

BeTAIL: Behavior Transformer Adversarial Imitation Learning from Human Racing Gameplay

Catherine Weaver¹, Chen Tang^{1,2}, Ce Hao^{1,3}, Kenta Kawamoto⁴, Masayoshi Tomizuka¹, Wei Zhan¹

Abstract—Autonomous racing poses a significant challenge for control, requiring planning minimum-time trajectories under uncertain dynamics and controlling vehicles at their handling limits. Current methods requiring hand-designed physical models or reward functions specific to each car or track. In contrast, imitation learning uses only expert demonstrations to learn a control policy. Imitated policies must model complex environment dynamics and human decision-making. Sequence modeling is highly effective in capturing intricate patterns of motion sequences but struggles to adapt to new environments or distribution shifts that are common in real-world robotics tasks. In contrast, Adversarial Imitation Learning (AIL) can mitigate this effect, but struggles with sample inefficiency and handling complex motion patterns. Thus, we propose BeTAIL: Behavior Transformer Adversarial Imitation Learning, which combines a Behavior Transformer (BeT) policy from human demonstrations with online AIL. BeTAIL adds an AIL residual policy to the BeT policy to model the sequential decision-making process of human experts and correct for out-of-distribution states or shifts in environment dynamics. We test BeTAIL on three challenges with expert-level demonstrations of real human gameplay in the high-fidelity racing game Gran Turismo Sport. Our proposed BeTAIL reduces environment interactions and improves racing performance and stability, even when the BeT is pretrained on different tracks than downstream learning. Videos and code available at: <https://sites.google.com/berkeley.edu/BeTAIL/home>.

Index Terms—Imitation Learning, Reinforcement Learning, Deep Learning Methods

I. INTRODUCTION

AUTONOMOUS racing is of growing interest to inform controller design at the limits of vehicle handling and provide a safe alternative to racing with human drivers [1]. An autonomous racer’s driving style should resemble *human-like racing behavior* in order to behave in safe and predictable ways that align with the written and social rules of the race [2]. High-fidelity racing simulators, such as the world-leading Gran Turismo Sport (GTS) game, can test policies in a safe and realistic environment and benchmark comparisons between autonomous systems and human drivers [2], [3]. Reinforcement learning (RL) outperforms expert human players but requires iterative tuning of dense rewards [2], which is susceptible

Manuscript received: January 29, 2024; Revised May 1, 2024; Accepted May 31, 2024.

This paper was recommended for publication by Editor Aleksandra Faust upon evaluation of the Associate Editor and Reviewers’ comments.

¹Department of Mechanical Engineering, University of California Berkeley, CA, USA. C. Weaver is supported by NSF GFRP Grant No. DGE 1752814. Contact: catherine22@berkeley.edu

²Department of Computer Science, University of Texas, Austin, USA.

³School of Computing, National University of Singapore, Singapore

⁴Sony Research, Tokyo, Japan.

Digital Object Identifier (DOI): see top of this page.

to ad hoc trial and error [4]. Imitation learning (IL) is a potential solution that mimics experts’ behavior with offline demonstrations [5]. We propose a novel IL algorithm to model non-Markovian decision-making of human experts in GTS.

Human racing includes complex decision-making and understanding of environment dynamics [6], and the performance of Markovian policies can deteriorate with human demonstrations [7]. Sequence-based transformer architectures [8], [9], similar to language models [10], accurately model the complex dynamics of human thought [11]. The Behavior Transformer (BeT) [12], and Trajectory Transformer [9], which do not require pre-defined environment rewards, are casually conditioned on the past to accurately model long-term dynamics [9]. Policies are trained via supervised learning to autoregressively maximize the likelihood of trajectories in the offline dataset. Policies are limited by dataset quality [13] and are sensitive to variation in system dynamics and out-of-distribution states.

Adversarial Imitation Learning (AIL) [14] overcomes the issues with offline learning with adversarial training and reinforcement learning. A discriminator network encourages the agent to match the state occupancy of online rollouts and expert trajectories, reducing susceptibility to distribution shift when encountering unseen states [5]. However, AIL requires extensive environment interactions, and its performance deteriorates with human demonstrations [7]. Thus, AIL in racing is unstable and sample inefficient. AIL also exhibits shaky steering behavior and often spins off the track, since AIL does not model humans’ non-Markovian decision-making.

We propose Behavior Transformer Adversarial Imitation Learning (BeTAIL), which leverages offline sequential modeling and online occupancy-matching fine-tuning to 1.) capture the sequential decision-making process of human demonstrators and 2.) correct for out-of-distribution states or minor shifts in environment dynamics. First, a BeT policy is learned from offline human demonstrations. Then, an AIL mechanism finetunes the policy to match the state occupancy of the demonstrations. BeTAIL adds a residual policy, e.g. [15], to the BeT action prediction; the residual policy refines the agent’s actions while remaining near the action predicted by the BeT. Our contributions are as follows:

- 1) We propose Behavior Transformer Adversarial Imitation learning (BeTAIL) to pre-train a BeT and fine-tune it with a residual AIL policy to learn complex, non-Markovian behavior from human demonstrations.
- 2) We show that when learning a racing policy from real human gameplay in Gran Turismo Sport, BeTAIL outperforms BeT or AIL alone while closely matching non-Markovian patterns in human demonstrations.

- 3) We show BeTAIL when pre-training on a library of demonstrations from multiple tracks to improve sample efficiency and performance when fine-tuning on an unseen track with a single demonstration trajectory.

In the following, we discuss related works (Section II) and preliminaries (Section III). Then we introduce BeTAIL in Section IV and describe our method for imitation of human gameplay in Section V. Finally, Section VI describes three challenges in GTS with concluding remarks in Section VII.

II. RELATED WORKS

A. Behavior Modeling

Behavior modeling aims to capture human behavior, which is important for robots and vehicles that operate in proximity to humans [16]. AIL overcomes the problem of cascading errors with traditional techniques like behavioral cloning and parametric models [17]. Latent variable spaces allow researchers to model multiple distinct driving behaviors [18], [19], [20], [21]. However, sample efficiency and training stability are common problems with AIL from scratch, which is exacerbated when using human demonstrations [7]. Augmented rewards [18], [20] or negative demonstrations [22] can accelerate training but share the same pitfalls as reward shaping in RL.

B. Curriculum Learning and Guided Learning

Structured training regimens can accelerate RL. Curriculum learning gradually increases the difficulty of tasks [23] and can be automated with task phasing, which gradually transitions a dense imitation-based reward to a sparse environment reward [24]. “Teacher policies” can accelerate RL through policy intervention to prevent unsafe actions [25], [26] or guided policy search [27] to guide the objective of the agent. Thus, large offline policies can be distilled into lightweight RL policies [11]. BeTAIL uses residual RL policies [28], [15], [29], which adapt to task variations by adding a helper policy that can be restricted close to the teacher’s action [30], [31].

C. Sequence Modeling

Sequence-based modeling in offline RL predicts the next action in a trajectory sequence, which contains states, actions, and optionally goals. Commonly, goals are set to the return-to-go, i.e. the sum of future rewards [8], [9], but advantage conditioning improves performance in stochastic environments [32]. Goal-conditioned policies are fine-tuned with online trajectories and automatic goal labeling [13]. Sequence models can be distilled into lightweight policies with offline RL and environment rewards [11]. However, when rewards are not available, e.g. IL [12], [9], offline sequence models suffer from distribution shift and poor dataset quality [13].

III. PRELIMINARIES

A. Problem Statement

We model the learning task as a Markov Process (MP) defined by $\{\mathcal{S}, \mathcal{A}, T\}$ of states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, and the transition probability $T(s_t, a_t, s_{t+1}) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$.

Note that, unlike RL, we do not have access to an environment reward. We have access to human expert demonstrations in the training environment consisting of a set of trajectories, $D_E = (\tau_0^E, \tau_1^E, \dots, \tau_M^E)$, of states and actions at every time step $\tau = (s_t, a_t, \dots)$. The underlying expert policy, π_E , is unknown. The goal is to learn the agent’s policy π , that best approximates the expert policy π_E . Each expert trajectory τ^E , consists of states and action pairs: $\tau^E = (s_0, a_0, s_1, a_1, \dots, s_N, a_N)$. The human decision-making process of the expert is unknown and likely non-Markovian [33]; thus, imitation learning performance can deteriorate with human trajectories [7].

B. Unimodal Decision Transformer

The Behavior Transformer (BeT) processes the trajectory τ_E as a sequence of 2 types of inputs: states and actions. The original BeT implementation [12] employed a mixture of Gaussians to model a dataset with multimodal behavior. For simplicity and to reduce the computational burden, we instead use an unimodal BeT that uses a deterministic similar to the one originally used by the Decision Transformer [8]. However, since residual policies can be added to black-box policies [15], BeTAIL’s residual policy could be easily added to the k-modes present in the original BeT implementation.

At timestep t , the BeT uses the tokens from the last K timesteps to generate the action a_t , where K is referred to as the context length. Notably, the context length during evaluation can be shorter than the one used for training. The BeT learns a deterministic policy $\pi_{\text{BeT}}(a_t | s_{-K,t})$, where $s_{-K,t}$ represents the sequence of K past states $s_{\max(1,t-K+1):t}$. The policy is parameterized using the minGPT architecture [34], which applies a causal mask to enforce the autoregressive structure in the predicted action sequence.

A notable strength of BeT is that the policy can model non-Markovian behavior; in other words, rather than modeling the action probability as $P(a_t | s_t)$, the policy models the probability $P(a_t | s_t, s_{t-1}, \dots, s_{t-h+1})$. However, the policy is trained using *only* the offline dataset, and even minor differences between the training data and evaluation environment can lead to large deviations in the policies’ performance [5].

IV. BEHAVIOR TRANSFORMER-ASSISTED ADVERSARIAL IMITATION LEARNING

We now present Behavior Transformer-Assisted Adversarial Imitation Learning (BeTAIL), summarized in Fig. 1. BeTAIL consists of a pretrained BeT policy and a corrective residual policy [30]. First, an unimodal, causal Behavior Transformer (BeT) [12] is trained on offline demonstrations to capture the sequential decision-making of the human experts. Then, the BeT policy is frozen, and a lightweight, residual policy is trained with Adversarial Imitation Learning (AIL). Thus, BeTAIL combines offline sequential modeling and online occupancy-matching fine-tuning to capture the decision-making process of human demonstrators and adjust for out-of-distribution states or environment changes.

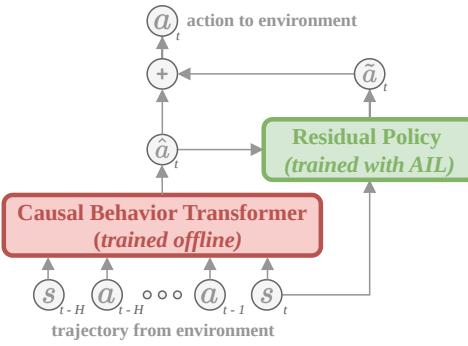


Fig. 1: BeTAIL rollout collection. The pre-trained BeT predicts action \hat{a}_t from the last H state-actions. Then the residual policy specifies action \tilde{a}_t from the current state and \hat{a}_t , and the agent executes $a_t = \hat{a}_t + \tilde{a}_t$ in the environment.

A. Behavior Transformer (BeT) Pretraining

A unimodal BeT policy $\hat{a} = \pi_{\text{BeT}}(a_t | s_{-K,t})$ is used to predict a base action \hat{a} . Following [8], [13], sub-trajectories of length K in the demonstrations are sampled uniformly and trained to minimize the difference between the action predicted by the BeT and the next action in the demonstration sequence. While the action predicted by the BeT, \hat{a} , considers the previous state history $s_{-K,t}$, the action may not be ideal in stochastic environments [32]. Existing methods to update Transformer models require rewards and rely on supervised learning or on-policy algorithms [35]. However, the complex racing tasks requires off-policy RL to converge to the maximum reward [3]. Further, we assume an environment reward is not available; rather, we wish to replicate human demonstrations. Thus, we freeze the weights of the BeT after the pretraining stage, and add a corrective residual policy [15] that can be readily trained with off-policy AIL.

B. Residual Policy Learning for Online Fine-tuning

The agent's action is the sum of the action specified by the BeT and the action from a residual policy [15], which corrects or improves the actions from the BeT base policy. We define an augmented MP: $\tilde{\mathcal{M}} = \{\tilde{\mathcal{S}}, \tilde{\mathcal{A}}, \tilde{T}\}$. The state space $\tilde{\mathcal{S}}$ is augmented with the base action: $\tilde{s}_t \doteq [s_t \ a_t]^\top$. The action space, $\tilde{\mathcal{A}}$, is the space of the residual action \tilde{a} . The transition probability, $\tilde{T}(\tilde{s}_t, \tilde{a}_t, \tilde{s}_{t+1})$, includes both the dynamics of the original environment, i.e. $T(s_t, a_t, s_{t+1})$, and also the dynamics of the base action $\hat{a}_t = \pi_{\text{BeT}}(\cdot | s_{-K,t})$.

The residual action is predicted by a Gaussian residual policy, $\tilde{a} \sim f_{\text{res}}(\tilde{a} | s_t, \hat{a}_t) = \mathcal{N}(\mu, \sigma)$, conditioned on the current state and the base policy's action. The action in the environment, a_t , is the sum of the base and residual actions:

$$a_t = \hat{a}_t + \text{clip}(\tilde{a}_t, -\alpha, \alpha). \quad (1)$$

The residual policy is constrained between $[-\alpha, \alpha]$, constraining how much the environment action a_t is allowed to deviate from the base action \hat{a} [30], [15]. For small α , the environment action must be close to the BeT base action, \hat{a}_t .

Assuming the action space of $f_\theta(\cdot | s_t, \hat{a}_t)$ is restricted to the range $[-\alpha, \alpha]$, we can define the policy $\pi_A(a | s_{-k,t})$:

$$\begin{aligned} \pi_A(a | s_{-k,t}) &= \\ \tilde{a}_t | \tilde{a}_t \sim f_{\text{res}}(\cdot | s_t, \pi_{\text{BeT}}(\cdot | s_{-k,t})) &+ \pi_{\text{BeT}}(\cdot | s_{-k,t}). \end{aligned} \quad (2)$$

Eq. 2 follows the notation of [36]. Because the BeT policy is non-Markovian, the agent's policy π_A is also non-Markovian. However, using the definition of the augmented state, \tilde{s} , the input to the AIL residual policy is the current state and the base action predicted by the BeT. Thus, π_A is in fact Markovian with respect to the augmented state \tilde{s} :

$$\pi_A(a | s_t, \hat{a}_t) = \tilde{a}_t | \tilde{a}_t \sim f_{\text{res}}(\cdot | s_t, \hat{a}_t) + \hat{a}_t. \quad (3)$$

Per prior works [15], [36], during online training, the agent's policy during rollouts is given by π_A in (3) as the sum of the action from the base policy π_{BeT} and the action from the residual policy f_{res} (2). After pretraining (Section IV-A), the base policy, π_{BeT} , is frozen; then, the agent's policy is improved by updating the residual policy f_{res} with AIL.

C. Residual Policy Training with AIL

To train the residual policy, f_{res} , we adapt Adversarial Imitation Learning (AIL) [37] so that the agent's policy solves the AIL objective. AIL is defined in terms of a single Markovian policy that is not added to a base policy [14]. In this section, we detail our changes to employ AIL to update the residual policy f_{res} . Given the definition of π_A (3), we define the AIL objective for residual policy learning as

$$\underset{f_{\text{res}}}{\text{minimize}} \ D_{\text{JS}}(\rho_{\pi_A}, \rho_{\pi_E}) - \lambda H(f_{\text{res}}). \quad (4)$$

Similar to the standard AIL objective [37], we still aim to minimize the distance between the occupancy measures of the expert's and agent's policies, denoted as ρ_{π_E} and ρ_{π_A} respectively in Eq. (4). The minimization with respect to f_{res} is valid since Eq. (3) simply defines a more restrictive class of policies than standard single-policy AIL. Since the contribution from the base policy is deterministic, we regularize the problem using only the entropy of the residual policy, and the policy update step is replaced with

$$\max_{f_{\text{res}}} \mathbb{E}_{\tilde{\tau} \sim f_{\text{res}}} \left[\sum_{t=0}^{\infty} \gamma^t \left(\tilde{R}^E(\tilde{s}_t, \tilde{a}_t) + \lambda H(f_{\text{res}}(\cdot | \tilde{s}_t)) \right) \right], \quad (5)$$

where $\tilde{\tau}$ represents the trajectory $\tilde{\tau} = (\tilde{s}_0, \tilde{a}_0, \dots)$ collected using f_{res} on $\tilde{\mathcal{M}}$. Thus, f_{res} is updated using the augmented state and residual action. In residual policy RL [15], the reward is calculated using the action taken in the environment. Similarly, we define \tilde{R}^E as AIL's proxy reward on $\tilde{\mathcal{M}}$:

$$\tilde{R}^E(\tilde{s}_t, \tilde{a}_t) = -\log(1 - D_\omega^E(s, \hat{a} + \tilde{a})), \quad (6)$$

where $D_\omega^E(s, \hat{a} + \tilde{a})$ is a binary classifier trained to minimize

$$\begin{aligned} \mathcal{L}_{D, \mathcal{D}_E}(\omega) &= -E_{\tau_E \sim \mathcal{D}_E} [\log(D_\omega^E(s, a))] \\ &- E_{\tilde{\tau} \sim f_{\text{res}}} [\log(1 - D_\omega^E(s, \hat{a} + \tilde{a}))], \end{aligned} \quad (7)$$

which is equivalent to

$$\begin{aligned} \mathcal{L}_{D, \mathcal{D}_E}(\omega) &= -E_{\tau_E \sim \mathcal{D}_E} [\log(D_\omega^E(s, a))] \\ &- E_{\tau \sim \pi_A} [\log(1 - D_\omega^E(s, a))]. \end{aligned} \quad (8)$$

Eq. (6) implies that given the pair (\tilde{s}, \tilde{a}) , the reward is the probability that the state-action pair $(s, a) = (s, \hat{a} + \tilde{a})$ comes from the expert policy, according to the discriminator. Thus, by iterating between the policy learning objective (5) and the discriminator loss (8), AIL minimizes (4) to find the f_{res} that allows π_A to match the occupancy measure of the expert π_E .

V. IMITATION OF HUMAN RACING GAMEPLAY

We describe our method to learn from human gameplay in GTS, including the state features, environment, and training.

A. State Feature Extraction and Actions

The state includes features that were previously shown as important in RL. We include the following states exactly as described in [3]: 1) linear velocity, $v_t \in \mathbb{R}^3$, and linear acceleration $\dot{v}_t \in \mathbb{R}^3$ with respect to the inertial frame of the vehicle; 2) Euler angle $\theta_t \in (-\pi, \pi]$ between the 2D vector that defines the agent's rotation in the horizontal plane and the unit tangent vector that is tangent to the centerline at the projection point; 3) a binary flag with $w_t = 1$ indicating wall contact; and 4) N sampled curvature measurement of the course centerline in the near future $c_t \in \mathbb{R}^N$.

Additionally, we select features similar to those used in [2]: 5.) The cosine and sine of the vehicle's current heading, $\cos(\psi)$ and $\sin(\psi)$; and 6.) The relative 2D distance from the vehicle's current position to the left, $e_{LA,l}$, right, $e_{LA,r}$, and center, $e_{LA,c}$, of the track at 5 "look-ahead points." The look-ahead points are placed evenly using a look-ahead time of 2 seconds, i.e., they are spaced evenly over the next 2 seconds of track, assuming the vehicle maintains its current speed. The full state is a vector composed of $s_t = [v_t, \dot{v}_t, \theta_t, w_t, c_t, \cos(\psi), \sin(\psi), e_{LA,l}, e_{LA,r}, e_{LA,c}]$. The state is normalized using the mean and standard deviation of each feature in the demonstrations.

GTS receives the steering command $\delta \in [-\pi/6, \pi/6]$ rad and a throttle-brake signal $\omega_\tau \in [-1, 1]$ where $\omega_\tau = 1$ denotes full throttle and $\omega_\tau = -1$ denotes full brake [3], [2]. For all baseline comparisons, the steering command in the demonstrations is scaled to be between $[-1, 1]$. The agent specifies steering actions between $[-1, 1]$, which are scaled to $\delta \in [-\pi/6, \pi/6]$ before being sent to GTS.

When a residual policy is learned (see BeTAIL and BCAIL in the next section), the residual network predicts $\tilde{a} \in [-1, 1]$, and then \tilde{a} is scaled¹ to $[-\alpha, \alpha]$ and then added to $a = \hat{a} + \tilde{a}$. Since $\hat{a} \in [-1, 1]$, it is possible for a to be outside the bounds of $[-1, 1]$, so we clip a before sending the action to GTS. The choice of the hyperparameter α determines the maximum magnitude of the residual action. Depending on the task and the strength of the base policy, relatively large α allows the residual policy to correct for bad actions [36], or small α ensures the policy does not deviate significantly from the base policy [15], [31]. In this work, we set α as small as possible so that the policy remains near the offline BeT, which captures non-Markovian human racing behavior.

¹The choice to scale the residual policy between $[-1, 1]$ is made based on the practice of using scaled action spaces with SAC [38]

BeT Pretraining		AIL Discriminator Training	
Layers	4	Disc. Net. Arch.	[32,32]
Attention heads	4	Updates per iter	32
Embedding dim.	512	Learning Rate	0.005
Train context length	20	Entropy Scale	0.001
Dropout	0.1	Grad. Pen. Scale	10.0
Nonlinearity function	ReLU	Grad. Pen. Target	1.0
Batch size	256	Demo Batch Size	2,000
Pretrain Updates	500,000	Optimizer	Adam
Eval context length	5	AIL Policy Training (SAC)	
Optimizer	Lamb[13]	Network Arch.	[256,256]
Learning rate	0.0001	Polyak Update	0.002
Weight decay	0.0005	Discount Factor	0.99
		Gradient Steps	2500
		Replay Buffer	1M
		Learning rate	0.0003
		Batch size	4096
		Optimizer	Adam
Demo Datasets		env steps (~demonstration time)	
BeT(1-track)	294k(1.6hr)	AIL (Maggiore)	294k(1.6hr)
BeT(4-track)	456k(2.1hr)	AIL (Drag. Tail)	75k(21min)
		AIL (Mt. Pan)	8k(2min)

TABLE I: Training Hyperparameters and Demonstrations

B. Environment and Data Collection

1) *Gran Turismo Sport Racing Simulator:* We conduct experiments in the high-fidelity PlayStation (PS) game Gran Turismo Sport (GTS) (<https://www.gran-turismo.com/us/>), developed by Polyphony Digital, Inc. GTS takes two continuous inputs: throttle/braking and steering. The vehicle positions, velocities, accelerations, and pose are observed. The agent's and demonstrator's state features and control inputs are identical. RL achieved super-human performance against human opponents in GTS [2] but required tuned, dense reward functions to behave "well" with opponents. Rather than crafting a reward function, we explore if expert demonstrations can inform a top-performing agent.

To collect rollouts and evaluation episodes, the GTS simulator runs on a PS5, while the agent runs on a separate desktop computer. To accelerate training and evaluation, 20 cars run on the PS, starting at evenly spaced positions on the track [3]. The desktop computer (Alienware-R13, CPU Intel i9-12900, GPU Nvidia 3090) communicates with GTS over a dedicated API via an Ethernet as described in [3]. The API provides the current state of 20 cars and accepts car control commands, which are active until the next command is received. While the GTS state is updated every 60Hz, we limit the control frequency of our agent to 10Hz [3], [2]. Unlike prior works that collected rollouts on multiple PSs, we employ a single PS to reduce the desktop's computational burden and ensure BeTAIL inference meets the 10Hz control frequency. During training, rollouts are 50s (500 steps); during evaluation, rollouts are 500s (5,000 steps) to allow sufficient time to complete full laps. Experiments use 8M online environment steps (~222 hours); with 20 cars collecting data, this results in ~25 hours wall-clock.

2) *Training and Baselines:* For each challenge, the BeT, $\pi_{\text{BeT}}(\cdot | s_{-K,t})$, is pretrained on offline human demonstrations from one or more tracks. Then our **BeTAIL** fine-tunes control by training an additive residual policy, f_{res} , with AIL on the downstream environment and human demonstrations. The residual policy is constricted between $[-\alpha, \alpha]$, where α is

specified individually for each challenge. Training hyperparameters are listed in Table I. In Fig. 3a-c, **red** corresponds to demonstrations for the BeT and **green** corresponds to the downstream environment and demonstrations for the residual AIL policy. The **BeT** baseline uses the BeT policy, $\pi_{\text{BeT}}(\cdot|s_{-K,t})$, with the pretraining demonstrations **red**. **AIL** consists of a single policy, $\pi_A(a|s)$, trained via AIL using only the downstream demonstrations and environment in **green**. **BCAIL** trains a BC policy, $\pi_A(a|s)$, on the offline demonstrations (in **red**) and then fine-tunes a residual policy in the downstream environment (in **green**) following the same training scheme as BeTAIL. Table I lists hyperparameters for BeT pretraining and AIL training.

Agents use soft actor-critic (SAC) [38] for the reinforcement learning algorithm in adversarial learning (i.e. for AIL, BCAIL, and BeTAIL). Since SAC is off-policy, the replay buffer can include historical rollouts from many prior iterations. However, the discriminator, and thus the reward associated with a state-action pair, may have changed. When sampling data from the replay buffer for policy and Q-network training, we re-calculate the reward for each state-action pair with the current discriminator network. Finally, discriminator overfitting can deteriorate AIL performance, so for all baselines, AIL employs two discriminator regulators: gradient penalty and entropy regularization [7].

3) Demonstrations: All demonstrations were recorded from *different, expert-level* human players GTS participating in real gameplay, “time-trial” competitions with the Audi TT Cup vehicle. Each demonstration contains a full trajectory of states and actions, around a single lap of the track. The trajectories were recorded at a frequency of 60Hz, which we downsample to 10Hz (the agent’s control frequency). Each trajectory (i.e. lap) contains approximately 7000 timesteps, which are split into 500-step segments to match the length of the training episodes. Listed in Fig. 3a-c and Table I, we test three training schemes, either finetuning the BeT on the same race track as pretraining or on new, unseen tracks.

VI. RACING EXPERIMENT RESULTS

There are three challenges with distinct pretraining datasets and downstream environments. For evaluation, 20 cars are ran-

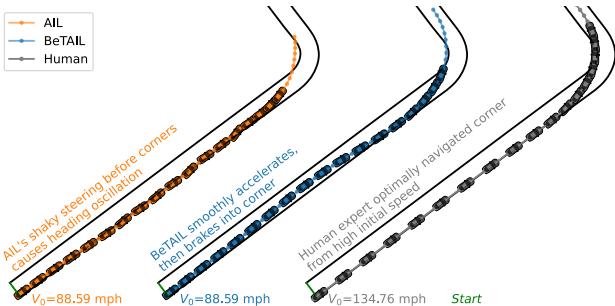


Fig. 2: Agent trajectories on Lago Maggiore. We deliberately set AIL and BeTAIL to start at a lower initial speed than the human. Car drawing is placed at the vehicle’s location and heading every 0.4s. See website for the animated version.

domly placed evenly on the track on one PS. Each car’s initial speed is set to the expert’s speed at the nearest position in a demonstration. Each car has 500 seconds (5000 steps) to complete a full lap. In Fig. 3d-f, we provide two evaluation metrics during training: the proportion of cars that finish a lap (top) and the average lap times of cars that finish (bottom). Higher success rates and lower lap times indicate better performance. Error bars show the total standard deviation across the 20 cars and 3 seeds. Fig. 3g-i provide the mean \pm standard deviation of the best policy’s lap time and absolute change in steering $|\delta_t - \delta_{t-1}|$, as RL policies can exhibit undesirable shaky steering behavior [2]. Videos of trajectories and GTS game are provided at <https://sites.google.com/berkeley.edu/BeTAIL/home>.

A. *Lago Maggiore Challenge: Fine-tuning in the same environment with BeTAIL*

We test if BeTAIL can fine-tune the BeT policy on the same track as the downstream environment (Fig. 3a). BeTAIL(0.05) employs a small residual policy, $\alpha = 0.05$, so that the agent’s action is close to the action predicted by the BeT [30]. The ablation BeTAIL(1.0) uses a large residual policy, $\alpha = 1.0$. There are 49 demonstrations from different human players on the Lago Maggiore track. The BeT is trained on the 49 trajectories, then we run BeTAIL on the same 49 trajectories and the same environment as the demonstrations. BCAIL follows the same training scheme ($\alpha = 0.05$). AIL is trained on the same 49 trajectories.

Fig. 3d evaluates each agent during training. BeTAIL outperforms all other methods and rapidly learns a policy that can consistently finish a full lap and achieve the lowest lap time. AIL eventually learns a policy that navigates the track and achieves a low lap time; however, even towards the end of the training, AIL is less consistent, and multiple cars may fail to finish a full lap (higher standard deviation in the top of Fig. 3d). The other baselines perform poorly on this task, so they are not tested on the other, more difficult challenges. A residual BC policy (BCAIL) worsens performance, since the BC policy performs poorly in the online environment. Impressively, the BeT finishes some laps even though it is trained exclusively on offline data; but, the lap time is poor compared to BeTAIL and AIL, which exploit online rollouts.

In Fig. 3g, BeTAIL(1.0) achieves a lower lap time than BeTAIL(0.05), since larger α allows the residual policy to apply more correction to the BeT base policy. However, BeTAIL(1.0) is prone to oscillate steering back and forth, as indicated by the higher deviation in the steering command between timesteps. We hypothesize that it is because the non-Markovian human-like behavior diminishes as α becomes larger. It is supported by the observation that the Markovian AIL policy also tends to have extreme changes in the steering command. Fig. 2 provides insight AIL’s extreme steering rates. We deliberately initialize the race cars at a lower speed than the human to test the robustness of AIL and BeTAIL agents. The heading of the AIL agent oscillates due to its tendency to steer back and forth, which causes the AIL agent to lose control and collide with the corner. Conversely, BeTAIL smoothly accelerates then brakes into the corner.

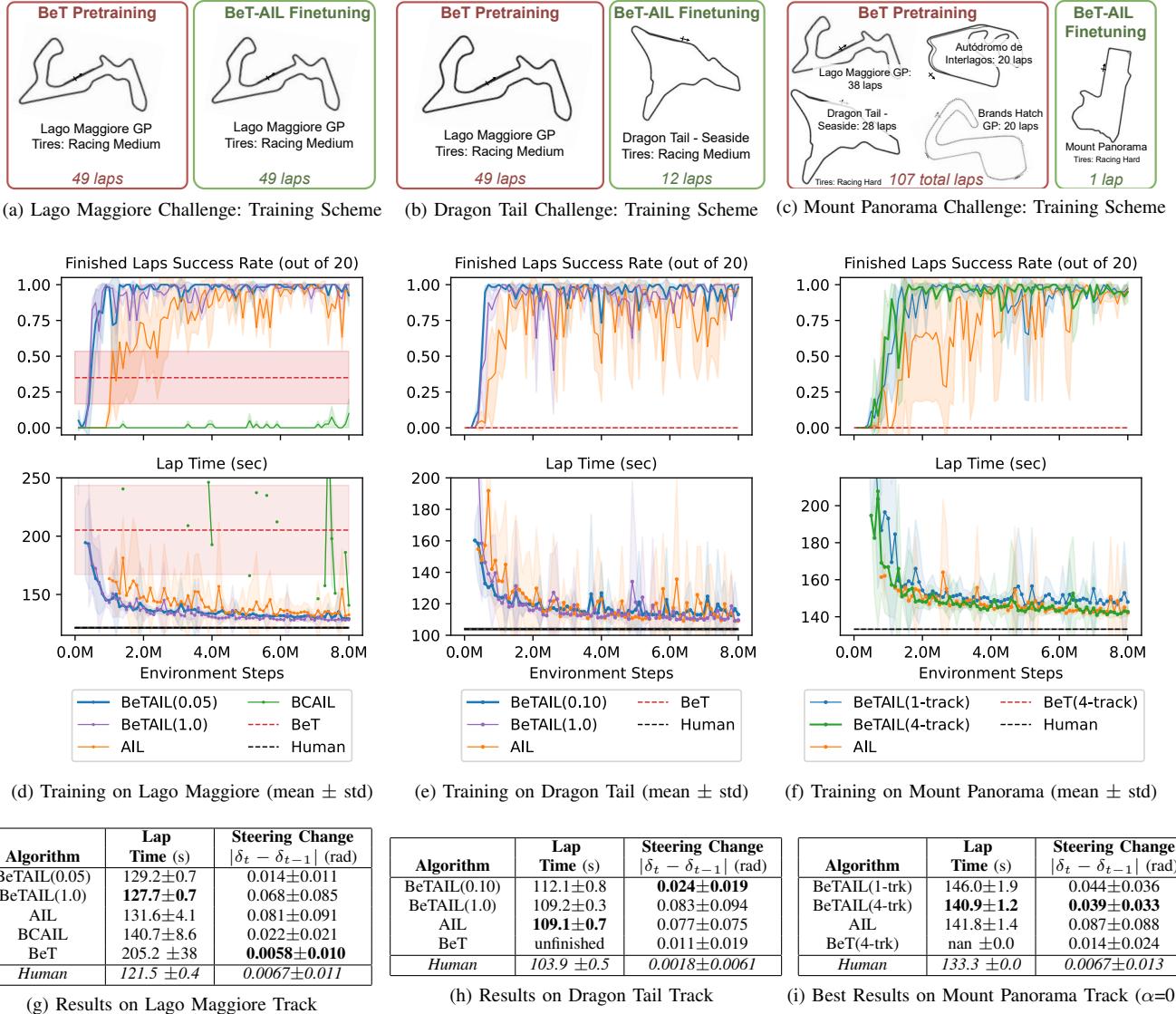


Fig. 3: Experimental results on three racing challenges. (a) Lago Maggiore challenges pretrains the BeT on the same demonstrations and downstream environments. (b) Dragon Tail transfers the BeT policy to a new track with BeTAIL finetuning. (c) The Mount Panorama challenge pretrains the BeT on a library of 4 tracks, and BeTAIL finetunes on an unseen track. (d)-(f) evaluation of mean (std) success rate to finish laps and mean (std) of lap times. (g)-(i) Best policy’s mean \pm std lap time and change in steering from previous time step. (8M steps \approx 25 hours w/ 20 cars collecting data)

B. Dragon Tail Challenge: Transferring the BeT policy to another track with BeTAIL

The second challenge tests if BeTAIL can fine-tune the BeT when the downstream environment is different from the BeT demonstrations (Fig. 3b). The BeT from the Lago Maggiore Challenge is used as the base policy; however, the downstream environment is a different track (Dragon Trail), which has 12 demonstration laps for AIL training. The residual policy is allowed to be larger, BeTAIL(0.10), since the downstream environment is different than the BeT pretraining dataset. The vehicle dynamics are unchanged.

The results for training and evaluation on the Dragon Tail track are given in Fig. 3e/h. Again, BeTAIL employs the BeT to guide policy learning and quickly learns to navigate the

track at a high speed. Additionally, small α ensures that non-Markovian human behavior is preserved, resulting in smooth steering. Conversely, AIL learn policies that are capable of achieving low lap times; however, they exhibit undesirable rapid changes in steering and are significantly more prone to fail to finish a lap (top in Fig. 3e). The pretrained BeT, which was trained from demonstrations on a different track, is unable to complete any laps.

C. Mount Panorama Challenge: Learning a multi-track BeT policy and solving an unseen track with BeTAIL

Finally, a BeT policy is trained on a library of trajectories on four different tracks: Lago Maggiore GP (38 laps), Autodromo de Interlagos (20 laps), Dragon Tail - Seaside (28 laps), and

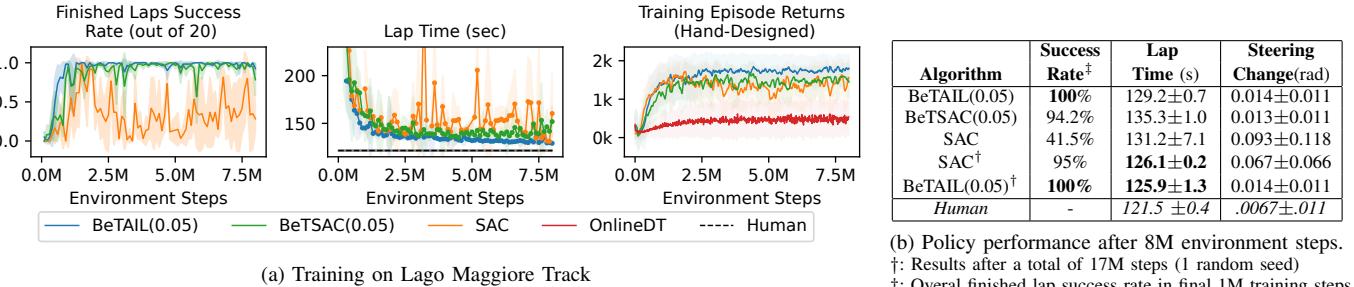


Fig. 4: Ablation study on Lago Maggiore (Fig. 3a). **SAC** trains a Markov policy, replacing the AIL reward with the reward in (9). **BeTSAC(0.05)** replaces the AIL residual policy finetuning step with SAC finetuning using (9). 3 seeds unless noted.

Brands Hatch GP (20 laps). For BeTAIL fine-tuning, BeTAIL is trained on a single demonstration on the Mount Panorama Circuit. Due to the availability of trajectories, there is a slight change in vehicle dynamics from the first two challenges due to the use of different tires (Racing Hard); the vehicle dynamics in downstream training and evaluation employ the same Racing Hard tires as the library of trajectories. The Mount Panorama circuit has more complex course geometry with hills and sharp banked turns than the pretraining tracks; thus, the residual policy is larger to correct for errors in the offline BeT. In Fig. 3f, **BeTAIL(4-track)** indicates the BeT is trained on the library of trajectories (Fig. 3c) with $\alpha = 0.2$. As an ablation, we compare **BeTAIL(1-track)** with $\alpha = 0.2$, where the pretrained BeT is the one used in Fig. 3a/b with Racing Medium tires, which have a lower coefficient of friction than the Racing Hard tires in the downstream environment.

As with previous challenges, both BeTAIL(1-track) and BeTAIL(4-track) navigate the track with less environment interactions than AIL alone. BeTAIL(4-track) achieves faster lap times and smoother steering than AIL, indicating that BeT pre-training on a library of tracks can assist learning on new tracks. BeTAIL(1-track) exhibits a slight performance drop when transferring between different vehicle dynamics compared to BeTAIL(4-track). However, BeTAIL(1-track) still accelerates training to achieve a high success rate faster than AIL alone. In our website’s videos, all agents struggle at the beginning of the track, but BeTAIL and AIL navigate the track rapidly despite the complicated track geometry and hills. AIL exhibits undesirable shaking behavior, but BeTAIL smoothly navigates with the highest speed.

D. Reinforcement Learning Ablation on Lago Maggiore

BeTAIL is a reward-free IL algorithm. In this subsection, we further compare it against several RL baselines that have access to explicit environment rewards. We employ a reward identical to [3]:

$$R(s_t, a_t) \doteq (cp_t - cp_{t-1}) - c_w \|\mathbf{v}_t\|^2 \mathbb{I}_t^{o.c.}, \quad (9)$$

where the first term, $(cp_t - cp_{t-1})$, represents the course progress along the track centerline, and the second term is a penalty given by the indicator flag $\mathbb{I}_t^{o.c.}$ when the vehicle goes off course (outside track boundaries). The weight c_w is set

identical to [3]. **SAC** is trained on (9) with soft actor critic [38] with identical network architecture as the AIL baseline. The ablation **BeTSAC** is identical to BeTAIL except the residual policy is trained with SAC on (9) instead of AIL.

The Online Decision Transformer (**OnlineDT**) [13], includes a return-to-go (rtg) conditioning in addition to the state-action inputs to the BeT. Since the DT includes rtg, collecting online rollouts improves performance [13]. Our OnlineDT baseline is pretrained with the same data and number of iterations as the BeT, and then online rollouts are collected and labeled with (9). Following [13], the rollout rtg is 5000, which is double the maximum expected rewards for the 500-step training rollouts and 500-step offline dataset trajectory segments. Prior work does not examine rtg with evaluation episodes that are longer than training rollouts [13]. Thus, we only evaluate OnlineDT in the 500-step training episodes and do not test 5000-step lap time evaluations.

Fig. 4 shows that BeTAIL and BeTSAC reduce the number of environment steps required for training in comparison to SAC. BeTSAC performs worse than BeTAIL, and SAC actually performs worse than AIL (see Fig. 3d) within 8M environment steps; we hypothesize that it is because the AIL objective provides better guidance than the heuristically shaped dense rewards. OnlineDT struggles to maximize rewards even in training rollouts, likely because rtg conditioning is insufficient for stochastic environments where rewards are not guaranteed [32]. The results call for further exploration of best practices for online training of Transformer architectures [35].

Fig. 4b provides a summary of the results at the end of training. Longer training (17M environment steps/ ~ 2.5 days wall-clock) improves the SAC lap time near that of BeTAIL. However, SAC is less stable, as shown by an average of the lap success rate in the final 1M steps of training. BeTAIL(0.05) finishes every evaluation lap after only 8M training steps, whereas SAC still fails to always finish every lap after 17M training steps. Previous SAC successes required custom rewards, custom training regimes and starting positions, and extensive environment interactions lasting multiple days for training on multiple devices [2], [3]. Conversely, BeTAIL exploits the pretrained BeT (using only a couple hours of offline data) to accelerate training and learn high-performing policies from demonstrations.

VII. CONCLUSION

We proposed BeTAIL, a Behavior Transformer augmented with residual Adversarial Imitation Learning. BeTAIL leverages sequence modeling and online imitation learning to learn racing policies from human demonstrations. In three experiments in the high-fidelity racing simulator Gran Turismo Sport, we show that BeTAIL can leverage both demonstrations on a single track and a library of tracks to accelerate downstream AIL on unseen tracks. BeTAIL policies exhibit smoother steering and reliably complete racing laps. In a small ablation, we show that BeTAIL can accelerate learning under minor dynamics shifts when the BeT is trained with different tires and tracks than the downstream residual AIL.

Limitations and Future Work: BeTAIL employs separate pre-trained BeT and residual AIL policy networks. The residual policy and the discriminator network are Markovian, rather than exploiting the sequence modeling in the BeT. Future work could explore alternate theoretical frameworks that improve the BeT action predictions themselves. Also, sequence modeling could be introduced into AIL frameworks to match the policy's and experts' trajectory sequences instead of single-step state-action occupancy measures. Finally, there is still a small gap between BeTAIL's lap times and the lap times achieved in the expert demonstrations. Future work will explore how to narrow this gap with improved formulations or longer training regimens.

REFERENCES

- [1] J. Betz, H. Zheng, A. Liniger, U. Rosolia, *et al.*, “Autonomous vehicles on the edge: A survey on autonomous vehicle racing,” *Open Journal Intell. Trans. Syst.*, vol. 3, pp. 458–488, 2022.
- [2] P. R. Wurman, S. Barrett, K. Kawamoto, J. MacGlashan, *et al.*, “Outracing champion gran turismo drivers with deep reinforcement learning,” *Nature*, vol. 602, no. 7896, pp. 223–228, 2022.
- [3] F. Fuchs, Y. Song, E. Kaufmann, D. Scaramuzza, and P. Dürr, “Superhuman performance in gran turismo sport using deep reinforcement learning,” *Robot. Autom. Letters*, vol. 6, no. 3, pp. 4257–4264, 2021.
- [4] S. Booth, W. B. Knox, J. Shah, S. Niekuum, *et al.*, “The perils of trial-and-error reward design: Mismatch through overfitting and invalid task specifications,” *Proc. AAAI Conf. Artificial Intell.*, vol. 37, no. 5, pp. 5920–5929, Jun. 2023.
- [5] M. Zare, P. M. Kebria, A. Khosravi, and S. Nahavandi, “A survey of imitation learning: Algorithms, recent developments, and challenges,” *arXiv:2309.02473*, 2023.
- [6] E. L. Zhu, F. L. Busch, J. Johnson, and F. Borrelli, “A gaussian process model for opponent prediction in autonomous racing,” in *Int. Conf. Intell. Robots Syst. (IROS)*. IEEE, 2023, pp. 8186–8191.
- [7] M. Orsini, A. Raichuk, L. Hussienot, D. Vincent, *et al.*, “What matters for adversarial imitation learning?” *Adv. Neural Inform. Processing Syst.*, vol. 34, pp. 14 656–14 668, 2021.
- [8] L. Chen, K. Lu, A. Rajeswaran, K. Lee, *et al.*, “Decision transformer: Reinforcement learning via sequence modeling,” *Adv. Neural Inform. Processing Syst.*, vol. 34, pp. 15 084–15 097, 2021.
- [9] M. Janner, Q. Li, and S. Levine, “Offline reinforcement learning as one big sequence modeling problem,” *Adv. Neural Inform. Processing Syst.*, vol. 34, pp. 1273–1286, 2021.
- [10] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, *et al.*, “Improving language understanding by generative pre-training.” 2018.
- [11] J. Li, X. Liu, B. Zhu, J. Jiao, *et al.*, “Guided online distillation: Promoting safe reinforcement learning by offline demonstration,” *arXiv:2309.09408*, 2023.
- [12] N. M. Shafullah, Z. Cui, A. A. Altanzaya, and L. Pinto, “Behavior transformers: Cloning k modes with one stone,” *Adv. Neural Inform. Processing Syst.*, vol. 35, pp. 22 955–22 968, 2022.
- [13] Q. Zheng, A. Zhang, and A. Grover, “Online decision transformer,” in *Int. Conf. Machine Learning*. PMLR, 2022, pp. 27 042–27 059.
- [14] J. Ho and S. Ermon, “Generative adversarial imitation learning,” *Adv. Neural Inform. Processing Syst.*, vol. 29, 2016.
- [15] T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling, “Residual policy learning,” *arXiv:1812.06298*, 2018.
- [16] K. Brown, K. Driggs-Campbell, and M. J. Kochenderfer, “A taxonomy and review of algorithms for modeling and predicting human driver behavior,” *arXiv:2006.08832*, 2020.
- [17] A. Kuefler, J. Morton, T. Wheeler, and M. Kochenderfer, “Imitating driver behavior with generative adversarial networks,” in *Intell. Vehicles Sym. (IV)*. IEEE, 2017, pp. 204–211.
- [18] Y. Li, J. Song, and S. Ermon, “Infogail: Interpretable imitation learning from visual demonstrations,” *Adv. Neural Inform. Processing Syst.*, vol. 30, 2017.
- [19] R. Bhattacharyya, B. Wulfe, D. J. Phillips, A. Kuefler, *et al.*, “Modeling human driving behavior through generative adversarial imitation learning,” *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 3, pp. 2874–2887, 2022.
- [20] T. Fernando, S. Denman, S. Sridharan, and C. Fookes, “Learning temporal strategic relationships using generative adversarial imitation learning,” *arXiv:1805.04969*, 2018.
- [21] A. Sharma, M. Sharma, N. Rhinehart, and K. M. Kitani, “Directed-info gail: Learning hierarchical policies from unsegmented demonstrations using directed information,” *arXiv:1810.01266*, 2018.
- [22] G. Lee, D. Kim, W. Oh, K. Lee, and S. Oh, “Mixgail: Autonomous driving using demonstrations with mixed qualities,” in *IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*. IEEE, 2020, pp. 5425–5430.
- [23] Y. Song, H. Lin, E. Kaufmann, P. Dürr, and D. Scaramuzza, “Autonomous overtaking in gran turismo sport using curriculum reinforcement learning,” in *Int. Conf. Robot. Autom. (ICRA)*. IEEE, 2021, pp. 9403–9409.
- [24] V. Bajaj, G. Sharon, and P. Stone, “Task phasing: Automated curriculum learning from demonstrations,” in *Int. Conf. Automated Planning Scheduling*, vol. 33, no. 1, 2023, pp. 542–550.
- [25] Z. Xue, Z. Peng, Q. Li, Z. Liu, and B. Zhou, “Guarded policy optimization with imperfect online demonstrations,” in *Int. Conf. Learning Representations*, 2022.
- [26] X.-H. Liu, F. Xu, X. Zhang, T. Liu, *et al.*, “How to guide your learner: Imitation learning with active adaptive expert involvement,” in *Pro. Int. Conf. Autonomous Agents Multiagent Syst.*, 2023, pp. 1276–1284.
- [27] S. Levine and V. Koltun, “Guided policy search,” in *Pro. Int. Conf. Machine Learning*, S. Dasgupta and D. McAllester, Eds., vol. 28, no. 3. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 1–9.
- [28] R. Zhang, J. Hou, G. Chen, Z. Li, *et al.*, “Residual policy learning facilitates efficient model-free autonomous racing,” *IEEE Robot. Autom. Letters*, vol. 7, no. 4, pp. 11 625–11 632, 2022.
- [29] T. Johannink, S. Bahl, A. Nair, J. Luo, *et al.*, “Residual reinforcement learning for robot control,” in *Int. Conf. Robot. Autom. (ICRA)*. IEEE, 2019, pp. 6023–6029.
- [30] K. Rana, M. Xu, B. Tidd, M. Milford, and N. Sünderhauf, “Residual skill policies: Learning an adaptable skill-based action space for reinforcement learning for robotics,” in *Conf. Robot Learning*. PMLR, 2023, pp. 2095–2104.
- [31] J. Won, D. Gopinath, and J. Hodgins, “Physics-based character controllers using conditional vae’s,” *Trans. Graphics (TOG)*, vol. 41, no. 4, pp. 1–12, 2022.
- [32] C. Gao, C. Wu, M. Cao, R. Kong, *et al.*, “Act: Empowering decision transformer with dynamic programming via advantage conditioning,” *arXiv:2309.05915*, 2023.
- [33] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, *et al.*, “What matters in learning from offline human demonstrations for robot manipulation,” *arXiv:2108.03298*, 2021.
- [34] T. Brown, B. Mann, N. Ryder, M. Subbiah, *et al.*, “Language models are few-shot learners,” *Adv. Neural Inform. Processing Syst.*, vol. 33, pp. 1877–1901, 2020.
- [35] W. Li, H. Luo, Z. Lin, C. Zhang, *et al.*, “A survey on transformers in reinforcement learning,” *Trans. Machine Learning Research*, 2023.
- [36] R. Trumpp, D. Hoornaert, and M. Caccamo, “Residual policy learning for vehicle control of autonomous racing cars,” *arXiv:2302.07035*, 2023.
- [37] I. Kostrikov, K. K. Agrawal, D. Dwibedi, S. Levine, and J. Tompson, “Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning,” *arXiv:1809.02925*, 2018.
- [38] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” in *Int. Conf. Machine Learning*. PMLR, 2018, pp. 1861–1870.