Exploring multimodal implicit behavior learning for vehicle navigation in simulated cities

Eric Aislan Antonelo[1], **Gustavo Claudio Karl Couto**[1],**Christian Möller**[2]

[1] Automation and Systems Engineering Department,
Federal University of Santa Catarina, Florianopolis, Brazil

[2]Faculty of Science and Engineering, Information Technology
Åbo Akademi University, Finland

eric.antonelo@ufsc.br, gustavo.karl.couto@posgrad.ufsc.br

christian.moller@abo.fi

Abstract. *Standard Behavior Cloning (BC) fails to learn multimodal driving decisions, where multiple valid actions exist for the same scenario. We explore Implicit Behavioral Cloning (IBC) with Energy-Based Models (EBMs) to better capture this multimodality. We propose Data-Augmented IBC (DA-IBC), which improves learning by perturbing expert actions to form the counterexamples of IBC training and using better initialization for derivative-free inference. Experiments in the CARLA simulator with Bird's-Eye View inputs demonstrate that DA-IBC outperforms standard IBC in urban driving tasks designed to evaluate multimodal behavior learning in a test environment. The learned energy landscapes are able to represent multimodal action distributions, which BC fails to achieve.*

1. Introduction

Many learning-based approaches for autonomous driving rely on Behavior Cloning (BC) [Pomerleau 1991], a supervised method that learns from offline expert demonstrations [Bojarski et al. 2016, Xu et al. 2017, Bansal et al. 2018, Codevilla et al. 2018, Codevilla et al. 2019]. In BC, a human driver provides input observations paired with corresponding control commands. For example, Codevilla [Codevilla et al. 2018, Codevilla et al. 2019] applied BC for autonomous driving in the CARLA simulator. A large dataset of human driving data was collected and augmented using image processing techniques to train end-to-end policies conditioned on the desired route.

In autonomous driving, multimodality describes the presence of multiple valid control choices or trajectories available to a driver under the same circumstances. For example, at an intersection, a vehicle might continue forward, make a left turn, or turn right, depending on elements like surrounding traffic and pedestrian activity. Similarly, when overtaking, a driver can opt for different acceleration profiles or merge timings based on the current traffic flow [Bansal et al. 2018]. Braking behavior also varies—some drivers may begin slowing down smoothly and early, while others may delay braking and decelerate more abruptly. Traditional BC can not capture such variability, as it typically learns a deterministic policy that maps each observation $\mathbf{o}$ to a single action $\hat{\mathbf{a}}$, as in $\hat{\mathbf{a}} = F_\theta(\mathbf{o})$. When the demonstration data includes multiple valid behaviors for the same input, BC averages these actions, resulting in mode collapse, a failure mode where the

learned policy does not reflect the diversity of real-world driving decisions. For instance, if demonstrations include both left and right turns in similar contexts, the model may output a steering command between the two, causing the vehicle to drive straight and potentially fail. In this context, a unimodal policy, such as the one learned by BC, captures only one possible behavior, while a multimodal policy can represent multiple valid actions for the same sensory input.

Energy-Based Models (EBMs) offer a compelling alternative for capturing the inherently multimodal nature of driving behavior. Rather than directly predicting an action, EBMs learn an energy landscape over the action space, where desirable actions are associated with lower energy values and less favorable ones with higher energy. This formulation naturally accommodates multiple plausible actions by assigning low energy to all feasible choices, avoiding the need to commit to a single output. As a result, EBMs can represent diverse, human-like driving responses without collapsing to an average of conflicting behaviors. In contrast to BC, which minimizes a supervised loss on expert actions, EBMs are trained using a contrastive approach, encouraging low energy for expert demonstrations and high energy for non-expert samples. This learning strategy enhances generalization and helps the model reject unrealistic actions during inference.

In this work, we follow the implicit behavioral cloning approach [Florence et al. 2022], adopting a reformulation of BC using implicit models: $\hat{\mathbf{y}} = \arg\min_{\mathbf{y} \in Y} E_\theta(\mathbf{x}, \mathbf{y})$ instead of $\hat{\mathbf{y}} = F_\theta(\mathbf{x})$. This formulation casts imitation as a conditional energy-based modeling problem and, at inference time (given $\mathbf{x}$), performs implicit regression by optimizing the best action $\hat{\mathbf{y}}$ through sampling or gradient-based optimization [Du and Mordatch 2019]. Findings from [Florence et al. 2022] demonstrate that implicit models for BC can learn long-horizon, closed-loop visuomotor tasks more effectively than their explicit counterparts. Their success is partially attributed to their capacity to represent not only multimodal distributions but also discontinuous functions. While implicit models for BC have been applied to robotic behavior learning [Florence et al. 2022], in this work, we propose their use in scenarios involving multimodal action selection for autonomous vehicles navigating urban environments, specifically leveraging the high-fidelity autonomous driving simulator CARLA [Dosovitskiy et al. 2017].

Energy-based approaches have been explored to capture multimodality in sequential decision-making. In [Pang et al. 2021], pedestrian motion is modeled by combining latent variables with energy functions to generate diverse future trajectories. Although not targeting vehicle control, it demonstrates how EBMs can handle multimodal forecasting in interactive environments. Deep Imitative Models [Rhinehart et al. 2018] learn a distribution over expert trajectories and perform inference by optimizing full paths, but rely on maximum likelihood training and trajectory-level reasoning and not on EBMs. The work in [Balesni et al. 2023] applies EBMs to control steering in a road following task, but lacks generalization to complex urban settings. In contrast, our proposed IBC method applies EBMs directly to action-level inference in urban driving.

However, conventional IBC has some known limitations when applied to many problems [Singh et al. 2023]. To improve IBC, we propose a new method for generating counterexamples, namely Data-Augmented IBC, which draws action samples from the expert set and adds a perturbation to each one to form the set of counterexamples,

making learning in IBC more effective. This makes the negative sampler better since it generates more realistic counterfactual actions than randomly sampling over the complete action space as done in standard IBC. We evaluate this new method and show its effective multimodal behavior learning in an autonomous driving setting where the route to follow is unknown to the agent. Our main contributions are: 1) we improve IBC training through expert-based counterexample sampling; 2) more effective inference initialization and sampling; 3) we apply the proposed method in the realistic CARLA simulator for urban driving, showing that the energy function learns to model different actions in the same situations.

2. Methods

2.1. Bird's-Eye View - BEV representation

The bird's-eye view of a vehicle represents its position and movement in a top-down coordinate system [Bansal et al. 2018]. The vehicle's location, heading, and speed are represented by p_t, θ_t, and s_t respectively. The top-down view is defined so that the agent's starting position is always at a fixed point within an image (the center of it). Furthermore, it is represented by a multi-channel image, where each channel is a binary map indicating the presence or absence of a specific semantic class (e.g., road, vehicle, pedestrian), with values restricted to 0 or 1, and rendered at a ground sampling resolution of ϕ meters per pixel. The BEV of the environment moves as the vehicle moves, allowing the agent to see a fixed range of meters in front of it. For instance, the BEV representation for a vehicle located just before a traffic light is given in Fig. 1(c), where drivable area and lane boundaries are shown as two BEV channels. Besides, other three BEV channels for the traffic lights representing the state of the traffic lights (green or red) at different timesteps are shown in Fig. 1(d). Fig. 1(b) shows the integrated five-channel BEV image, with each channel rendered with a different color. The mid-level BEV input can be learned as in [Couto and Antonelo 2024], but in this work we use the ground-truth BEV generated using the CARLA simulator.

2.2. Behavior Cloning (BC)

Behavior cloning is a supervised imitation learning method that trains a policy to mimic expert behavior. In our work, a CNN learns to output two continuous actions—acceleration and steering—from a Bird's-Eye View (BEV) representation and additional state variables.

Let $\pi_\theta(s)$ denote the policy mapping state x to action y, and $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ be the expert dataset. Rather than using a standard MSE or cross-entropy loss, we model a stochastic policy with a Beta distribution, which naturally handles variables bounded in $[-1, 1]$. For each sample i, with actions $y_i = (y_i^{\text{acc}}, y_i^{\text{steer}})$ and policy outputs $\pi_\theta(x_i) = (\alpha_i^{\text{acc}}, \beta_i^{\text{acc}}, \alpha_i^{\text{steer}}, \beta_i^{\text{steer}})$, we define the Beta density as:

$$f(z; \alpha, \beta) = \frac{z^{\alpha-1}(1-z)^{\beta-1}}{B(\alpha, \beta)},$$

with $B(\alpha, \beta)$ as the normalization constant. Actions are rescaled from $[-1, 1]$ to $[0, 1]$ via $y_i' = \frac{y_i+1}{2}$. The negative log-likelihood loss is then:

$$\mathcal{L}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\Big[\log f\left(\frac{y_i^{\text{acc}}+1}{2}; \alpha_i^{\text{acc}}, \beta_i^{\text{acc}}\right) + \log f\left(\frac{y_i^{\text{steer}}+1}{2}; \alpha_i^{\text{steer}}, \beta_i^{\text{steer}}\right)\Big]. \quad (1)$$

(a) Frontal Cameras

(b) five-channel BEV image

(c) drivable area, and lane boundaries (d) BEV's traffic lights channels at timesteps (-1, -9, -16)

Figure 1. a) Images from three frontal cameras positioned on the left, center, and right side of the vehicle. These images were taken after the initial interactions of the agent in the CARLA simulation environment. The corresponding bird's-eye view channels used as inputs to our agent are shown next: b) The final BEV image displays all five channels combined in different colors; c) Two BEV 192×192 channels that represent, from left to right, the drivable area, and the lane boundaries; d) BEV's traffic lights channels at timesteps (-1, -9, -16), i.e., representing the past states of the traffic light channel (grey color is red light off; white color means red ligh on).

This approach lets the model capture uncertainty by learning the distribution parameters instead of fixed action values.

2.3. Implicit Behavior Cloning (IBC)

As defined in [Florence et al. 2022], we consider an implicit model to be any composition $(\arg\min_y \circ E_\theta(x, y))$ in which inference is performed using a general-purpose function approximator $(E : \mathbb{R}^{m+n} \to \mathbb{R}^1)$ to solve the optimization problem $[\hat{y} = \arg\min_y E_\theta(x, y)]$. We leverage techniques from the energy-based model (EBM) literature to train such a model. Given a dataset of samples $(\{x_i, y_i\})$ and regression bounds $(y_{\min}, y_{\max} \in \mathbb{R}^m)$, training consists of generating a set of negative counterexamples $\{\tilde{y}_i^j\}_{j=1}^{N_{\text{neg}}}$ for each sample (x_i) in a batch and employing an InfoNCE-style loss function [Oord et al. 2018]. This loss corresponds to the negative log-likelihood of $[p_\theta(y|x) = \frac{\exp(-E_\theta(x,y))}{Z(x,\theta)}]$, where the counterexamples are used to estimate $Z(x_i, \theta)$:

$$\mathcal{L}_{\text{InfoNCE}}(\theta) = \sum_{i=1}^{N} -\log\left(\tilde{p}_\theta(y_i \mid x_i, \{\tilde{y}_i^j\}_{j=1}^{N_{\text{neg}}})\right), \tag{2}$$

$$\tilde{p}_\theta(y_i \mid x_i, \{\tilde{y}_i^j\}_{j=1}^{N_{\text{neg}}}) = \frac{e^{-E_\theta(x_i,y_i)}}{e^{-E_\theta(x_i,y_i)} + \sum_{j=1}^{N_{\text{neg}}} e^{-E_\theta(x_i,\tilde{y}_i^j)}}. \tag{3}$$

Thus, the lower the energy $E_\theta(x, y)$, the higher the probability of the action y, conditioned on the observation x. With a trained energy model $E_\theta(x, y)$, implicit inference can be performed via stochastic optimization to solve $\hat{y} = \arg\min_y E_\theta(x, y)$ (see Section 2.4.1).

Moreover, this implicit model allows for the representation of probability distributions over actions with multiple modes, meaning that more than one action may be appropriate for a given context (for the same input image frame). In contrast, conventional (explicit) BC does not support this, as it models only unimodal distributions (such as a Gaussian distribution). If such a model were placed in a scenario requiring multimodal action selection, the policy would likely choose an action that represents the mean of all modes. This could lead a vehicle to drive straight ahead when it should turn left or right, resulting in an invalid or unsafe action.

2.4. Data-Augmented IBC (DA-IBC)

In [Florence et al. 2022], for each observation x_i in a batch, the set of negative counter-examples $\{\tilde{y}_j^i\}_{j=1}^{N_{\text{neg}}}$ is generated by uniformly sampling from the action space: $\tilde{y}_j^i \sim U(y_{\min}, y_{\max})$. In this work, instead of uniform sampling, we generate negative counter-examples by sampling with replacement from the expert set[1] and subsequently adding Gaussian noise drawn from $\mathcal{N}(0, \sigma^2 I)$ to introduce perturbations. We refer to our method as *Data-Augmented IBC (DA-IBC)*. The set $\{\tilde{y}_j^i\}_{j=1}^{N_{\text{neg}}}$ is generated randomly and dynamically within the training loop for each observation x_i in a batch $\mathcal{B} = \{(x_i, y_i)\}$ from the expert dataset $\mathcal{D}$. As a result, different sets of counter-examples are created in each training iteration. Note that the actions in the set $\{\tilde{y}_j^i\}_{j=1}^{N_{\text{neg}}}$ are not dependent on any particular value of x_i, unlike [Singh et al. 2023] for instance. If the number of available expert actions is smaller than N_{neg}, we sample with replacement until the desired number is reached. Finally, each action in the set is perturbed with Gaussian noise to form the negative counter-examples.

2.4.1. Derivative-Free Optimization (DFO)

We use an adapted version of DFO from [Florence et al. 2022], shown in Algorithm 1, to perform implicit inference $\hat{y} = \arg\min_y E_\theta(x, y)$. In DFO, the samples $\{\tilde{y}_i\}$ are initialized by sampling from a categorical distribution of the expert actions with probabilities equal to the corresponding weights $\{\tilde{w}_i\}$, with $w_i = \frac{1}{\hat{g}(y_i^{\text{expert}})}$ obtained by KDE (Section 2.5). After the N_{iters} iterations of the method, instead of returning the action with maximum probability, a sample is returned by drawing from a categorical distribution of the actions $\{\tilde{y}_i\}$ and its respective probabilities $\{\tilde{p}_i\}$ found in the last iteration. This modification enhances the agent's potential to perform multimodal behavior.

2.5. Kernel Density Estimation (KDE)

Mathematically, the KDE [Silverman 1986] for a point x is given by:

$$\hat{g}(x) = \frac{1}{Nh} \sum_{j=1}^{N} K\left(\frac{x - x_j}{h}\right), \tag{4}$$

where $\hat{g}(x)$ is the estimated density at point x, N is the number of data points, h is the bandwidth, x_j are the data points, and K is the kernel function. In particular, we employ the Gaussian kernel $K(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}}$.

[1] The sampling is actually made from a random minibatch of actions from the expert dataset.

Algorithm 1 Derivative-Free Optimizer

Result: $\hat{y}$
Initialize: $\{\tilde{y}_i\}_{i=1}^{N_{\text{samples}}} \sim \text{Categorical}(\{y_i^{\text{expert}}\}, \{w_i\})$, $w_i \propto \text{KDE}(y_i^{\text{expert}})$, $\sigma = \sigma_{\text{init}}$
for iter $= 1, 2, \ldots, N_{\text{iters}}$ **do**
  $\{E_i\}_{i=1}^{N_{\text{samples}}} = \{E_\theta(\mathbf{x}, \tilde{y}_i)\}_{i=1}^{N_{\text{samples}}}$ (Compute energies);
  $\{\tilde{p}_i\}_{i=1}^{N_{\text{samples}}} = \left\{ \frac{e^{-E_i}}{\sum_{j=1}^{N_{\text{samples}}} e^{-E_j}} \right\}$ (Softmax probab.);
  if iter $< N_{\text{iters}}$ **then**
    $\{\tilde{y}_i\}_{i=1}^{N_{\text{samples}}} \sim \text{Multinomial}(N_{\text{samples}}, \{\tilde{p}_i\}_{i=1}^{N_{\text{samples}}},$
    $\{\tilde{y}_i\}_{i=1}^{N_{\text{samples}}})$ (Resample with replacement);
    $\{\tilde{y}_i\}_{i=1}^{N_{\text{samples}}} \leftarrow \{\tilde{y}_i\}_{i=1}^{N_{\text{samples}}} + \mathcal{N}(0, \sigma)$ (Add noise);
    $\{\tilde{y}_i\}_{i=1}^{N_{\text{samples}}} = \text{clip}(\{\tilde{y}_i\}_{i=1}^{N_{\text{samples}}}, y_{\min}, y_{\max})$
    $\sigma \leftarrow K\sigma$ (Shrink sampling scale);
  end if
end for
Return: $\hat{y} \sim \text{Categorical}(\{\tilde{y}_i\}, \{\tilde{p}_i\})$

In this work, the KDE is used to weigh the training samples in the BC and IBC losses in (1) and (2), respectively, and also to form the initial samples in the DFO inference Algorithm 1 . By using the inverse of the kernel density estimation of the actions as weights, we give more importance to less frequent actions. This approach enhances the model's ability to learn from sparsely represented regions of the action space. The weighted IBC loss is as follows:

$$\mathcal{L}_{\text{InfoNCE}}(\theta) = \sum_{i=1}^{N} -w_i \log \left(\tilde{p}_\theta(y_i \mid x_i, \{\tilde{y}_i^j\}_{j=1}^{N_{\text{neg}}}) \right). \tag{5}$$

where: $w_i = \frac{1}{\hat{g}(y_i)}$; and $\hat{g}(y_i)$ is the kernel density estimate evaluated at action y_i. For BC, the loss function is weighted analogously [Antonelo et al. 2024]. In practice, when training the model using stochastic gradient descent or ADAM, instead of directly applying the weighted loss, we employ a sampling strategy that increases the frequency of less common actions in the dataset. Specifically, at each training iteration, a minibatch is drawn using a weighted sampling approach, where the sampling probabilities are defined by the weights, mitigating data imbalance during training.

3. Agent

3.1. Input Representation

The agent's input is a five-channel 192×192 bird's-eye view (BEV) image generated by a CARLA module with city map access. This BEV includes two channels for the road and lane, plus three channels encoding traffic light history at time steps -16, -9, and -1. Additionally, the agent receives the vehicle's current speed and the previous control actions (acceleration and steering).

3.2. Action Representation

The vehicle in the CARLA simulator has three actuators: steering $[-1, 1]$, throttle $[0, 1]$, and brake $[0, 1]$. The agent's action space is $\mathbf{a} \in [-1, 1]^2$, where the two com-

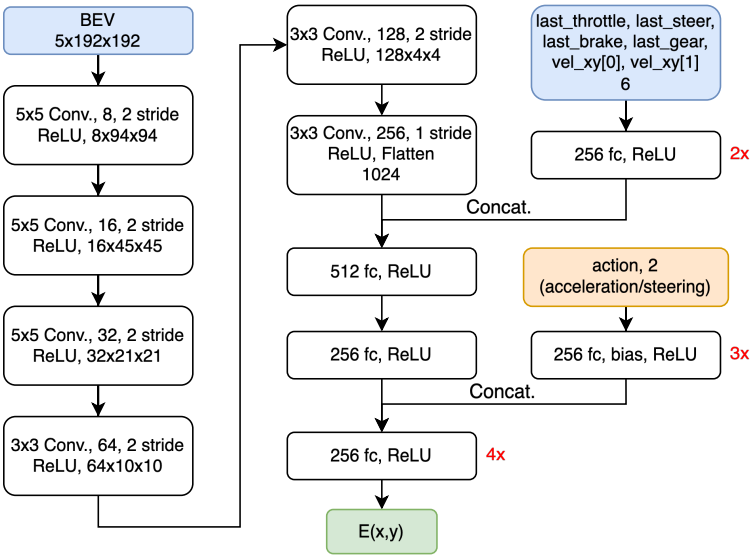


Figure 2. Detailed EBM network architecture. The BEV input has five 192×192 channels, while the state input consists of previous actions and current speed. The two-dimensional action input corresponds to acceleration and steering values. The output $E(x, y)$ is the energy for the observation-action pair (x, y).

ponents correspond to steering and acceleration. Negative acceleration implies braking, preventing simultaneous acceleration and braking [Petrazzini and Antonelo 2021]. While the network representing the agent outputs a energy value in IBC, which depends not only on an observation but also on an *action input*, in BC the output models a Beta distribution, which is naturally bounded and avoids manual clipping or squashing [Petrazzini and Antonelo 2021]. The policy network π_θ in BC outputs α and β, shaping the Beta distribution to adapt to diverse driving scenarios.

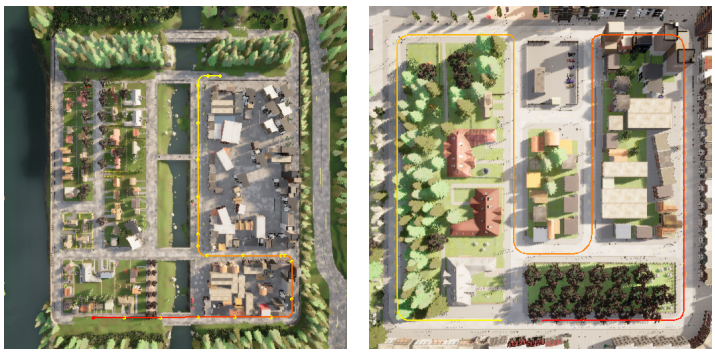
3.3. Network Architecture

The goal of the EBM architecture, shown in Fig. 2, is to learn the density of actions conditioned on the current BEV, and some state variables. The BEV can have a different number of channels, depending on the current experimental setup. In our case, it contains all five BEV channels from Fig. 1.

A Convolutional Neural Network (CNN) extracts a 256-dimensional embedding from the BEV. Similarly, a Multi-Layer Perceptron (MLP) encodes scalar state variables into a 256-dimensional vector. Together they form the current representation of the environment the agent faces, i.e., the agent's observation. The resulting embedding vector for a observation x and an action y are processed by another MLP to map to the energy value $E_\theta(x, y)$. Fig. 2 shows the detailed network structure.

4. Experiments

In this section, we compare the proposed DA-IBC agent with a traditional BC agent with a Beta policy, presented previously in [Antonelo et al. 2024], and to the conventional IBC agent.



(a) *town01* (b) *town02*

Figure 3. (a) The *town01* environment of the agent includes one of the routes used by the expert to collect data. This highlighted path measures 740 meters (b) The *town02* is used to evaluate the trained agent in a new environment.

4.1. Dataset Generation

The expert dataset is sourced from the CARLA Leaderboard evaluation platform [2], specifically the *town01* environment with ten predefined trajectories. A deterministic agent, guided by a dense point trajectory and a PID controller [Chen et al. 2020], generates the dataset. The expert follows a dense trajectory for precise navigation and obeys traffic signals, stopping at red lights and accelerating on green. Fig. 3(a) illustrates one of the ten routes used for training. This trajectory, invisible to the agent, is shown as a gradient line from yellow to red.

The dataset was recorded at 10 Hz, yielding 10 observation-action pairs per second. The shortest route contains 1480 samples (2.5 minutes), with an average route having 2129 samples (3.5 minutes). In total, the dataset comprises 21,287 labeled samples (30 GB), covering approximately 8 km or 36 minutes of driving.

4.2. Settings

All agents were trained with ADAM step size of 10^{-5}, for a maximum of 400 epochs, using minibatch size of $N = 128$. For EBM training, $N_{\text{neg}} = 1024$. For DFO inference, $N_{\text{samples}} = 16384$, $\sigma_{\text{init}} = 0.5$, $N_{\text{iters}} = 5$, and $K = 0.5$. The kernel bandwidth for KDE was set to $h = 0.2$ through trial-and-error experimentation, given the sensitivity of this parameter and its strong influence on the smoothness of the estimated density function. Additionally, Python and Pytorch were used to implement the networks and their training procedures.

4.3. Results

We trained agents with data collected in *town01* based on three methods, the proposed DA-IBC, conventional IBC, and BC with the outputs modeling a Beta distribution. The evaluation was done in *town02* CARLA environment (Fig. 3(b)), with the episode score computed as the traveled distance multiplied by the traffic light penalty. This score only shows the ability to drive without infractions, not taking into account action multimodality. In Fig. 4, this score is shown for all three methods, where the average was computed

[2] `https://leaderboard.carla.org/`

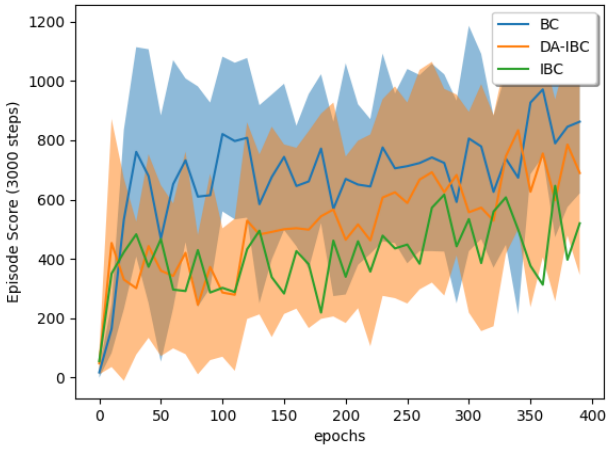


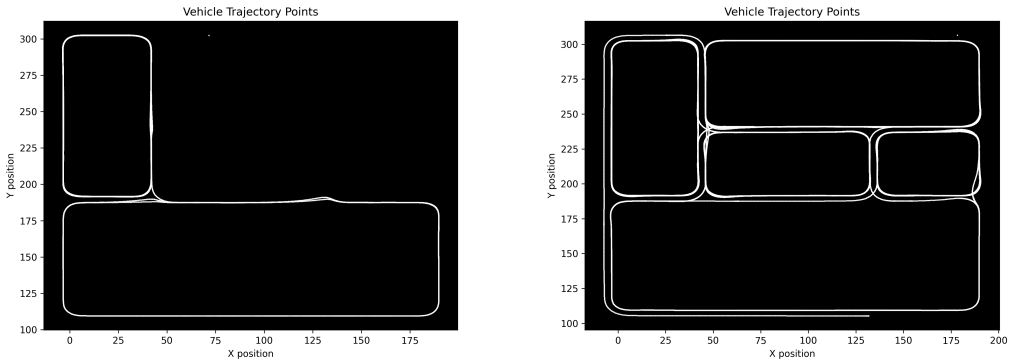
Figure 4. Training progress of agents BC, IBC and DA-IBC with training data from *town01* and evaluation in *town02*. The score is the distance traveled multiplied by the *traffic light penalty* (0.7^{n_i}, where n_i is the number of infractions in the episode of $300s$), and it primarily reflects safe driving behavior and not explicitly multimodality. For each method, the average and standard deviation are computed over three different agents evaluated ten times for $3,000$ steps (or $300s$). This evaluation is done each 10 training epochs. Note that both BC and IBC methods are offline training methods, and the above plot serves to monitor the intermediate steps of learning.

for three agents per method and over 10 episodes each of $3,000$ timesteps for each agent. BC achieves higher scores more quickly, but its task is easier: it predicts the average action for each observation, while IBC must learn an energy landscape over actions.

In order to evaluate multimodal decision making, we let both BC and DA-IBC agents navigate freely in *town02*, after their training with data from *town01*. In Fig.5, the BC agent primarily follows a single trajectory through the town, with occasional deviations caused by stochasticity in the simulator. These deviations do not reflect true multimodal behavior, but rather arise from noise during action execution. In contrast, the DA-IBC agent learns a genuinely multimodal policy that allows it to take distinct actions at the same decision points, enabling it to explore and cover the entire city. Neither simulation resulted in crashes, though some traffic light infractions occurred.

We also investigated the energy landscape of the trained agent in three situations in *town02*. In Fig. 6, we can see the agent choosing to turn left [3] or right [4] at the same T-intersection, and the corresponding energy function, where the blue color corresponds to lower energy. Ten inferences were made with the proposed DFO algorithm and plot as 10 white points in the energy landscape, for each case. In Fig. 6, the energy landscape for the vehicle stopping at the red light indicates a slightly different learned function, with the minimum at (0,0), which signifies zero acceleration and zero steering. These energy landscapes act like probability distributions, where low-energy actions are the most likely, even under conflicting expert strategies. Thus, as seen in Fig. 6, both turning behaviors

[3] https://youtu.be/_SLzqnMWtt8
[4] https://youtu.be/AhALxsZm7wg



BC DA-IBC

Figure 5. Trajectory of agents during a simulation of $3,000s$ in *town02*, without a route to follow. Left: The BC agent can not learn multimodal behavior, being stuck into one or two blocks (likely due to noise) of the city. Right: The DA-IBC agent learned multimodal action distributions, being able to make different decisions at the same intersections of the city.

were learned and modeled by the EBM even though no explicit command of turning left or right were present in the expert training set. Notably, the energy landscapes exhibit flatter minima in multimodal settings and sharper minima in unimodal ones—such as the third column in Fig. 6, corresponding to stopping at a red light. Additionally, the ten DFO inferences tend to cluster tightly, suggesting that the model concentrates probability mass in small regions of the action space.

The average traveled distances of the BC, IBC, and DA-IBC agents in Town2 were 9,074, 4,855, and 12,741, respectively. Our proposed DA-IBC significantly outperformed the baseline IBC in terms of traveled distance. However, it exhibited a higher traffic light infraction rate compared to BC—an issue that we aim to address in future work. We also hypothesize that the higher infraction rate observed in IBC may stem from its greater exploratory behavior. Unlike BC, which tends to mimic expert demonstrations conservatively, IBC explores a broader range of scenarios, potentially encountering more opportunities for infractions in the absence of explicit penalty mechanisms.

5. Conclusion

In this work, we proposed an EBM architecture for autonomous vehicle navigation in urban environments simulated in CARLA, that maps an observation-action pair to an energy value. As the same observation (a Bird's-Eye View of the vehicle scene, last actions, and speed) can be paired to multiple conflicting actions in the expert training set, the EBM can model multimodal action distributions by learning an energy landscape as a function of the action input. We proposed DA-IBC (Data-Augmented IBC), an extension of standard IBC that introduces two key innovations: (1) it modifies the inference DFO algorithm to initialize action samples from the expert dataset instead of a uniform distribution; and (2) it updates the training loss to generate counterexamples by perturbing the expert's actions, refining the energy function to better separate expert behavior from nearby alternatives.

We omitted the BEV route channel to better test multimodal decisions at T-intersections in a new city (*town02*), using agents trained in *town01*. In this way, even

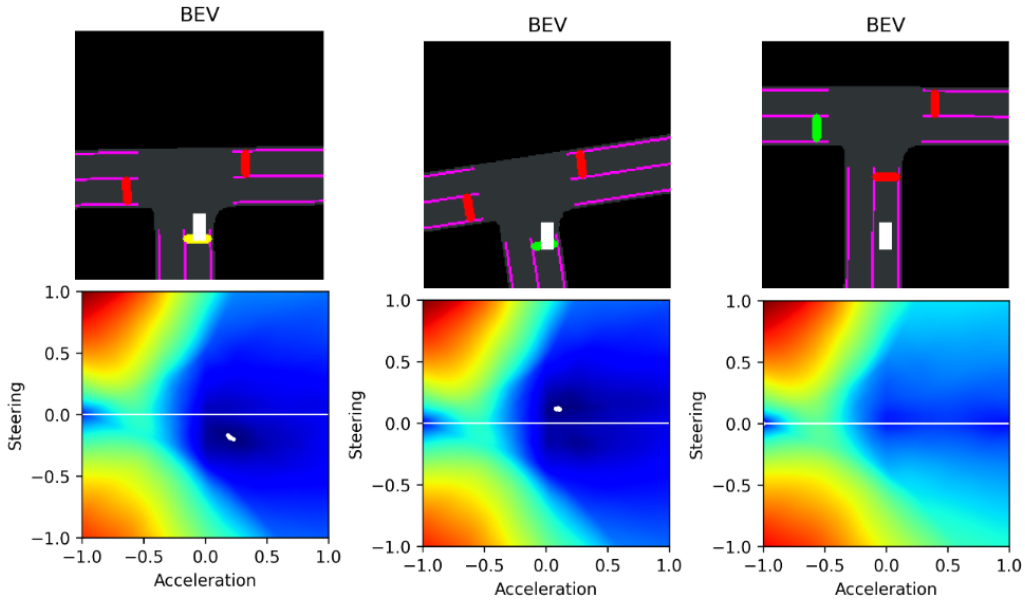


Figure 6. Top: the BEV scene as perceived by the agent. Bottom: energy $E(s, a)$ as a function of the two-dimensional action a (i.e., steering and acceleration). A horizontal white line marks the zero steering action. The first two columns represent situations where the steering action distribution is multimodal, such as entering an intersection: the minimum in the energy function (blue color) along the steering axis spans a broader range than the third column (stopping at a red light). The first column corresponds to a path turning left, while the second one shows the agent turning right at the same intersection. Ten DFO inferences were run with the BEV input fixed to the above scene, with corresponding action inferences plotted as small white points in the energy function. The latter column shows the steering action distribution nearly unimodal, with DFO inferences at (0,0).

though the navigation is routeless, it serves the purpose to evaluate the capacity of the EBM to learn the multimodal distributions, which we have shown to be successful. Future works include the extension to cities with pedestrians and other vehicles, and the design of a goal-directed agent from a routeless model as trained in this work, which can take high-level commands in order to reach a destination.

References

Antonelo, E. A., Couto, G. C. K., Möller, C., and Fernandes, P. H. (2024). Investigating behavior cloning from few demonstrations for autonomous driving based on bird's-eye view in simulated cities. In *Brazilian Conference on Intelligent Systems*, pages 155–168. Springer.

Balesni, M., Tampuu, A., and Matiisen, T. (2023). Controlling steering with energy-based models. *arXiv preprint arXiv:2301.12264*.

Bansal, M., Krizhevsky, A., and Ogale, A. (2018). Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., et al. (2016). End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*.

Chen, D., Zhou, B., Koltun, V., and Krähenbühl, P. (2020). Learning by cheating. In *Conference on robot learning*, pages 66–75. PMLR.

Codevilla, F., Müller, M., Dosovitskiy, A., López, A. M., and Koltun, V. (2018). End-to-end driving via conditional imitation learning. *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 1–9.

Codevilla, F., Santana, E., Lopez, A., and Gaidon, A. (2019). Exploring the limitations of behavior cloning for autonomous driving. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9328–9337.

Couto, G. C. K. and Antonelo, E. A. (2024). Hierarchical generative adversarial imitation learning with mid-level input generation for autonomous driving on urban environments. *IEEE Transactions on Intelligent Vehicles*, pages 1–14.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. (2017). CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16.

Du, Y. and Mordatch, I. (2019). Implicit generation and modeling with energy based models. *Advances in Neural Information Processing Systems*, 32.

Florence, P., Lynch, C., Zeng, A., Ramirez, O. A., Wahid, A., Downs, L., Wong, A., Lee, J., Mordatch, I., and Tompson, J. (2022). Implicit behavioral cloning. In *Conference on Robot Learning*, pages 158–168. PMLR.

Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Pang, B., Zhao, T., Xie, X., and Wu, Y. N. (2021). Trajectory prediction with latent belief energy-based model. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11814–11824.

Petrazzini, I. G. and Antonelo, E. A. (2021). Proximal policy optimization with continuous bounded action space via the beta distribution. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8. IEEE.

Pomerleau, D. A. (1991). Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3(1):88–97.

Rhinehart, N., McAllister, R., and Levine, S. (2018). Deep imitative models for flexible inference, planning, and control. *arXiv preprint arXiv:1810.06544*.

Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, London.

Singh, S., Tu, S., and Sindhwani, V. (2023). Revisiting energy based models as policies: Ranking noise contrastive estimation and interpolating energy models. *arXiv preprint arXiv:2309.05803*.

Xu, H., Gao, Y., Yu, F., and Darrell, T. (2017). End-to-end learning of driving models from large-scale video datasets. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2174–2182.