GAN-MPC: Training Model Predictive Controllers with Parameterized Cost Functions using Demonstrations from Non-identical Experts

Returaj Burnwal[+†], Anirban Santara[*], Nirav P. Bhatt[+†],
Balaraman Ravindran[+†] and Gaurav Aggarwal[*]
[+]Indian Institute of Technology, Madras
[†]Robert Bosch Centre for Data Science and AI
[*]Google Research

Abstract

Model predictive control (MPC) is a popular approach for trajectory optimization in practical robotics applications due to their guarantees on safety, optimality, generalizability, interpretability, and explainability. Traditional MPC needs a hand-crafted cost function for trajectory optimization. However, some behaviors are complex and hand-crafting is difficult and error-prone. A special class of MPC policies called Learnable-MPC addresses this difficulty using imitation learning from expert demonstrations. However, they require the demonstrator and the imitator agents to have identical state-action spaces and transition dynamics which is hard to satisfy in many practical applications of robotics. In this paper, we address this practical problem through a novel approach that uses a generative adversarial network (GAN) to match state-trajectory distributions of the demonstrator and the imitator. We evaluate our approach on a variety of simulated robotics tasks of DeepMind Control suite and demonstrate the efficacy of our approach at learning the demonstrator's behavior without having to copy their actions.

1 Introduction

Large-scale deployment of robots in real-world human-centric environments is faced with the challenges of safety, social compatibility and robustness to unforeseen changes in the environment [1]. Model predictive control (MPC) [2, 3, 4, 5] is a popular approach for trajectory optimization in robotics. MPC policies can optimize trajectory parameters under kinodynamic and safety constraints, and they provide guarantees on safety, optimality, and generalizability. However, it is difficult to hand-craft an MPC objective function for complex behaviors. *Learnable MPC* [6, 7, 8, 9, 10, 11, 1] addresses this difficulty using imitation learning. Learnable MPC policies use a parameterized objective function that can be trained from expert demonstrations. The learnable parameters also allow it to easily adapt to a wide variety of robot-environment situations. However, even state-of-the-art learnable MPC formulations require the demonstrator and the imitator to be identical. This is a limitation because robots in real-world applications may have different dynamics. Changes to a robot's dynamics can be caused by internal changes, such as mechanical faults [12], dropping battery charge-level [13], and external changes, such as changes in the operating environment, e.g., surface friction [14], or the robot's task, e.g., increased load [13]. In this paper, we address the practical problem of training Learnable-MPC policies when the demonstrator and the imitator do not share the same dynamics and their state spaces only have a partial overlap. Our proposed method uses a generative adversarial network (GAN) to match the state-trajectory distributions of the demonstrator and the imitator by minimizing the Jensen Shannon (JS) divergence [15]. The GAN consists of two networks: a generator and a discriminator. The generator is a neural network modeling the learnable cost function. This, along with the engineered cost is minimized by the imitator to produce trajectories. The discriminator is responsible for distinguishing between state trajectories from the demonstrator and the imitator. At Nash equilibrium [16], the state-trajectory distributions of the demonstrator and the imitator would be identical. Empirical evaluation on three continuous control tasks of DeepMind Control Suite [17] shows that our method is effective in mimicking complex behaviors even when the dynamics of the demonstrator and the imitator are widely different.

2 Problem Statement

Imitation learning [18] involves two agents - demonstrator (also referred to as the "expert") $\mathbf{D}$ and the imitator $\mathbf{I}$. Let $\mathcal{M}^{\mathbf{D}} = (S^{\mathbf{D}}, A^{\mathbf{D}}, T^{\mathbf{D}}, \rho^{\mathbf{D}})$ and $\mathcal{M}^{\mathbf{I}} = (S^{\mathbf{I}}, A^{\mathbf{I}}, T^{\mathbf{I}}, \rho^{\mathbf{I}})$ be the Markov Decision Processes (MDPs) [19] associated with the $\mathbf{D}$ and $\mathbf{I}$ respectively. Equation 1 describes the optimization problem solved by MPC.

$$\begin{aligned} \mathbf{a}^*_{1:H} &= arg\min\nolimits_{\mathbf{a}_{1:H-1}} J(s_t, a_{1:H-1}) \\ &= arg\min\nolimits_{\mathbf{a}_{1:H-1}} \sum_{t=1}^{H-1} C_{stg}(s_t, a_t, t) + \gamma C_{term}(s_H) \\ s.t. \quad & \forall t,\ \ s_{t+1} = \tilde{T}(s_t, a_t),\ g(s_t, a_t) = 0,\ h(s_t, a_t) \leq 0 \end{aligned} \tag{1}$$

H is the planning horizon of the MPC. $C_{stg} : S \times A \to \mathbb{R}$ is the staging cost that applies to each step of the plan and $C_{term} : S \to \mathbb{R}$ is the terminal cost that applies only to the final state. $g : S \times A \to \mathbb{R}$ and $h : S \times A \to \mathbb{R}$ are equality and inequality constraints on the solution. γ is a hyperparameter that controls the relative weightage of the staging and the terminal costs. $\tilde{T}$ is a local model of the transition dynamics T around the initial control guess. At every step of planning, the MPC plans a trajectory $\mathbf{a}^*_{1:H-1}$ of length H that minimizes the objective in Equation 1. To address the inevitability of modeling error in the estimation of $\tilde{T}$, MPC only executes the first action a^*_1 and updates $\tilde{T}$ with the observed outcome. We denote an MPC policy by $\pi^{MPC} : S \to A$ where $\pi^{MPC}(s_t) = a^*_1$. This planning algorithm is repeated for every step of the agent's trajectory. Motivated by real world applications in robotics and accessibility, we study the problem of imitation learning of Learnable MPC policies when the demonstrator and the imitator do not share the same dynamics - $T^{\mathbf{D}} \neq T^{\mathbf{I}}$. Our method can also be applied to settings where the state and action spaces do not overlap completely, by considering only the overlapping state and action variables.

2.1 Challenges

MPC requires a model of the transition dynamics for planning. This is challenging in real world complex continuous control tasks with large state-action spaces. Some parts of the state-action space are difficult to reach and hence difficult to collect data from. Also, parts of the state-action space are often inaccessible due to hard kinodynamic constraints. Neural networks provide an efficient way of modeling highly non-linear functions over large state-action spaces. However, they find it hard to model the constraints and end up halucinating in the inaccessible areas, often leading to infeasible solutions. MPC solvers like iLQR can be highly sensitive to the "initial control guess" in complex non-linear dynamical systems. The challenge is to predict an $a^g_{0:H-1}$ close to the optimal solution a^*_0. The terminal cost C_{term} is used to measure how close the agent would get to a "target" state at the end of the planning horizon H. For dynamic tasks like `Cheetah Run` the target state is different for each time step and making it difficult to calculate C_{term}.

3 Proposed Method: GAN-MPC

The proposed appraoch uses the GAN framework [20] that consists of a generator and a discriminator. Given a set of expert demonstrations, the task of the discriminator is to learn an accurate binary classifier to tell apart expert demonstrations from other trajectories. The task of the generator is to produce samples that are indistinguishable from demonstrator's trajectories. Our generator is the Learnable MPC policy $\pi^{MPC}(\cdot|\Phi^{gen})$ of $\mathbf{I}$ along with a model of the transition dynamics $\tilde{T}^{\mathbf{I}}$. Φ^{gen} is the set of learnable parameters of the terminal cost function. Given a demonstrated trajectory $\tau^{\mathbf{D}}_s = (s^{\mathbf{D}}_0, s^{\mathbf{D}}_1, s^{\mathbf{D}}_2, \ldots) \in \mathcal{X}^{\mathbf{D}}_s$, a generator rollout $\tau^{\mathbf{I},g} = (s^{\mathbf{I},g}_0, a^{\mathbf{I},g}_0, s^{\mathbf{I},g}_1, a^{\mathbf{I},g}_1, s^{\mathbf{I},g}_2, a^{\mathbf{I},g}_2, \ldots, s^{\mathbf{I},g}_{P-1})$ of maximum length P (a hyper parameter) is created by starting from the same initial state $s^{\mathbf{I},g}_0 = s^{\mathbf{D}}_0$, solving for actions using the MPC policy $a^{\mathbf{I},g}_t = \pi^{MPC}(s^{\mathbf{I},g}_t)$ and the next state from the transition dynamics model $s^{\mathbf{I},g}_{t+1} = \tilde{T}^{\mathbf{I},g}(s^{\mathbf{I},g}_t, a^{\mathbf{I},g}_t)$. We denote the state trajectory distribution of the generator rollouts by $\mathcal{G}_s(\cdot|\Phi^{gen}, \Theta^{\mathbf{I}})$. The discriminator $Q(\cdot|\Phi^{disc})$ is modelled using an LSTM network with parameters Φ^{disc}.

The performance of an MPC policy is strongly dependent on the accuracy of transition dynamics model T. As noted in Section 2.1 learning a model of $T^{\mathbf{I}}$ can be challenging in large state-action spaces. The dynamics function must be trained on (s_t, a_t, s_{t+1}) transitions collected by the agent while interacting with the environment. In order to model the function accurately in the regions of the state-action space traversed during the execution of the target task, enough data must be collected from those regions. This is not a big issue when $\mathbf{D}$ and $\mathbf{I}$ are identical as the demonstrated trajectories $\mathcal{X}^{\mathbf{D}}$ can be used for training $T^{\mathbf{I}}$. However, in our case, getting $\mathbf{I}$ to the desired regions of

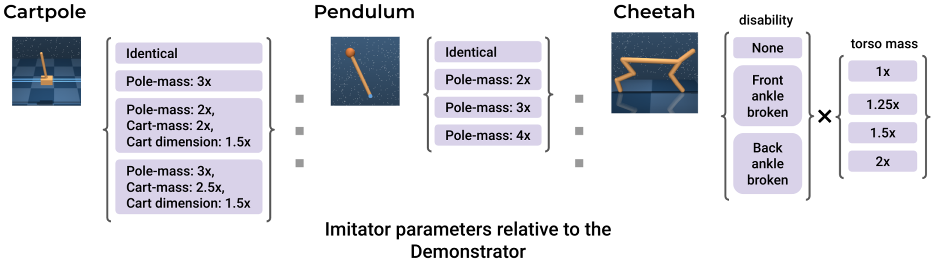


Figure 1: Physical properties of the imitators relative to the demonstrators in our experiments. We have 4 imitators each for `Cartpole-Balance` and `Pendulum-Standup`. In case of `Cheetah-Run`, we have 12 imitators with different levels of disability and different torso-masses as denoted by the set product "×" in the figure.

the state-action space can be as hard as learning the policy. We address this challenge by by pre-training $T^{\mathbf{I}}$ on $\mathcal{X}^{\mathbf{D}}$ for a small number of epochs under the assumption that the demonstrator and the imitator dynamics have some degree of similarity. We continue to update the dynamics model in each training iteration with transitions recorded from physical interaction of $\mathbf{I}$ with the environment with π^{MPC}. We use the popular iLQR solver in our experiments. As noted in Section 2.1, the performance is a strong function of the initial control guess $a^{\mathbf{I},g}_{0:H-1}$. We again make the assumption that the demonstrator and imitator dynamics have some degree of similarity. We train a behavior cloning policy $\pi^{BC}_{\chi} : S^{\mathbf{I}} \rightarrow A^{\mathbf{I}}$ with parameters χ on $\mathcal{X}^{\mathbf{D}}$. At each iteration of iLQR, we set $a^{\mathbf{I},g}_t = \pi^{BC}(\tilde{s}^{\mathbf{I}}_t)$. The terminal component of the MPC cost function C_{term} is intended to estimate how far the agent would be from the target state at the end of the planning horizon. In dynamic tasks like `Cheetah Run`, the target state is not singular making it difficult to specify C_{term}. With a motivation to set as target state as somewhere the expert would be in the next time step, we train a neural network model $\mathcal{N}_{\Psi} : S^{\mathbf{D}} \rightarrow S^{\mathbf{D}}$ with trainable parameters Ψ on $\mathcal{X}^{\mathbf{D}}$ to predict the next state $s^{\mathbf{D}}_{t+1}$ given the current state $s^{\mathbf{D}}_t$.

Our algorithm, GAN-MPC, starts by pre-training the dynamics model of the imitator on $\mathcal{D}$ for a small number of epochs. In the main training loop, in the first step, we let the imitator interact with the environment for K time steps and use this data to update the dynamics model by running a small number of epochs N^{dyn} of training. Next, the discriminator network is trained on $\mathcal{D}^s$ and the imitator's state trajectories. In the final step, the learnable parameters of the MPC policy and the relative weight of the engineered and learnable cost components are updated slowly [21].

4 Experiments

Our experimental study aims to understand whether GAN-MPC can learn an expert's skills by trying to visit the same sequence of states and planning an appropriate sequence of actions, even though the imitator's actions may be different from the expert's due to differences in dynamics. We evaluate the efficacy of GAN-MPC on three continuous control tasks from the DeepMind Control suite: `CartPole-Balance`, `Pendulum-Standup`, and Cheetah Run. For each task we train a SAC agent for sampling demonstrator trajectories. We choose a set of imitator agents that have similar morphology as the demonstrators but different physical properties as described in Figure 1. We compare the performance of our proposed algorithm (**GAN-MPC**) with Behavioral Cloning (**BC**) and two Learnable-MPC formulation that minimizes the L2 distance between the demonstrator and imitator trajectories: a) **L2-MPC-SA** that matches state-action trajectories and b) **L2-MPC-S** that matches state-only trajectories of the demonstrator and the imitator. In many practical applications, the entire state space of the demonstrator may not be observable or the state spaces of the demonstrator and the imitator may only overlap partially. We study this case also in the Cheetah Run task environment.

In all experiments, a training set of 50 trajectories is collected from the demonstrator. **L2-MPC-SA**, **L2-MPC-S** and **GAN-MPC** imitators are allowed to interact with the environment for a total of 5000 steps for `Cartpole-Balance` and `Pendulum-Swingup`; and 10000 steps for `Cheetah-Run`. The performance of each agent is measured by rolling out 50 trajectories with different random seeds and computing the average trajectory reward R^{τ}. We measure the performance of the imitators in terms of average trajectory reward relative to the demonstrator, $\tilde{R}^{\tau} = \frac{R^{\tau}_{imitator}}{R^{\tau}_{demonstrator}}$. Figures 2, 3 and 4 provide a summary of the results. The bars represent means and the whiskers represent standard deviations. We observe that GAN-MPC outperforms or matches the baselines in most of the settings. We also

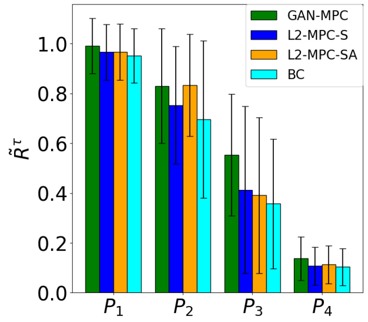


Figure 2: Results of the `Pendulum-Swingup` experiment. The imitators are denoted by P_x where P stands for *pole mass* and $x = P_{imitator}/P_{demonstrator}$.

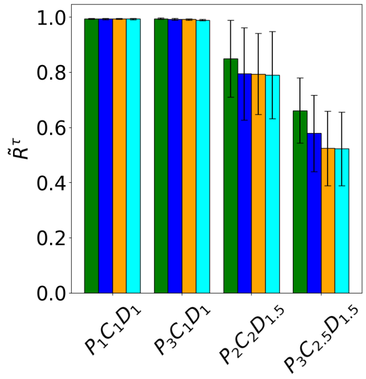
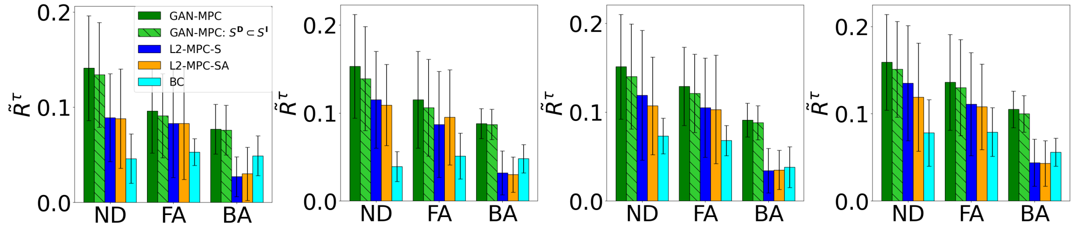


Figure 3: Results of the `Cartpole-Balance` experiment. The imitators are denoted by $P_xC_yD_z$ where P, C and D stand for *pole mass*, *cart mass* and *cart dimension*, respectively. The subscripts - x, y and z - denote ratios relative to the demonstrator, e.g. $x = P_{imitator}/P_{demonstrator}$. The legend of Figure 2 has been followed.



(a) Same torso-mass (b) 1.25× torso-mass (c) 1.5× torso-mass (d) 2× torso-mass

Figure 4: Results of the `Cheetah-Run` experiment. The captions of the sub-figures mention "torso-mass" of the imitators relative to the demonstrator. As described in Section 4 and Figure 1, we have three categories of imitators in terms of disability - No Disability (**ND**), Front Ankle broken (**FA**) and Back Ankle broken (**BA**). All the agents except " GAN-MPC: $S^{\mathbf{D}} \subset S^{\mathbf{I}}$ " are trained on the same set of demonstrations $\mathcal{X}_s^{\mathbf{D}}$. As described in Section 4, " GAN-MPC: $S^{\mathbf{D}} \subset S^{\mathbf{I}}$ " is trained on $\mathcal{X}_s^{\mathbf{D}}$ but only a subset of the state variables are exposed.

observe that the performance of GAN-MPC gracefully degrades (like most of the baselines) as the dynamics of the imitator becomes more and more different from the demonstrator. In our experiments on `Cheetah-Run` we observe that the disabled imitators, in their quest to learn the fit demonstrator's skills, learn alternative strategies to work around their disabilities. This establishes GAN-MPC as a viable step towards achieving the goal of learning skills from non-identical experts without having to copy their actions. In Figure 4, we also observe that under partial observability of the demonstrator's state space the GAN-MPC agents (" GAN-MPC: $S^{\mathbf{D}} \subset S^{\mathbf{I}}$ ") are able to learn the desired behavior and outperform the baselines that have access to the full state observations. This shows the viability of GAN-MPC as a method to learn skills from experts with non-identical dynamics and partial observability of their state spaces.

5 Conclusion

In this paper, we study imitation learning of MPC policies with parameterised cost functions. We consider the practical challenges of mismatch in the dynamics of the demonstrator and the imitator agents and partial observability of the state space of the demonstrator. We propose a novel approach called GAN-MPC that minimizes the statistical divergence between state-trajectories of the demonstator and the imitator using the GAN framework. Experiments on continuous control tasks of the DeepMind Control suite demonstrate the viability of the proposed method. The GAN-MPC framework needs significantly fewer samples of real world interaction of the imitator compared to RL based methods and this makes it viable for real world applications.

References

[1] Xuesu Xiao, Tingnan Zhang, Krzysztof Choromanski, Edward Lee, Anthony Francis, Jake Varley, Stephen Tu, Sumeet Singh, Peng Xu, Fei Xia, et al. Learning model predictive controllers with real-time attention for real-world navigation. *arXiv preprint arXiv:2209.10780*, 2022.

[2] Manfred Morari, Carlos E Garcia, and David M Prett. Model predictive control: theory and practice. *IFAC Proceedings Volumes*, 21(4):1–12, 1988.

[3] Yang Wang and Stephen Boyd. Fast model predictive control using online optimization. *IEEE Transactions on control systems technology*, 18(2):267–278, 2009.

[4] Spyros Maniatopoulos, Dimitra Panagou, and Kostas J Kyriakopoulos. Model predictive control for the navigation of a nonholonomic vehicle with field-of-view constraints. In *2013 American control conference*, pages 3967–3972. IEEE, 2013.

[5] Thomas Fork, H Eric Tseng, and Francesco Borrelli. Models and predictive control for nonplanar vehicle navigation. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 749–754. IEEE, 2021.

[6] Rahul Shridhar and Douglas J Cooper. A tuning strategy for unconstrained siso model predictive control. *Industrial & Engineering Chemistry Research*, 36(3):729–746, 1997.

[7] Rahul Shridhar and Douglas J Cooper. A tuning strategy for unconstrained multivariable model predictive control. *Industrial & engineering chemistry research*, 37(10):4003–4016, 1998.

[8] Jorge L Garriga and Masoud Soroush. Model predictive control tuning methods: A review. *Industrial & Engineering Chemistry Research*, 49(8):3505–3515, 2010.

[9] William Edwards, Gao Tang, Giorgos Mamakoukas, Todd Murphey, and Kris Hauser. Automatic tuning for data-driven model predictive control. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7379–7385. IEEE, 2021.

[10] Andre Shigueo Yamashita, Antônio Carlos Zanin, and Darci Odloak. Tuning of model predictive control with multi-objective optimization. *Brazilian Journal of Chemical Engineering*, 33:333–346, 2016.

[11] Valarmathi Ramasamy, Rakesh Kumar Sidharthan, Ramkumar Kannan, and Guruprasath Muralidharan. Optimal tuning of model predictive controller weights using genetic algorithm with interactive decision tree for industrial cement kiln process. *Processes*, 7(12):938, 2019.

[12] Vandi Verma, Geoff Gordon, Reid Simmons, and Sebastian Thrun. Real-time fault diagnosis [robot fault diagnosis]. *IEEE Robotics & Automation Magazine*, 11(2):56–66, 2004.

[13] Marco Hutter, Christian Gehring, Andreas Lauber, Fabian Gunther, Carmine Dario Bellicoso, Vassilios Tsounis, Péter Fankhauser, Remo Diethelm, Samuel Bachmann, Michael Blösch, et al. Anymal-toward legged robots for harsh environments. *Advanced Robotics*, 31(17):918–931, 2017.

[14] Lei Hao, Roberto Pagani, Manuel Beschi, and Giovanni Legnani. Dynamic and friction parameters of an industrial robot: Identification, comparison and repetitiveness analysis. *Robotics*, 10(1):49, 2021.

[15] Gérard Biau, Benoît Cadre, Maxime Sangnier, and Ugo Tanielian. Some theoretical properties of gans. 2020.

[16] Farzan Farnia and Asuman Ozdaglar. Do gans always have nash equilibria? In *International Conference on Machine Learning*, pages 3029–3039. PMLR, 2020.

[17] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[18] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

[19] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

[20] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[21] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992.