across all samples would therefore introduce semantic ambiguity. To avoid this issue, we standardise relative state components only for samples where the corresponding presence indicator satisfies $\mathbb{1}_i = 1$. Otherwise, all relative-state values are set to zero. We split the dataset into 80% for training, 10% for validation, and 10% for testing, resulting in approximately 1.7 million training samples and around 0.2 million samples each for validation and testing.

We employ a simple MLP with two hidden layers of 64 units each, using ReLU activations throughout. The batch size is set to 1024, and the learning rate is fixed at 10^{-3} . Training is performed for a maximum of 100 epochs with early stopping applied using a patience of 10 epochs. Early stopping is triggered at the 88th epoch, yielding the best test loss of 0.62.

G. MDN

A mixture density network models a target variable as a weighted mixture of probability distributions (typically Gaussians). Instead of predicting a single point estimate, it outputs the mixture weights and parameters, enabling it to represent multimodal and uncertain relationships between inputs and outputs. We employ a MDN in this study because human driving behaviour is inherently multimodal. For example, drivers may adopt different strategies for car-following and lane-changing.

We reuse the same state representation and action space as those used for BC with an MLP. The MDN models the conditional action distribution as a mixture of Gaussians:

$$p(\mathbf{a} | \mathbf{s}) = \sum_{k=1}^K \alpha_k(\mathbf{s}) \mathcal{N}(\mathbf{a} | \boldsymbol{\mu}_k(\mathbf{s}), \boldsymbol{\Sigma}_k(\mathbf{s})), \quad (18)$$

where $\alpha_k(\mathbf{s})$ denotes the mixing coefficient of the k -th component, $\boldsymbol{\mu}_k \in \mathbb{R}^2$ is the mean vector, and $\boldsymbol{\Sigma}_k \in \mathbb{R}^{2 \times 2}$ is the covariance matrix.

In this study, we assume independence between longitudinal and lateral accelerations, and thus restrict the covariance matrices to be diagonal:

$$\boldsymbol{\Sigma}_k = \begin{pmatrix} \sigma_{k,\text{lon}}^2 & 0 \\ 0 & \sigma_{k,\text{lat}}^2 \end{pmatrix}. \quad (19)$$

The MDN is trained by minimising a composite loss that extends the standard negative log-likelihood (NLL) with additional regularisation terms to encourage mode utilisation and diversity. For a batch of size B , the loss is:

$$\begin{aligned} \mathcal{L}_{\text{MDN}} = & \underbrace{-\frac{1}{B} \sum_{i=1}^B \log \left(\sum_{k=1}^K \alpha_k^{(i)} \mathcal{N}(\mathbf{a}^{\text{gt},(i)} | \boldsymbol{\mu}_k^{(i)}, \boldsymbol{\Sigma}_k^{(i)}) \right)}_{\text{NLL}} \\ & + \lambda_{\text{usage}} \underbrace{D_{\text{KL}}(\bar{\mathbf{r}} \| U)}_{\text{Mode usage penalty}} \\ & + \lambda_{\text{div}} \underbrace{\frac{1}{K(K-1)B} \sum_{i=1}^B \sum_{k \neq j} \exp(-D_{\text{KL}}(\mathcal{N}_k^{(i)} \| \mathcal{N}_j^{(i)}))}_{\text{Component diversity penalty}}, \end{aligned} \quad (20)$$

where $\alpha_k^{(i)}$ is the mixture weight for the k -th component of sample i , and $\boldsymbol{\mu}_k^{(i)}$ and $\boldsymbol{\Sigma}_k^{(i)}$ are the corresponding mean and covariance. The vector $\bar{\mathbf{r}} = \frac{1}{B} \sum_{i=1}^B \mathbf{r}^{(i)}$ is the average responsibility vector over the batch, where each $\mathbf{r}^{(i)} \in \mathbb{R}^K$ contains the posterior probabilities (responsibilities) of the K mixture components for sample i :

$$r_k^{(i)} = \frac{\alpha_k^{(i)} \mathcal{N}(\mathbf{a}^{\text{gt},(i)} | \boldsymbol{\mu}_k^{(i)}, \boldsymbol{\Sigma}_k^{(i)})}{\sum_{j=1}^K \alpha_j^{(i)} \mathcal{N}(\mathbf{a}^{\text{gt},(i)} | \boldsymbol{\mu}_j^{(i)}, \boldsymbol{\Sigma}_j^{(i)})}. \quad (21)$$

And U represents a uniform distribution over the mixture components. $\mathcal{N}_k^{(i)}$ or $\mathcal{N}_j^{(i)}$ denotes the Gaussian for component k or j of sample i , respectively.

The first term encourages the model to fit the ground-truth actions, the second term discourages mode collapse by promoting uniform usage of mixture components across the batch, and the third term encourages diversity between components by penalising overlapping or redundant Gaussians. The hyperparameters λ_{usage} and λ_{div} control the strength of these regularisation terms.

The same training, validation, and test splits used for the MLP-based behaviour cloning model are also used for training and evaluating the MDN. We employ a simple MDN with two hidden layers of 128 units each and ReLU activations throughout, producing $K = 15$ mixture components. The

batch size is set to 512, and the learning rate is fixed at 10^{-3} . Training is conducted for a maximum of 100 epochs with early stopping applied using a patience of 10 epochs. The regularisation weights λ_{usage} and λ_{div} are initially set to 0.05 and 0.01, respectively. The diversity weight λ_{div} is annealed exponentially during training according to

$$\lambda_{\text{div},e} = \lambda_{\text{div},0} \exp(-\gamma \cdot e),$$

where $\lambda_{\text{div},0}$ is the initial value, $\gamma = 0.05$ is the decay factor, and e denotes the current epoch number. Early stopping is triggered at the 48th epoch, resulting in the test loss of 1.88.

H. Attention

The MLP and MDN models do not utilise the history of states, which can be informative for trajectory forecasting. To address this, we adopt an attention-based design that incorporates past states into the prediction. The multi-head attention (MHA) function, as proposed in the Transformer [9], has been widely used to model interactions between elements in a sequence. Given queries $Q \in \mathbb{R}^{N_q \times d_k}$, keys $K \in \mathbb{R}^{N_k \times d_k}$, and values $V \in \mathbb{R}^{N_k \times d_v}$, a single attention head computes

$$\text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V, \quad (22)$$

where d_k and d_v are the dimension of the keys and values, and N_q and N_k are the number of tokens of queries and keys, respectively. Multi-head attention splits the input into h heads, applies the attention function in parallel, and concatenates the results:

$$\text{MHA}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W_O, \quad (23)$$

where each head is $\text{head}_i = \text{Attn}(QW_Q^i, KW_K^i, VW_V^i)$ and W_Q^i, W_K^i, W_V^i, W_O are learnable projections.

In this study, we use the MHA function to learn correlations between the ego vehicle's history states and its current state. Following the same notation as used in MLP and MDN, we denote the state of the ego vehicle at time t as $\mathbf{s}_{\text{ego}}^t \in \mathbb{R}^6$:

$$\mathbf{s}_{\text{ego}}^t = [v_e^{\text{lon}}(t), v_e^{\text{lat}}(t), l_e, \delta y_e(t), \mathbb{1}_{\text{left}}(t), \mathbb{1}_{\text{right}}(t)], \quad (24)$$

and its history over h steps as

$$\mathbf{s}_{\text{ego}}^{t-h:t-1} = [\mathbf{s}_{\text{ego}}^{t-h}, \mathbf{s}_{\text{ego}}^{t-h+1}, \dots, \mathbf{s}_{\text{ego}}^{t-1}] \in \mathbb{R}^{h \times 6}. \quad (25)$$

We then apply a cross-attention function that allows the current ego state $\mathbf{s}_{\text{ego}}^t$ to query its history states $\mathbf{s}_{\text{ego}}^{t-h:t-1}$, producing the ego embedding $\mathbf{z}_{\text{ego}} \in \mathbb{R}^{H \cdot d_z}$, where H is the number of attention heads and d_z is the embedding dimension (also the dimension of the values):

$$\mathbf{z}_{\text{ego}} = \text{Concat}_H(\text{MHA}(\mathbf{s}_{\text{ego}}^t, \mathbf{s}_{\text{ego}}^{t-h:t-1}, \mathbf{s}_{\text{ego}}^{t-h:t-1})). \quad (26)$$

This is followed by another cross-attention, in which the embedded ego state $\mathbf{z}_{\text{ego}}$ queries the states of the seven surrounding vehicles:

$$\mathbf{s}_{\text{surr}} = [\mathbf{s}_i^{t-h:t}], \quad i \in \{p, la, lp, lf, ra, rp, rf\}, \quad (27)$$

where each $\mathbf{s}_i^t$ is the corresponding relative state of surrounding vehicle i at time t as defined in paper Equation

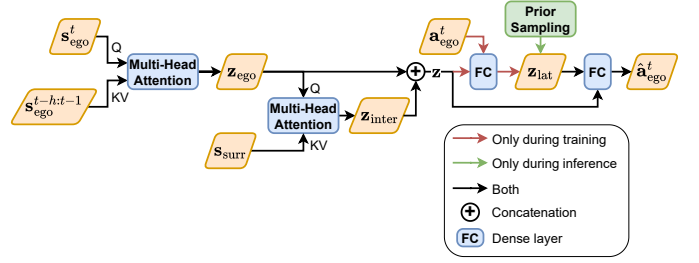


Fig. 1. An illustration of the attention-based model architecture. At each time step t , the current ego state $\mathbf{s}_{\text{ego}}^t \in \mathbb{R}^6$ (where bold symbols denote vectors) queries its past h ego states $\mathbf{s}_{\text{ego}}^{t-h:t-1} \in \mathbb{R}^{h \times 6}$ to produce the ego embedding $\mathbf{z}_{\text{ego}} \in \mathbb{R}^{H \cdot d_z}$. This embedding then queries the states of the N surrounding vehicles ($N = 7$ in our case) $\mathbf{s}_{\text{surr}} \in \mathbb{R}^{N \times (h+1) \times 6}$ to generate the interaction embedding $\mathbf{z}_{\text{inter}} \in \mathbb{R}^{H \cdot d_z}$. The ego and interaction embeddings are concatenated into a single scene representation $\mathbf{z} = \text{Concat}(\mathbf{z}_{\text{ego}}, \mathbf{z}_{\text{inter}}) \in \mathbb{R}^{2H \cdot d_z}$, which is used as the conditioning input to a Conditional Variational Autoencoder (CVAE) [10]. During training, the encoder takes both $\mathbf{z}$ and the ground-truth control action $\mathbf{a}_{\text{ego}}^t \in \mathbb{R}^2$ to infer a latent variable $\mathbf{z}_{\text{lat}}$. The decoder then generates the control action $\hat{\mathbf{a}}_{\text{ego}}^t \in \mathbb{R}^2$ conditioned on the concatenation of $\mathbf{z}$ and a sampled latent vector. During inference, $\mathbf{z}_{\text{lat}}$ is sampled from a prior distribution.

7. The surrounding states $\mathbf{s}_{\text{surr}} \in \mathbb{R}^{N \times (h+1) \times 6}$ are reshaped into a two-dimensional tensor of size $\mathbb{R}^{N(h+1) \times 6}$ prior to the attention operation. This transformation yields a sequence of $N(h+1)$ tokens, each corresponding to the state of a surrounding vehicle at a specific time step. The cross-attention MHA outputs the embedding of interactions between the ego and surrounding vehicles:

$$\mathbf{z}_{\text{inter}} = \text{Concat}_H(\text{MHA}(\mathbf{z}_{\text{ego}}, \mathbf{s}_{\text{surr}}, \mathbf{s}_{\text{surr}})) \in \mathbb{R}^{H \cdot d_z}. \quad (28)$$

Finally, we concatenate the ego and interaction embeddings into a single vector,

$$\mathbf{z} = \text{Concat}(\mathbf{z}_{\text{ego}}, \mathbf{z}_{\text{inter}}) \in \mathbb{R}^{2H \cdot d_z}. \quad (29)$$

We use a conditional variational autoencoder (CVAE) [10] to model the distribution over control actions in order to capture the inherent multi-modality and uncertainty in the ego vehicle's future actions. During training, the encoder takes both $\mathbf{z}$ and the ground-truth action $\mathbf{a}_{\text{ego}}^t \in \mathbb{R}^2$ to compute the posterior distribution over the latent variable:

$$q_\phi(\mathbf{z}_{\text{lat}} | \mathbf{a}_{\text{ego}}^t, \mathbf{z}) = \mathcal{N}(\boldsymbol{\mu}_\phi, \text{diag}(\boldsymbol{\sigma}_\phi^2)), \quad \mathbf{z}_{\text{lat}} \in \mathbb{R}^{d_{\text{lat}}}. \quad (30)$$

A latent sample $\mathbf{z}_{\text{lat}}$ is drawn using the reparameterisation trick and concatenated with the embedding $\mathbf{z}$ to produce the predicted action via the decoder:

$$\hat{\mathbf{a}}_{\text{ego}}^t = f_\theta([\mathbf{z}, \mathbf{z}_{\text{lat}}]) \in \mathbb{R}^2. \quad (31)$$

During inference, since $\mathbf{a}_{\text{ego}}^t$ is unavailable, the latent vector is sampled from the prior

$$\mathbf{z}_{\text{lat}} \sim \mathcal{N}(0, I), \quad (32)$$

and the decoder generates one or multiple plausible control actions conditioned on $\mathbf{z}$. The architecture of the customised attention-based model is illustrated in Figure 1.

The CVAE is trained by minimising the negative variational lower bound (ELBO), which consists of a reconstruction loss

and a Kullback–Leibler (KL) divergence regularisation term. Given the conditioning embedding $\mathbf{z}$ and the ground-truth ego action $\mathbf{a}_{\text{ego}}$, the objective is defined as

$$\mathcal{L}_{\text{CVAE}} = \mathbb{E}_{q_{\phi}(\mathbf{z}_{\text{lat}} | \mathbf{a}_{\text{ego}}, \mathbf{z})} [\mathcal{L}_{\text{rec}}(\mathbf{a}_{\text{ego}}, \hat{\mathbf{a}}_{\text{ego}})] + \beta D_{\text{KL}}(q_{\phi}(\mathbf{z}_{\text{lat}} | \mathbf{a}_{\text{ego}}, \mathbf{z}) \parallel p(\mathbf{z}_{\text{lat}})), \quad (33)$$

where $q_{\phi}(\mathbf{z}_{\text{lat}} | \mathbf{a}_{\text{ego}}, \mathbf{z})$ denotes the encoder posterior, and the prior is chosen as a standard normal distribution $p(\mathbf{z}_{\text{lat}}) = \mathcal{N}(0, I)$. The scalar β controls the trade-off between reconstruction accuracy and latent regularisation.

The reconstruction loss $\mathcal{L}_{\text{rec}}$ is defined as the NLL of the decoder output. Assuming a Gaussian likelihood with fixed diagonal covariance, it reduces to a mean squared error:

$$\mathcal{L}_{\text{rec}} = \|\mathbf{a}_{\text{ego}} - \hat{\mathbf{a}}_{\text{ego}}\|_2^2. \quad (34)$$

The KL divergence term has a closed-form expression:

$$D_{\text{KL}} = \frac{1}{2} \sum_{i=1}^{d_{\text{lat}}} (\mu_i^2 + \sigma_i^2 - \log \sigma_i^2 - 1), \quad (35)$$

where μ and σ are the mean and standard deviation predicted by the encoder.

This architecture can be viewed as a simplified attention-based variant of *Trajectron++*, a graph-based recurrent model for trajectory forecasting [11]. In contrast to *Trajectron++*, the long short-term memory (LSTM) modules [12] are replaced by attention mechanisms to model temporal dependencies, and continuous control actions are predicted directly instead of future trajectories generated by gated recurrent units (GRUs) [13]. Moreover, we do not incorporate road context maps or explicit group agents into categories, as the vehicle type is implicitly encoded in the state representation through the vehicle length.

We set the history length to $h = 4$, and use the same training, validation, and test splits as those adopted for the MLP and MDN models. The attention-based architecture employs an embedding dimension of 32 and 4 attention heads, with a dropout rate of 0.1, layer normalisation, and residual connections applied to both cross-attention modules.

For the CVAE, the encoder consists of a two-layer MLP with a hidden dimension of 64, while the decoder is a three-layer MLP with the same hidden dimension. The latent dimension is set to 16. During training, we use a batch size of 64, a learning rate of 10^{-3} , and the AdamW optimiser with a weight decay of 10^{-4} . Gradient clipping is applied with a maximum norm of 1.0, and the weighting coefficient is fixed at $\beta = 1.0$. Training is conducted for a maximum of 200 epochs with early stopping patience set to 30 epochs. We do not apply β -annealing, as preliminary experiments indicate that both linear and cyclical annealing degrade performance. Early stopping is triggered at the 41st epoch, achieving a best test loss of 0.52.

I. Model Validation

To verify that the calibrated models reproduce the statistical characteristics of naturalistic driving, we compare the distributions of key driving features generated by each model

against those observed in the highD data. Figure 2 shows the corresponding kernel density estimates, while Table V summarises the feature statistics and Table VI reports the corresponding KL divergences. The results show that the calibrated models generally capture the central tendencies of the data, although their variability differs across features. In particular, the data-driven models MLP, MDN, and Attention achieve lower KL divergence for several features, while the rule-based models also reproduce the observed distributions reasonably well. Overall, these results confirm that the calibrated models capture meaningful distributional characteristics of naturalistic driving.

III. HANDOVER GIVER MODEL DETAILS

A. Minimum Jerk Model

The Minimum Jerk model is a classical model of human reaching movement that assumes humans generate trajectories by minimising the integrated squared jerk, where jerk is the third derivative of position with respect to time [14]. Given an initial position $\mathbf{p}_0$ and a target position $\mathbf{p}_T$, the model generates a smooth trajectory $\mathbf{p}(t)$ by solving

$$\min_{\mathbf{p}(t)} \int_0^T \left\| \frac{d^3 \mathbf{p}(t)}{dt^3} \right\|_2^2 dt, \quad (36)$$

subject to the boundary conditions

$$\begin{aligned} \mathbf{p}(0) &= \mathbf{p}_0, & \mathbf{p}(T) &= \mathbf{p}_T, \\ \dot{\mathbf{p}}(0) &= \dot{\mathbf{p}}(T) = \mathbf{0}, & \ddot{\mathbf{p}}(0) &= \ddot{\mathbf{p}}(T) = \mathbf{0}. \end{aligned} \quad (37)$$

The resulting trajectory can be expressed as

$$\mathbf{p}(t) = \mathbf{p}_0 + (10\tau^3 - 15\tau^4 + 6\tau^5) (\mathbf{p}_T - \mathbf{p}_0), \quad (38)$$

where $\tau = t/T \in [0, 1]$ denotes the normalised movement time.

In our handover setting, the Minimum Jerk model is used to generate the trajectory of the human giver's hand from its initial position towards the handover location. The model therefore produces a smooth, deterministic movement between the initial and target positions, with the movement duration T controlling the temporal evolution of the trajectory. The Minimum Jerk model has no learnable parameters. However, generating a trajectory requires specifying its initial position, final position, and duration. To obtain these quantities, we model the distributions of the trajectory start positions, end positions, and lengths using Gaussian distributions fitted to the corresponding quantities in the human training data. During deployment, these quantities are independently sampled from their respective Gaussian distributions and used to generate a Minimum Jerk trajectory.

B. Elliptical Model

The Elliptical model represents each human trajectory as a planar movement along an ellipse. For each demonstration, we first assess whether the trajectory can be reasonably approximated as planar by performing principal component analysis (PCA) on its three-dimensional coordinates. We perform a

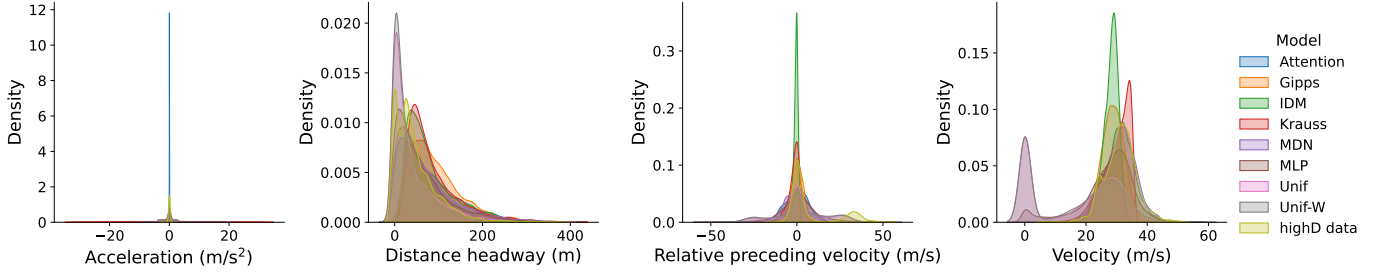


Fig. 2. Kernel density estimates of four driving feature distributions generated by each calibrated car-following model, overlaid with the corresponding distributions from highD data.

planarity test based on the variance explained by the three principal components. For each trajectory, PCA is applied independently to its three-dimensional coordinates, yielding three eigenvalues corresponding to the variance along PC1, PC2, and PC3. A planar trajectory is expected to exhibit substantially smaller variance along PC3 than along the first two principal axes.

We first compare the eigenvalues across the three principal axes using a repeated-measures ANOVA, where the trajectory index is treated as the subject and the principal axis (PC1, PC2, or PC3) is the within-subject factor. The ANOVA indicates a significant difference in variance among the three principal axes ($F(2, 1136) = 6628.24$, $p_{GG} < 0.001$). The Greenhouse-Geisser correction is applied because the sphericity assumption is violated ($W = 0.00163$, $p < 0.001$; $\epsilon = 0.5004$).

Following the significant omnibus test, we perform pairwise paired t -tests between the principal axes with Bonferroni correction. All three pairwise comparisons are statistically significant: PC1 has significantly greater variance than PC2 ($t(568) = 81.48$, $p_{corr} < 0.001$), PC1 has significantly greater variance than PC3 ($t(568) = 81.37$, $p_{corr} < 0.001$), and PC2 has significantly greater variance than PC3 ($t(568) = 23.80$, $p_{corr} < 0.001$). The corresponding effect sizes are also substantial, with Hedges' g values of 4.78, 4.82, and 1.36 for the PC1-PC2, PC1-PC3, and PC2-PC3 comparisons, respectively.

Most importantly, the variance along PC3 is substantially smaller than that along both PC1 and PC2. We therefore consider the trajectories to have a dominant two-dimensional structure and use the first two principal components to define the movement plane. Each trajectory is subsequently projected onto this plane before fitting the elliptical model. An ellipse is fitted to the projected trajectory using the direct least-squares method of Fitzgibbon et al. [15], yielding the conic parameters

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0. \quad (39)$$

The fitted ellipse is parameterised by its centre, orientation, and semi-axis lengths. For each demonstration, we additionally record its start and end points, trajectory length, the two principal axes defining the movement plane, and the three-dimensional offset of the plane. These quantities are then modelled statistically to capture variation across human demonstrations.

Specifically, trajectory length is modelled using a univariate Gaussian distribution, while the start and end points together

with the ellipse parameters are modelled using a multivariate Gaussian. A separate multivariate Gaussian is fitted to the two principal axes and the three-dimensional plane offset. During generation, these distributions are sampled to obtain the trajectory geometry, spatial orientation, and duration.

For each sampled trajectory, the start and end points are projected onto the sampled principal plane, and the shortest arc of the sampled ellipse connecting these points is generated. The resulting two-dimensional trajectory is then mapped back into three-dimensional space using the sampled principal axes and offset. To impose a smooth temporal profile, the spatial trajectory is subsequently reparameterised according to a minimum-jerk profile,

$$s(\tau) = 10\tau^3 - 15\tau^4 + 6\tau^5, \quad \tau \in [0, 1]. \quad (40)$$

Finally, a linear endpoint correction is applied to ensure that the generated trajectory exactly matches the sampled start and end points. The resulting trajectory therefore captures both the characteristic elliptical spatial geometry and the variability in movement duration, endpoints, orientation, and trajectory shape observed in the human demonstrations.

C. Dynamic Movement Primitive (DMP)

The Dynamic Movement Primitive (DMP) model represents a movement as a combination of a stable spring-damper system and a nonlinear forcing function [16]. This formulation allows the general shape of a demonstrated trajectory to be captured while retaining the ability to adapt the initial position, goal position, and movement duration. For each spatial dimension, the DMP is defined by the transformation system

$$\tau \dot{v} = K(g - y) - Dv + (g - y_0)f(s), \quad (41)$$

$$\tau \dot{y} = v, \quad (42)$$

where y denotes the position, v is a scaled velocity variable, y_0 and g are the initial and goal positions, respectively, and τ controls the movement duration. The parameters K and D determine the stiffness and damping of the underlying spring-damper system. The nonlinear forcing term $f(s)$ captures the characteristic shape of the demonstrated movement and is modulated by a canonical system

TABLE V
SUMMARY STATISTICS OF DRIVING FEATURES ACROSS DIFFERENT CAR-FOLLOWING MODELS AND HIGHD DATA.

Feature	Model	Mean	Variance	Std. Dev.	Median	Q25	Q75
Velocity	highD data	31.182	27.570	5.251	31.450	27.960	34.380
	Gipps	28.642	15.168	3.895	28.796	26.219	31.289
	Krauss	29.936	20.549	4.533	31.045	27.201	33.648
	IDM	28.125	6.866	2.620	28.508	26.713	29.885
	MLP	26.619	83.163	9.119	28.578	23.183	32.446
	MDN	31.333	26.904	5.187	31.536	27.971	34.650
	Attention	30.831	26.577	5.155	31.068	27.632	34.128
	Unif	16.793	191.191	13.827	21.382	0.154	28.787
Relative preceding velocity	Unif-W	16.535	188.677	13.736	20.586	0.153	28.420
	highD data	7.756	198.075	14.074	0.820	-0.390	6.920
	Gipps	-0.442	17.629	4.199	-0.040	-2.820	2.365
	Krauss	-1.220	14.869	3.856	-0.687	-3.127	1.084
	IDM	-0.743	3.246	1.802	-0.372	-1.331	0.289
	MLP	1.184	110.508	10.512	0.878	-4.536	6.406
	MDN	-0.526	49.642	7.046	-0.350	-5.170	3.931
	Attention	-0.095	51.873	7.202	0.309	-4.766	4.746
Distance headway	Unif	2.408	214.666	14.651	0.128	-2.787	9.677
	Unif-W	2.379	245.052	15.654	0.135	-4.641	11.979
	highD data	48.062	2688.000	51.846	33.720	14.370	68.340
	Gipps	95.033	3843.995	62.000	80.656	49.331	125.472
	Krauss	87.350	3615.240	60.127	68.247	45.099	111.160
	IDM	82.293	3503.209	59.188	63.721	39.493	107.245
	MLP	60.134	3855.602	62.093	41.090	14.938	85.792
	MDN	70.036	4541.234	67.389	49.689	20.350	98.957
Acceleration	Attention	76.388	4438.971	66.626	58.192	25.220	110.644
	Unif	41.064	2870.817	53.580	20.481	3.814	59.314
	Unif-W	35.615	2301.892	47.978	17.797	3.969	50.476
	highD data	0.001	0.104	0.323	0.020	-0.170	0.190
	Gipps	0.351	38.156	6.177	0.332	-3.763	4.480
	Krauss	0.620	145.241	12.052	0.701	-7.717	9.111
	IDM	0.216	4.443	2.108	0.210	-1.195	1.627
	MLP	-0.205	2.734	1.654	-0.049	-0.567	0.326
	MDN	0.142	3.470	1.863	0.138	-0.195	0.428
	Attention	0.051	0.003	0.056	0.044	0.024	0.077
	Unif	-0.497	4.075	2.019	-0.489	-2.244	1.247
	Unif-W	-0.502	4.075	2.019	-0.496	-2.251	1.244

TABLE VI
KL DIVERGENCE $D_{KL}(P_{\text{HIGHD}} \parallel Q_{\text{MODEL}})$ OF DRIVING FEATURE DISTRIBUTIONS BETWEEN HIGHD DATA AND EACH CAR-FOLLOWING MODEL (LOWER IS BETTER).

Model	Velocity	Relative preceding velocity	Distance headway	Acceleration
Gipps	0.024	0.510	0.214	0.148
Krauss	0.067	0.515	0.316	0.188
IDM	0.142	0.687	0.219	0.087
MLP	0.010	0.071	0.013	0.029
MDN	0.002	0.338	0.031	0.024
Attention	0.001	0.340	0.053	0.267
Unif	0.030	0.048	0.024	0.099
Unif-W	0.032	0.055	0.031	0.099

$$\tau \dot{s} = -\alpha_s s, \quad (43)$$

whose solution is

$$s(t) = \exp\left(-\alpha_s \frac{t}{\tau}\right). \quad (44)$$

The forcing function is represented using a set of N Gaussian basis functions:

$$f(s) = \frac{\sum_{i=1}^N w_i \psi_i(s)}{\sum_{i=1}^N \psi_i(s)}, \quad (45)$$

where w_i denotes the weight of the i -th basis function and

$$\psi_i(s) = \exp\left[-h_i(s - c_i)^2\right]. \quad (46)$$

In our implementation, the basis-function centres are exponentially distributed in the phase variable according to

$$c_i = \exp\left(-\alpha_s \frac{i-1}{N-1}\right), \quad (47)$$

with the basis-function widths determined by

$$h_i = \frac{N}{c_i^2}. \quad (48)$$

We use $N = 20$, $K = 100$, $D = 20$, and $\alpha_s = 4.6$. To fit the DMP to a human demonstration, the position, velocity, and acceleration are first obtained from the recorded trajectory using numerical differentiation. The target forcing function is then computed from the demonstrated trajectory by rearranging the DMP transformation system:

$$f_{\text{target}}(t) = \frac{\tau^2 \ddot{y}(t) - K(g - y(t)) + D\tau \dot{y}(t)}{g - y_0}. \quad (49)$$

The basis-function weights are subsequently estimated using locally weighted regression. This procedure is performed independently for each spatial dimension of the trajectory. Rather than using a single set of fitted weights, we model the variability across human demonstrations probabilistically. Specifically, a DMP is fitted independently to every human trajectory, and the resulting weight vectors are collected across demonstrations. A multivariate Gaussian distribution is then fitted to these weight vectors, with the empirical mean and covariance used as the parameters of the model:

$$\mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w). \quad (50)$$

During trajectory generation, the initial position, goal position, and movement duration are sampled from the corresponding distributions learned from the human training data. A DMP weight vector is independently sampled from the fitted multivariate Gaussian distribution and used in the transformation system to generate the trajectory. The system is then numerically integrated using Euler integration. This produces a stochastic DMP model that captures both the characteristic movement shape and variation observed in the human demonstrations.

D. MLP

The original giver trajectories are downsampled to 20 Hz, and a sliding window is applied to construct the state representation. Each state consists of five consecutive frames, from $T - 4$ to T , containing the standardised 3D coordinates of the giver's and taker's right hands, resulting in a state representation $\mathbf{s}_T \in \mathbb{R}^{5 \times 6}$, which is flattened to $\mathbb{R}^{30}$ for the MLP. The prediction target is the standardised 3D displacement of the giver's hand between T and $T + 1$. The MLP follows the same architecture as the model described in Section II-F, with only the input and output dimensions adjusted to the handover task. No other architectural modifications are introduced, and the model is trained using the same supervised learning procedure as the highway MLP model.

E. MDN

The MDN model used in the object handover domain follows the same architecture and training procedure as the highway MDN model described in Section II-G. The only architectural modification is the number of Gaussian mixture components in the output distribution, which is reduced from 15 to 5. Specifically, given the input trajectory representation, the network predicts the parameters of a five-component Gaussian mixture distribution over the next-step displacement

of the giver's hand. The reduction in the number of mixture components is intended to reflect the lower behavioural complexity of the object handover task. All other architectural and training settings are kept consistent with the corresponding highway MDN model.

F. Attention

The attention-based model used in the object handover domain is designed to capture temporal dependencies among the frames in the input trajectory window. Unlike the attention model used in the highway domain, which uses a more complex architecture, the handover model consists of a single multi-head self-attention layer followed by temporal pooling and a small MLP.

The input to the model is a sequence of five consecutive frames, with each frame represented by the 6-dimensional Cartesian coordinates of the giver's and taker's right hands. Thus, the input has the form

$$\mathbf{X} \in \mathbb{R}^{T \times 6}, \quad T = 5. \quad (51)$$

Each 6-dimensional frame is first projected into a 128-dimensional embedding space using a linear layer,

$$\mathbf{Z} = \mathbf{X}\mathbf{W}_e + \mathbf{b}_e, \quad \mathbf{Z} \in \mathbb{R}^{T \times 128}. \quad (52)$$

The resulting sequence is then processed by a multi-head self-attention layer with four attention heads. Query, key, and value are all obtained from the same projected sequence, allowing the model to capture dependencies between different time steps. The attention output is combined with the original embedded features through a residual connection and subsequently normalised using layer normalisation:

$$\mathbf{Z}' = \text{LayerNorm}(\mathbf{Z} + \text{MHA}(\mathbf{Z}, \mathbf{Z}, \mathbf{Z})). \quad (53)$$

The temporal dimension is then removed using global average pooling,

$$\mathbf{z}_{\text{pool}} = \frac{1}{T} \sum_{t=1}^T \mathbf{Z}'_t, \quad \mathbf{z}_{\text{pool}} \in \mathbb{R}^{128}. \quad (54)$$

The pooled representation is passed through a two-layer MLP consisting of a 128-unit linear layer, a ReLU activation, and a final linear layer. The final layer predicts the 3-dimensional displacement of the giver's hand at the next time step. The model is trained using the same supervised learning procedure as the other data-driven handover models. This architecture provides a lightweight mechanism for modelling temporal dependencies within the five-frame observation window while avoiding the additional complexity of deeper attention architectures.

G. Model Validation

To verify that the calibrated models reproduce the statistical characteristics of human handover behaviour, we compare the distributions of key trajectory features generated by each giver model against those observed in the human data. Figure 3 shows the corresponding kernel density estimates, while Table VII summarises the feature statistics and Table VIII

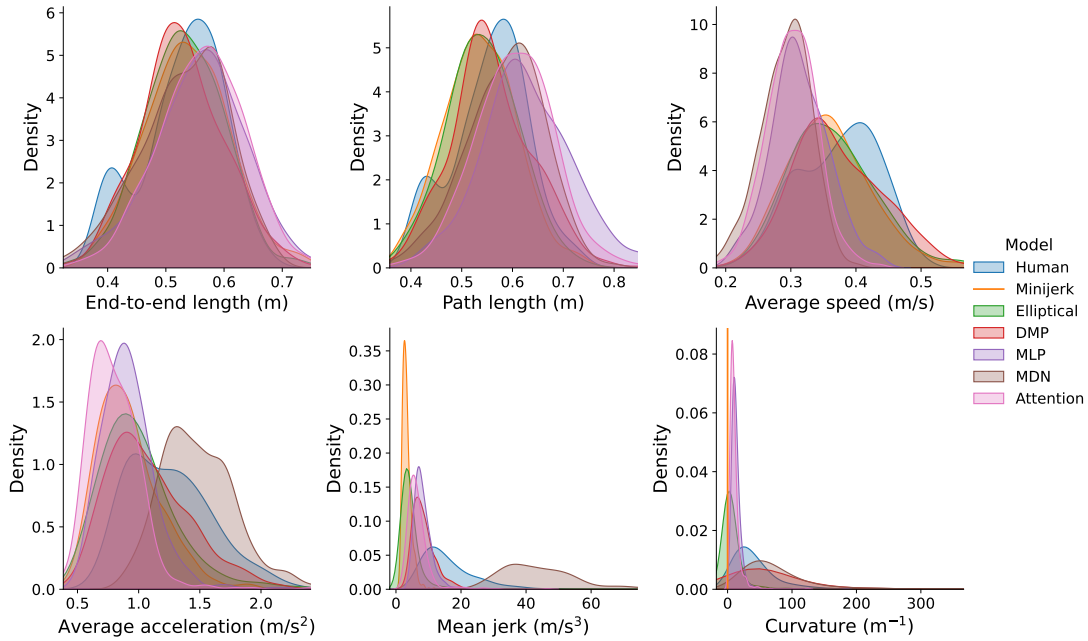


Fig. 3. Comparison of handover giver trajectory feature distributions between calibrated giver models and human data. Kernel density estimates are shown for end-to-end length, path length, average speed, average acceleration, movement smoothness, and trajectory curvature.

reports the corresponding KL divergences. The results show that the calibrated models capture several characteristics of the human trajectories, although their performance varies substantially across features. In particular, DMP achieves the lowest KL divergence for path length, average speed, and average acceleration, and performs consistently well across the remaining features. The data-driven models achieve the lowest divergence for some individual features, but generally show larger discrepancies in average speed and other trajectory characteristics. Overall, these results indicate that the calibrated models capture meaningful aspects of human handover behaviour while retaining distinct behavioural characteristics.

IV. TRAJECTORY GENERATION AND COLLECTION

We generate trajectories from each calibrated human behaviour model using a consistent simulation procedure within each domain. The generated trajectories are subsequently used for offline evaluation.

A. Highway Trajectory Generation

For the highway domain, the simulation scenario consists of a standard three-lane highway segment with a length of 420 m, matching the road-segment length of the highD dataset. At each simulation step, a new vehicle is spawned in each lane according to a spawn probability estimated from the three-lane recordings in the highD dataset, provided that a sufficiently large gap exists to ensure safe insertion.

For the rule-based driving models, each newly spawned vehicle is assigned model parameters sampled from the corresponding posterior distributions inferred from the human data. To initialise vehicle states with realistic values, we perform kernel density estimation (KDE) on the initial vehicle speeds

and vehicle lengths observed in the highD dataset. The initial speed and vehicle length of each newly spawned vehicle are then sampled from the corresponding KDE distributions.

For each driving model, simulations are run for 10,000 frames at 25 Hz, matching the sampling frequency of the highD dataset. A highway trajectory is defined as the time series of the 48-dimensional state representation described in Eq. 15 for an individual vehicle. We collect trajectories generated by each of the eight driving models in $\{H_i^{HW,train}\}$.

B. Handover Trajectory Generation

For the handover domain, trajectory generation follows the customised handover environment described in Section I-B. For each of the six giver models in $\{H_i^{HO,train}\}$, we run 500 episodes and collect one giver trajectory from each episode, resulting in 500 trajectories per model.

During trajectory generation, taker trajectories are sampled from the empirical human data and replayed as the taker's motion. Each generated handover trajectory therefore consists of the corresponding giver and taker trajectories, represented as a time series of their six-dimensional hand positions.

V. TRAJECTORY REPRESENTATIONS

Existing generative model evaluation metrics are typically computed in a feature space rather than directly on raw trajectories [17]. The choice of trajectory representation can therefore influence the resulting evaluation scores and, consequently, the assessment of human behaviour models. To investigate this effect, we consider three approaches for representing generated trajectories: PCA-based representations, learned latent representations obtained from an autoencoder, and manually extracted behavioural features. The first two

TABLE VII
SUMMARY STATISTICS OF HANDOVER GIVER TRAJECTORY FEATURES ACROSS GIVER MODELS AND HUMAN DATA.

Feature	Model	Mean	Variance	Std. Dev.	Median	Q ₂₅	Q ₇₅
End-to-end length	Human	0.529	0.005	0.071	0.541	0.490	0.581
	Minijerk	0.537	0.005	0.072	0.536	0.489	0.583
	Elliptical	0.531	0.005	0.069	0.529	0.485	0.579
	DMP	0.528	0.005	0.072	0.524	0.483	0.575
	MLP	0.556	0.007	0.082	0.561	0.506	0.608
	MDN	0.531	0.006	0.076	0.539	0.483	0.587
	Attention	0.560	0.006	0.076	0.564	0.511	0.614
Path length	Human	0.556	0.006	0.075	0.565	0.510	0.607
	Minijerk	0.537	0.005	0.072	0.536	0.489	0.583
	Elliptical	0.544	0.008	0.089	0.541	0.494	0.588
	DMP	0.559	0.006	0.080	0.550	0.510	0.609
	MLP	0.628	0.008	0.089	0.622	0.571	0.686
	MDN	0.584	0.006	0.079	0.594	0.536	0.638
	Attention	0.604	0.006	0.079	0.604	0.552	0.656
Average speed	Human	0.373	0.004	0.062	0.378	0.322	0.420
	Minijerk	0.360	0.004	0.067	0.355	0.313	0.401
	Elliptical	0.365	0.006	0.075	0.359	0.317	0.404
	DMP	0.372	0.005	0.068	0.361	0.322	0.418
	MLP	0.314	0.002	0.045	0.311	0.285	0.343
	MDN	0.292	0.002	0.039	0.297	0.268	0.319
	Attention	0.302	0.002	0.039	0.302	0.276	0.328
Average acceleration	Human	1.275	0.177	0.421	1.229	0.968	1.475
	Minijerk	0.926	0.075	0.275	0.882	0.726	1.054
	Elliptical	1.009	0.158	0.398	0.944	0.782	1.139
	DMP	1.085	0.117	0.342	1.017	0.835	1.296
	MLP	0.914	0.050	0.224	0.893	0.767	1.028
	MDN	1.489	0.093	0.306	1.465	1.269	1.690
	Attention	0.799	0.046	0.214	0.776	0.652	0.922
Movement smoothness	Human	16.448	108.472	10.415	13.789	10.281	19.494
	Minijerk	3.303	1.846	1.359	3.030	2.353	3.867
	Elliptical	4.638	30.531	5.525	3.508	2.662	4.874
	DMP	8.493	11.857	3.443	7.702	6.013	10.073
	MLP	8.307	8.120	2.850	7.708	6.444	9.442
	MDN	43.700	127.901	11.309	42.147	35.351	50.413
	Attention	6.763	16.035	4.004	5.904	4.659	7.671
Trajectory curvature	Human	45.331	3242.138	56.940	31.024	18.426	52.665
	Minijerk	0.000	0.000	0.000	0.000	0.000	0.000
	Elliptical	10.088	1273.335	35.684	1.458	0.445	5.266
	DMP	70.640	2.53e+04	159.159	44.277	29.865	69.188
	MLP	13.333	103.234	10.160	11.377	8.553	15.579
	MDN	80.261	7008.926	83.719	59.767	39.334	88.505
	Attention	10.602	80.425	8.968	8.348	5.946	12.159

representations are used to compute distributional evaluation metrics, while the manually extracted features are used to compare the corresponding feature distributions between human and generated behaviours.

A. PCA Representation

For the PCA-based representation, Principal Component Analysis (PCA) is applied to the collected human trajectories, and the resulting principal axes are used to project both human and generated trajectories into a lower-dimensional feature space. As the reference configuration, we retain the first six principal components for the highway domain and the first four principal components for the handover domain.

B. Latent Representation

Generative model evaluation metrics are commonly computed using representations extracted by a deep neural network trained on an auxiliary task using real data [17]. Since

metrics such as FID were originally developed for image generation, this auxiliary task is often image classification. However, the highD dataset does not naturally provide class labels. Therefore, rather than training a classifier, we adopt an autoencoder framework and learn latent representations through state sequence reconstruction.

Specifically, the trajectory of each identified vehicle in the highD dataset is segmented into multiple sequences of duration T seconds. The vehicle states are standardised before segmentation. As the highD data are downsampled to 5 Hz, each sequence contains $5T$ consecutive samples. We set $T = 5$ seconds in this study, resulting in sequences of 25 time steps.

The architecture of the autoencoder is illustrated in Figure 4. Given an input state sequence $X \in \mathbb{R}^{5T \times d}$, where $T = 5$ and $d = 48$ for the highway domain, the sequence is first processed by a bidirectional LSTM (BiLSTM) encoder, which has previously been used for driving behaviour modelling [18]. The encoder produces the final hidden states from the forward and

TABLE VIII
KL DIVERGENCE $D_{KL}(P_{\text{HUMAN}} \parallel Q_{\text{MODEL}})$ OF HANDOVER GIVER TRAJECTORY FEATURE DISTRIBUTIONS BETWEEN HUMAN DATA AND EACH GIVER MODEL (LOWER IS BETTER).

Model	End-to-end length	Path length	Average speed	Average acceleration	Movement smoothness	Trajectory curvature
Minijerk	0.054	0.064	0.084	0.696	11.743	1.602
Elliptical	0.060	0.060	0.069	0.399	2.298	1.109
DMP	0.067	0.057	0.068	0.250	2.299	0.288
MLP	0.118	0.336	1.059	0.951	2.397	4.083
MDN	0.037	0.093	4.088	0.644	7.449	0.270
Attention	0.140	0.211	1.824	1.619	1.871	3.770

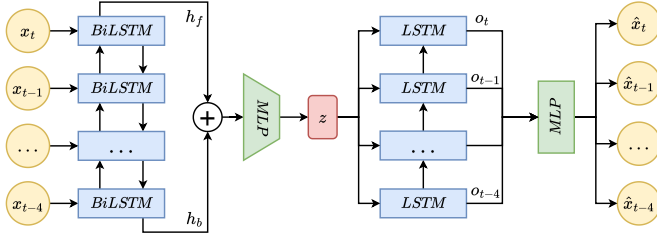


Fig. 4. Architecture of the autoencoder used for latent feature extraction. A BiLSTM encoder maps the input state sequence to a latent vector z , which is decoded by a repeated-vector LSTM and an MLP to reconstruct the original sequence.

backward passes, denoted by h_f and h_b , respectively. These hidden states are concatenated to form a joint representation, which is subsequently processed by an MLP to produce the latent vector z .

During training, the latent vector z is decoded using a repeated-vector LSTM [19]. Specifically, z is replicated across the temporal dimension and provided as input to the LSTM at each time step. The resulting sequence of hidden states is then processed by a second MLP to reconstruct the original state sequence, denoted by $\hat{X} \in \mathbb{R}^{5T \times d}$. The autoencoder is trained by minimising the mean squared reconstruction error,

$$\mathcal{L}_{\text{AE}} = \frac{1}{5Td} \sum_{t=1}^{5T} \|\mathbf{x}_t - \hat{\mathbf{x}}_t\|_2^2. \quad (55)$$

For the reference configuration, the hidden dimensions of both the BiLSTM encoder and repeated-vector LSTM decoder are set to 128. The encoder and decoder MLPs each contain a single hidden layer with 128 units, and the latent dimension is set to 32. The autoencoder is trained using the human trajectories from the highD dataset. The trained encoder is subsequently used to extract latent features z from both the human trajectories and trajectories generated by the human behaviour models. The generated trajectories are downsampled from 25 Hz to 5 Hz to match the input resolution used to train the autoencoder. These latent representations are then used to compute the model evaluation metrics.

For the handover domain, we use the same general autoencoder architecture, with the input adapted to the handover trajectory representation. Specifically, each LSTM node receives a single frame containing the 6-dimensional state of the giver and taker. The autoencoder is trained using the handover@RPL dataset and is subsequently used to extract

latent representations from both the human trajectories and the trajectories generated by each giver model.

C. Manually Extracted Behavioural Features

In the highway driving domain, we consider speed, acceleration, distance headway, and relative preceding speed. In the object handover domain, we consider acceleration, speed, movement smoothness, curvature, trajectory end-to-end length, and path length. For each feature, we estimate the distributions of the human and generated data using kernel density estimation (KDE) and compare them using KL divergence. As the reference configuration, we use a KDE sample size of 10,000 for the highway domain. For the handover domain, we use all available samples, as the trajectories are relatively short and the resulting computation is efficient.

VI. EVALUATION METRICS

A. Fréchet Distance (FD)

Fréchet Distance is a widely used metric for evaluating generative models, originally proposed for image generation tasks [20] as the *Fréchet Inception Distance (FID)*. It has since been adapted to other domains, including gesture and video generation [21], [22]. It measures the similarity between real and generated images by comparing feature representations extracted from a pretrained convolutional neural network. In practice, *FD* models the extracted features of real and generated samples as multivariate Gaussian distributions and computes the Fréchet distance [23] between these. A lower *FD* score indicates that the generated data distribution is closer to the real data distribution, suggesting better quality and diversity of the generated samples.

Assuming we have the generated data distribution and the real data distribution as two multivariate Gaussian distributions $P_{\text{gen}} \sim \mathcal{N}(\mu_{\text{gen}}, \Sigma_{\text{gen}})$ and $P_{\text{real}} \sim \mathcal{N}(\mu_{\text{real}}, \Sigma_{\text{real}})$, the *FD* between these is defined as

$$\text{FD} = \|\mu_{\text{gen}} - \mu_{\text{real}}\|_2^2 + \text{Tr}\left(\Sigma_{\text{gen}} + \Sigma_{\text{real}} - 2(\Sigma_{\text{gen}}\Sigma_{\text{real}})^{\frac{1}{2}}\right), \quad (56)$$

where $\|\cdot\|_2$ denotes the Euclidean norm and $\text{Tr}(\cdot)$ denotes the trace operator. A lower *FD* value indicates that the generated distribution is closer to the real distribution in terms of both mean and covariance, implying better generation quality.

Note that if the generated data perfectly matches the real data distribution, the *FD* would be zero. In practice, however, achieving an *FD* of zero is neither realistic nor desired. Therefore, *FD* should be interpreted as a relative metric for

comparing generative models. Specifically, given two models with $FD_{\text{gen1}} > 0$ and $FD_{\text{gen2}} > 0$, if

$$FD_{\text{gen1}} > FD_{\text{gen2}},$$

then generative model 2 is considered to produce samples that are more faithful to the real data distribution than generative model 1 [17].

B. Density

The *density* metric, an improved version of *precision* [24] introduced together with coverage [25], is used to evaluate generative models by assessing how well generated samples represent the real data manifold. *Density* focuses on the quality of generated samples in regions where real data exist.

Specifically, *density* quantifies how many generated samples fall within a local neighbourhood of real data points in the feature space. A higher *density* score indicates that generated samples are well aligned with the real data distribution and are not scattered in unrealistic regions.

Let $\mathcal{X}_{\text{real}} = \{x_i\}_{i=1}^N$ denote the set of real data samples and $\mathcal{X}_{\text{gen}} = \{\hat{x}_j\}_{j=1}^M$ denote the set of generated samples. For each real sample x_i , let $B(x_i, \text{NND}_k(x_i))$ be the hypersphere centred at x_i with radius $\text{NND}_k(x_i)$, defined as the distance from x_i to its k -th nearest neighbour. The *density* metric is then defined as

$$\text{density} = \frac{1}{kM} \sum_{j=1}^M \sum_{i=1}^N \mathbb{1}_{\hat{x}_j \in B(x_i, \text{NND}_k(x_i))}. \quad (57)$$

Higher values of *density* indicate that the generated samples closely cover the regions of the real data manifold. Note that *density* can exceed 1 if most generated samples fall within the hyperspheres of more than k real samples.

C. Coverage

The *coverage* metric [25] is an improved version of *recall* in [24]. It measures the fraction of real data points that are close to at least one generated sample in the feature space, indicating how well the generative model captures the support of the real data distribution. Formally, for a set of real samples $\mathcal{X}_{\text{real}} = \{x_i\}_{i=1}^N$ and generated samples $\mathcal{X}_{\text{gen}} = \{\hat{x}_j\}_{j=1}^M$, coverage is defined as

$$\text{coverage} = \frac{1}{N} \sum_{i=1}^N \mathbb{1}_{\exists j \text{ s.t. } \hat{x}_j \in B(x_i, \text{NND}_k(x_i))}, \quad (58)$$

where $B(x_i, \text{NND}_k(x_i))$ is defined the same as in 57. Higher coverage values indicate that the generated samples adequately cover the real data distribution.

D. Average Pair Distance (APD)

The *average pair distance* (APD) represents a general metric for measuring diversity. This metric has been applied in many existing studies, for example in image generation [26] and gesture generation [27].

Given a dataset, a random selection procedure is repeated R times. In each repetition, S pairs of samples are randomly

chosen, and the Euclidean distance is computed for each pair. The APD is then obtained by averaging these distances across all repetitions. Formally, for a dataset $\mathcal{X}$, APD is defined as

$$\text{APD}(\mathcal{X}) = \frac{1}{RS} \sum_{r=1}^R \sum_{i=1}^S \|x_i - \tilde{x}_i\|_2, \quad (59)$$

where x_i and $\tilde{x}_i$ denote the elements of the i -th pair of samples randomly drawn from $\mathcal{X}$.

E. Mean Maximum Similarity (MMS)

MMS was proposed in [28] as a novelty metric to complement diversity measures such as coverage, which can be misleading when generated samples perfectly replicate real samples, resulting in zero novelty. Formally, the MMS between generated samples $\mathcal{X}_{\text{gen}} = \{\hat{x}_j\}_{j=1}^M$ and real samples $\mathcal{X}_{\text{real}} = \{x_i\}_{i=1}^N$ is defined as

$$\text{MMS}(\mathcal{X}_{\text{gen}}, \mathcal{X}_{\text{real}}) = \frac{1}{M} \sum_{j=1}^M \min_{i \in \mathcal{X}_{\text{real}}} \|\hat{x}_j - x_i\|_2. \quad (60)$$

In [28], the novelty of generated samples is quantified as the gap between $\text{MMS}(\mathcal{X}_{\text{gen}}, \mathcal{X}_{\text{real}})$ and $\text{MMS}(\mathcal{X}_{\text{real}}, \mathcal{X}_{\text{real}})$, where the latter is computed on the real dataset itself. In this case, the Euclidean distance excludes self-comparisons when identifying the nearest neighbour.

F. KL Divergence

For a manually extracted feature f , let $p_{\text{H}}^{(f)}(x)$ and $p_{\text{G}}^{(f)}(x)$ denote the probability density functions estimated from the human and generated data, respectively. The KL divergence is computed as

$$D_{\text{KL}}^{(f)}(p_{\text{H}}^{(f)} \| p_{\text{G}}^{(f)}) = \int p_{\text{H}}^{(f)}(x) \log \frac{p_{\text{H}}^{(f)}(x)}{p_{\text{G}}^{(f)}(x)} dx. \quad (61)$$

Given F manually extracted features, the average KL divergence is defined as

$$\overline{D}_{\text{KL}} = \frac{1}{F} \sum_{f=1}^F D_{\text{KL}}^{(f)}. \quad (62)$$

VII. REINFORCEMENT LEARNING DETAILS

A. Highway Driving

We train the highway driving policies using the Deep Q-Network (DQN) algorithm implemented in Stable-Baselines3. Each policy is trained in the `highway-fast-v0` environment, with the behaviour of the surrounding vehicles determined by one of the human behaviour models described in the main paper. The same RL algorithm, policy architecture, and hyperparameters are used across all training configurations to ensure that differences in downstream performance arise from the human behaviour model used during training.

The DQN agent uses an MLP policy with two hidden layers of 256 units each. The learning rate is set to 5×10^{-4} , with a replay buffer size of 15,000 transitions and a batch size of 32. Learning begins after 200 environment steps. The discount factor is $\gamma = 0.8$, and one gradient step is performed after

every environment step. The target network is updated every 50 training steps. All models are trained on the CPU.

For each human behaviour model, training is repeated using three random seeds. The random seed is applied consistently to the RL training procedure to control stochasticity across repeated training runs. Each training run is performed for 40,000 environment steps. The resulting policies are saved and subsequently evaluated against all human behaviour models in the corresponding domain, as described in the main paper.

B. Object Handover

We train the object handover policies using the Soft Actor-Critic (SAC) algorithm implemented in Stable-Baselines3. Each policy is trained in the custom `HandoverSim` environment, with the behaviour of the giver determined by one of the human behaviour models described in the main paper. As in the highway domain, the same RL training configuration is used across all giver models.

The SAC agent uses an MLP policy and is trained using 16 parallel environments. The initial learning rate is 10^{-3} and is gradually reduced to 5×10^{-4} using a sigmoid learning-rate schedule over the course of training. The replay buffer contains up to 50,000 transitions, and learning begins after 1,000 environment steps. The batch size is 128, the discount factor is $\gamma = 0.99$, and the agent performs eight environment steps between gradient updates and eight gradient steps per update. All models are trained on the CPU.

Each training configuration is repeated using three random seeds, with explicit seeding applied to the RL algorithm and environment. Each policy is trained for 100,000 environment steps. During training, episode reward, episode duration, and task success are recorded for monitoring. The resulting policies are subsequently evaluated against each of the six testing giver models for 20 episodes per giver model. During evaluation, the trained policies act deterministically, and the mean episode reward, mean episode duration, and success rate are recorded.

For both domains, the RL training procedure is kept fixed across human behaviour models within the domain. Consequently, the human behaviour model used during training is the primary experimental factor, allowing the downstream effects of different human behaviour models to be compared under a consistent RL training procedure.

REFERENCES

- [1] R. Krajewski, J. Bock, L. Kloecker, and L. Eckstein, "The highD Dataset: A Drone Dataset of Naturalistic Vehicle Trajectories on German Highways for Validation of Highly Automated Driving Systems," in *ITSC*, 2018, pp. 2118–2125.
- [2] E. Leurent, "An environment for autonomous driving decision-making," <https://github.com/eleurent/highway-env>, 2018.
- [3] P. Khanna, M. Björkman, and C. Smith, "A Multimodal Data Set of Human Handovers with Design Implications for Human-Robot Handovers," in *RO-MAN*, Aug. 2023, pp. 1843–1850.
- [4] A. Kesting, M. Treiber, and D. Helbing, "General Lane-Changing Model MOBIL for Car-Following Models," *Transp. Res. Rec.*, vol. 1999, no. 1, pp. 86–94, 2007. [Online]. Available: <https://doi.org/10.3141/1999-10>
- [5] P. Gipps, "A behavioural car-following model for computer simulation," *Transp. Res. Part B Methodol.*, vol. 15, no. 2, pp. 105–111, 1981. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/0191261581900370>
- [6] A. Kucukelbir, D. Tran, R. Ranganath, A. Gelman, and D. M. Blei, "Automatic differentiation variational inference," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 430–474, Jan. 2017.
- [7] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic Traffic Simulation using SUMO," in *ITSC*. IEEE, 2018. [Online]. Available: <https://elib.dlr.de/124092/>
- [8] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Phys. Rev. E*, vol. 62, pp. 1805–1824, Aug. 2000. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevE.62.1805>
- [9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., Dec. 2017, pp. 6000–6010.
- [10] K. Sohn, H. Lee, and X. Yan, "Learning Structured Output Representation using Deep Conditional Generative Models," in *NeurIPS*, vol. 28. Curran Associates, Inc., 2015.
- [11] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data," in *ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham: Springer International Publishing, 2020, pp. 683–700.
- [12] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.
- [13] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," in *EMNLP*. Doha, Qatar: Association for Computational Linguistics, 2014, pp. 1724–1734.
- [14] T. Flash and N. Hogan, "The coordination of arm movements: An experimentally confirmed mathematical model," *J. Neurosci.*, vol. 5, no. 7, pp. 1688–1703, Jul. 1985.
- [15] A. Fitzgibbon, M. Pilu, and R. Fisher, "Direct least square fitting of ellipses," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 5, pp. 476–480, May 1999.
- [16] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors," *Neural Comput.*, vol. 25, no. 2, pp. 328–373, Feb. 2013.
- [17] A. Ismail-Fawaz, M. Devanne, S. Berretti, J. Weber, and G. Forestier, "Establishing a unified evaluation framework for human motion generation: A comparative analysis of metrics," *Comput. Vis. Image Underst.*, vol. 254, p. 104337, 2025. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1077314225000608>
- [18] G. Sun, H. Zhang, L. Zhong, and Q. Li, "Identification of driving behavior in continuous diverging sections of expressway system interchange based on CNN-BiLSTM," *Sci. Rep.*, vol. 15, no. 1, p. 10631, Mar. 2025.
- [19] S. Ghimire, R. C. Deo, H. Wang, M. S. Al-Musaylh, D. Casillas-Pérez, and S. Salcedo-Sanz, "Stacked LSTM Sequence-to-Sequence Autoencoder with Feature Selection for Daily Solar Radiation Prediction: A Review and New Modeling Results," *Energies*, vol. 15, no. 3, Jan. 2022.
- [20] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local nash equilibrium," in *NeurIPS*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 6629–6640.
- [21] A. Maiorica, H. Bohy, Y. Yoon, and T. Dutoit, "Objective Evaluation Metric for Motion Generative Models: Validating Fréchet Motion Distance on Foot Skating and Over-smoothing Artifacts," in *ACM SIGGRAPH MIG*, ser. MIG '23. New York, NY, USA: Association for Computing Machinery, Nov. 2023, pp. 1–11.
- [22] B. Wang, N. Sridhar, C. Feng, M. van der Merwe, A. Fishman, N. Fazeli, and J. J. Park, "This&That: Language-Gesture Controlled Video Generation for Robot Planning," in *ICRA*, May 2025, pp. 12 842–12 849.
- [23] M. M. Fréchet, "Sur quelques points du calcul fonctionnel," *Rend. Circ. Mat. Palermo*, vol. 22, no. 1, pp. 1–72, Dec. 1906.
- [24] M. S. M. Sajjadi, O. Bachem, M. Lučić, O. Bousquet, and S. Gelly, "Assessing Generative Models via Precision and Recall," in *NeurIPS*, 2018.
- [25] M. F. Naeem, S. J. Oh, Y. Uh, Y. Choi, and J. Yoo, "Reliable fidelity and diversity metrics for generative models," in *ICML*, ser. ICML'20. JMLR.org, 2020.
- [26] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric," in *CVPR*, Jun. 2018, pp. 586–595.
- [27] C. Guo, X. Zuo, S. Wang, S. Zou, Q. Sun, A. Deng, M. Gong, and L. Cheng, "Action2Motion: Conditioned Generation of 3D Human

- Motions,” in *ACM MM*, ser. MM ’20. New York, NY, USA: Association for Computing Machinery, Oct. 2020, pp. 2021–2029.
- [28] P. Cervantes, Y. Sekikawa, I. Sato, and K. Shinoda, “Implicit Neural Representations for Variable Length Human Motion Generation,” in *ECCV 2022*, S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, Eds. Cham: Springer Nature Switzerland, 2022, pp. 356–372.