2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Learning Multiple Initial Solutions to Optimization Problems

Elad Sharony[1], Heng Yang[2,3], Tong Che[2], Marco Pavone[2,4], Shie Mannor[1,2], and Peter Karkus[2]

***Abstract*— Sequentially solving similar optimization problems under strict runtime constraints is essential for many applications, such as robot control, autonomous driving, and portfolio management. The performance of local optimization methods in these settings is sensitive to the initial solution: poor initialization can lead to slow convergence or suboptimal solutions. To address this challenge, we propose learning to predict *multiple* diverse initial solutions given parameters that define the problem instance. We introduce two strategies for utilizing multiple initial solutions: (i) a single-optimizer approach, where the most promising initial solution is chosen using a selection function, and (ii) a multiple-optimizers approach, where several optimizers, potentially run in parallel, are each initialized with a different solution, with the best solution chosen afterward. Notably, by including a default initialization among predicted ones, the cost of the final output is guaranteed to be equal or lower than with the default initialization. We validate our method on three optimal control benchmark tasks: cart-pole, reacher, and autonomous driving, using different optimizers: DDP, MPPI, and iLQR. We find significant and consistent improvement with our method across all evaluation settings and demonstrate that it efficiently scales with the number of initial solutions required.**

I. Introduction

Many applications, ranging from trajectory optimization in robotics and autonomous driving to portfolio management in finance, require solving similar optimization problems sequentially under tight runtime constraints [1], [2], [3]. The performance of local optimizers in these contexts is often highly sensitive to the initial solution provided, where poor initialization can result in suboptimal solutions or failure to converge within the allowed time [4], [5]. The ability to consistently generate high-quality initial solutions is, therefore, essential for ensuring both performance and safety guarantees.

Conventional methods for selecting these initial solutions typically rely on heuristics or warm-starting, where the solution from a previously solved, related problem instance is reused. More recently, learning-based solutions have also been proposed, where neural networks are used to predict an initial solution. However, in more challenging cases, where the optimization landscape is highly non-convex or when consecutive problem instances rapidly change, predicting a single good initial solution is inherently difficult.

To this end, we propose *Learning Multiple Initial Solutions* (**MISO**) (Figure 1), in which we train a neural network to predict *multiple* initial solutions. Our approach facilitates two key settings: (i) a single-optimizer method, where a selection function leverages prior knowledge of the problem instance to identify the most promising initial solution, which is then supplied to the optimizer; and (ii) a multiple-optimizers method, where multiple initial solutions are generated jointly to support the execution of several optimizers, potentially running in parallel, with the best solution chosen afterward.

More specifically, our neural network receives a parameter vector that characterizes the problem instance and outputs K candidate initial solutions. The network is trained on a dataset of problem instances paired with (near-)optimal solutions and is evaluated on previously unseen instances. Crucially, the network is designed not only to predict *good* initial solutions—those close to the optimal—but also to ensure that these solutions are sufficiently diverse, potentially spanning all underlying modes of the problem in hand. To actively encourage this multimodality, we implement training strategies such as a winner-takes-all loss that penalizes only the candidate with the lowest loss, a dispersion-based loss term to promote dispersion among solutions, and a combination of both. Unlike prior work on learned warm starts that predict a single initialization, our key contribution is learning *multiple* initializations with explicit diversity-promoting losses that prevent mode collapse and systematically cover different modes of the solution space.

Notably, any existing initialization strategy can be combined with **MISO**, by simply including the existing initial solution among the predicted ones; and by design, **MISO** is guaranteed to be equal or better than the default initialization.

We evaluate **MISO** across three distinct local optimization algorithms applied to separate robot control tasks: First-order Box Differential Dynamic Programming (DDP), which utilizes first-order linearization for the cart-pole swing-up task; Model Predictive Path Integral (MPPI) control, a sampling-based method, for the reacher task; and the Iterative Linear Quadratic Regulator (iLQR), a trajectory optimization algorithm, for an autonomous driving task. Our results show that **MISO** significantly outperforms existing initialization methods that rely on heuristics, learn to predict a single initial solution or use ensembles of independently learned models.

In summary, our key contributions are as follows:

1) We present a novel framework for predicting *multiple* initial solutions for optimizers.
2) We introduce two distinct strategies for utilizing the predicted initial solutions: (i) *single-optimizer*, where the most promising solution is chosen based on a selection function, and (ii) *multiple-optimizers*, where multiple optimizers are initialized, potentially in parallel, with the best solution chosen afterward.
3) We design and implement specific training objectives to prevent mode collapse and ensure that the predicted

*Corresponding author: eladsharony@campus.technion.ac.il. [1]Technion, [2]NVIDIA Research, [3]Harvard University, [4]Stanford University.
Project page and code: https://esharony.me/projects/miso/

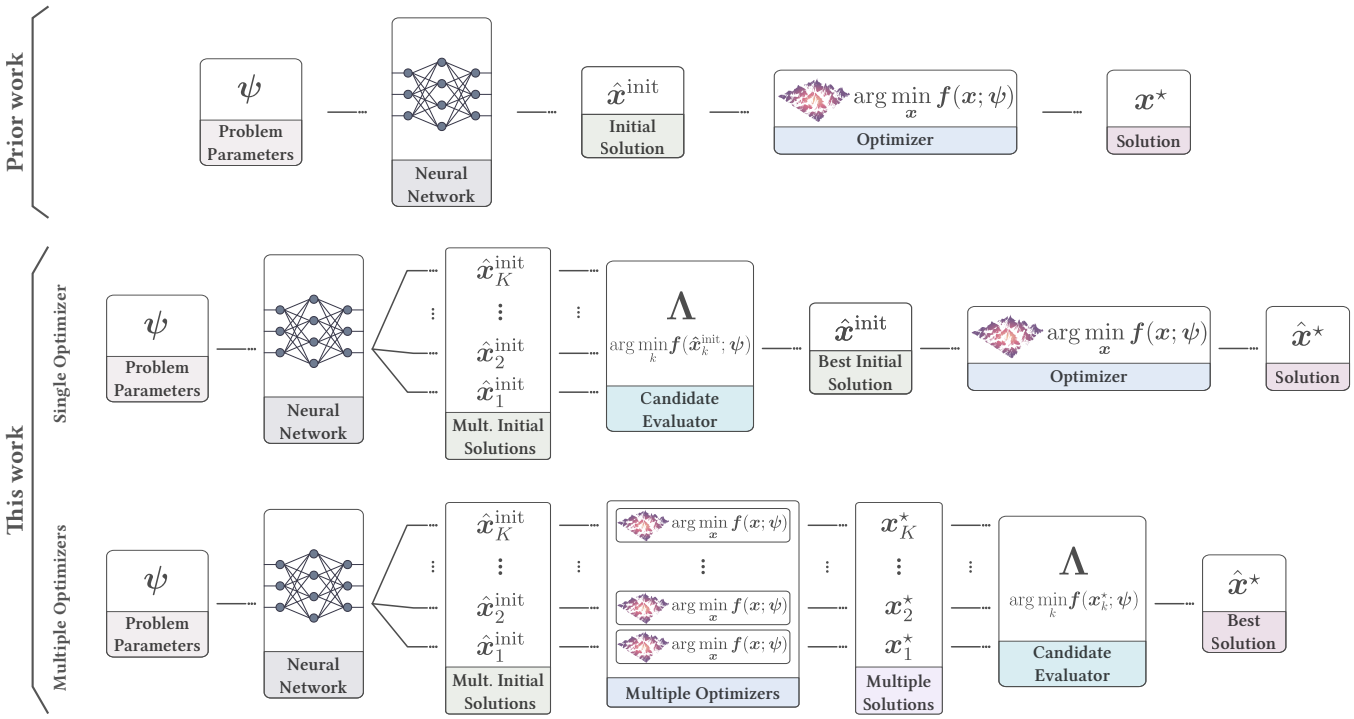


Fig. 1. As opposed to previous works **(top)** that predict a single initial solution, **MISO** trains a single neural network to predict *multiple* initial solutions. We use them to either initialize a single optimizer **(middle)** or jointly initialize multiple optimizers **(bottom)**.

solutions remain multimodal.

4) We apply our framework to three distinct sequential optimization tasks and perform extensive evaluation.

II. Related Work

Learning for optimization. Advancements in machine learning have introduced numerous learning-based approaches to optimization problems [6]. Early work by [7] replaced components of classical convex optimization algorithms with neural networks. More recent works aim to replace optimization methods entirely with end-to-end neural networks [8], [9] or generate new optimization algorithms [10] for specific classes of problems. Other works enhance optimization-based control algorithms [11], learn constraints [12], or learn objective functions and system dynamics [13], [14], [15], [16], [17], [18].

Learning initial solutions. Previous studies have proposed heuristic approaches to generate initial solutions for optimizers [19], [20]. More recently, learning-based methods for initializing optimizers have gained attention in various fields, aiming to enhance both computational efficiency and resulting solutions quality. In mixed-integer programming, neural networks have enhanced solver performance by predicting variable assignments [21], branching decisions [22], and integer variables [23]. [24] employed Random Forests to predict solutions for AC optimal power flow problems. [25] utilized nearest neighbor search to warm-start tight convex relaxations in nonconvex trajectory optimization problems. In robot control, neural networks were used to predict initializations for trajectory optimizers or Model Predictive Control (MPC) [26], [27], [28]. An exciting line of recent work developed *differentiable* optimization algorithms, which allow jointly learning objectives, constraints, and initializations by backpropagating through the optimization process [29], [30], [31], [32]. In contrast, we learn multiple initializations instead of one, and we do so without strong assumptions about the task or the optimizer. Concurrently, work in multi-agent driving formulates a differentiable potential-game layer that infers multiple candidate trajectories and refines them in parallel end-to-end [33]; unlike our approach, it targets a domain-specific, multi-agent setting and optimizes task losses through the solver. Relatedly, in neural combinatorial optimization, PolyNet [34] learns diverse solution strategies within a single decoder and selects the best candidate at inference, and Poppy [35] trains a population of complementary RL policies with a winner-takes-all population objective; both diversify full solutions rather than initial conditions for downstream solvers. Notably, [36] used multiple initializations by repurposing a motion prediction model and Bézier curve fitting for a downstream MPC; however, this approach is specifically tailored for autonomous driving, incorporating a dedicated motion prediction module.

Parallel optimizers. Leveraging parallelism has a long history in optimization research [37]. With recent advances in parallel computing hardware, such as GPUs, methods that execute multiple optimizers in parallel have also emerged. For example, [38] introduced cuRobo, a GPU-accelerated method combining L-BFGS and particle-based optimization for robotic manipulators. Similarly, [39] utilized massive parallel GPU computation for efficient inverse kinematics and trajectory optimization. [40] proposed a topology-driven

method that plans for multiple evasive maneuvers in parallel. [41] focused on initializing parallel optimizers through rough paths. However, these works have not utilized learning. [28] explored learning-based strategies for initializing trajectory optimizers based on a database of previous solutions and ensemble-learned models, particularly in manipulation and humanoid control tasks. In contrast, we propose a single neural network to generate multiple initializations, which, as shown in our experiments, significantly outperforms the ensemble-based approach.

III. Initializing Optimizers

Problem setup. In the most general form, we need to solve instances of a parameterized optimization problem,

$$\boldsymbol{x}^\star(\boldsymbol{\psi}) = \arg\min_{\boldsymbol{x}} J(\boldsymbol{x};\boldsymbol{\psi}) \quad \text{s.t.} \quad \begin{matrix} \boldsymbol{g}(\boldsymbol{x};\boldsymbol{\psi}) \leq \mathbf{0} \\ \boldsymbol{h}(\boldsymbol{x};\boldsymbol{\psi}) = \mathbf{0} \end{matrix},$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the variable vector to be optimized, J is the objective function, $\boldsymbol{g}$ and $\boldsymbol{h}$ are collections of inequality and equality constraints, and $\boldsymbol{\psi} \in \mathbb{R}^m$ is a parameter vector that defines the problem instance, e.g., parameters of the objective function and constraints that differ across problem instances. A local optimization algorithm, $\mathbf{Opt}$, attempts to find an optimum of J, namely,

$$\hat{\boldsymbol{x}}^\star = \mathbf{Opt}(J, \boldsymbol{\psi}, t_{\text{lim}}; \boldsymbol{x}^{\text{init}}),$$

where $\boldsymbol{x}^{\text{init}}$ is initial solution provided to the optimizer, and t_{lim} is the runtime limit.

Heuristic methods. A common choice of the initial solution, $\boldsymbol{x}_{\text{init}}$, is the solution to a previously solved similar problem instance, referred to as a *warm-start*. For example, in optimal control the warm-start is typically the solution from the previous timestep, shifted and padded with zeros, $\boldsymbol{x}^{\text{w.s.}} := \{\{\boldsymbol{x}^{\text{cand}}_{t+k}\}_{k=1}^{H-2}, \mathbf{0}\}$ [42]. This heuristic often works well in practice, however, it can struggle when large changes in the problem instance, $\boldsymbol{\psi}$, occur between consecutive time steps, leading to significant shifts in the optimal solution. For example, in autonomous driving, abrupt events like a traffic light switch or the sudden appearance of a pedestrian might drastically alter the reference trajectory or constraints. In such cases, the previous solution becomes a poor initialization, and the optimizer may fail to find a good solution within the allocated time frame.

IV. Learning Multiple Initial Solutions (MISO)

The main idea of **MISO** is to train a single neural network to predict multiple initial solutions to an optimization problem, such that the initial solutions cover promising regions of the optimization landscape, eventually allowing a local optimizer to find a solution close to the global optimum. The key questions are then how to design a multi-output predictor; how to utilize multiple initial solutions in existing optimizers; and how to train the predictor to output a diverse set of initial solutions. In the following, we discuss our proposed solutions to these questions, illustrate the need for multimodality with a toy example, and discuss applications to optimal control.

A. Multi-output predictor

Our multi-output predictor is a standard Transformer model (4 layers, 2 attention heads, 64-dimensional embeddings, trained for 125 epochs with AdamW)–chosen for its simplicity and clarity. It takes the problem instance, $\boldsymbol{\psi}$, as input and outputs K initial solutions for the optimization problem,

$$\{\hat{\boldsymbol{x}}^{\text{init}}_k\}_{k=1}^K = \boldsymbol{f}(\boldsymbol{\psi};\boldsymbol{\theta}),$$

where $\boldsymbol{\theta}$ are the learned parameters of the network. We train the network on a dataset of problem instances and their corresponding (near-)optimal solutions, $\{(\boldsymbol{\psi}_i, \boldsymbol{x}^\star_i)\}_{i=1}^n$. Such dataset can be generated offline, for example, by running a slow yet globally optimal solver, or allowing the same local optimizer to run with longer time limits, potentially many times from different initial solutions.

B. Optimization with multiple initial solutions

We propose two distinct settings to leverage multiple initial solutions: *single-optimizer* and *multiple-optimizers*. The resulting frameworks are illustrated in Fig. 1.

Single optimizer. In the single-optimizer setting we run a single instance of the optimizer with the most promising initial solution, $\hat{\boldsymbol{x}}^\star = \mathbf{Opt}(J, \boldsymbol{\psi}, t_{\text{limit}}; \hat{\boldsymbol{x}}^{\text{init}})$. We introduce a selection function, $\boldsymbol{\Lambda}$, which, given a set of candidate solutions and the problem instance $\boldsymbol{\psi}$, returns the most promising candidate, $\hat{\boldsymbol{x}}^{\text{init}} = \boldsymbol{\Lambda}(\{\hat{\boldsymbol{x}}^{\text{init}}_k\}_{k=1}^K, \boldsymbol{\psi})$. A reasonable choice for $\boldsymbol{\Lambda}$ used in our experiments is selecting the candidate that minimizes the objective function the optimizer aims to minimize, i.e., $\boldsymbol{\Lambda} := \arg\min_k J(\hat{\boldsymbol{x}}^{\text{init}}_k; \boldsymbol{\psi})$.

Multiple optimizers. In the multiple-optimizers setting, we assume multiple instances of the optimizer can be executed in parallel. We then initialize each optimizer with a different initial solution, $\boldsymbol{x}^\star_k = \mathbf{Opt}_k(J, \boldsymbol{\psi}, t_{\text{limit}}; \hat{\boldsymbol{x}}^{\text{init}}_k), \quad k \in \{1, \dots, K\}$. To select a single solution from the outputs of the optimizers, we can use the same selection function $\boldsymbol{\Lambda}$, as in the previous case, e.g., the solution that minimizes the objective function.

Guarantees. Our framework can be trivially generalized to allow a different number of optimizers and initial solution predictions, as well as using a heterogeneous set of optimization methods. To maintain performance guarantees, one may include traditional initialization methods, such as warm-start heuristics, as part of the set of initializations. **MISO** is guaranteed to improve over the existing default strategy by design. In the single-optimizer setting the best initial solution is always equal or better than the default according to the selection function $\boldsymbol{\Lambda}$; and in the multiple-optimizer setting the final solution is equal or better than using only the default initialization.

C. Training strategies

The ultimate goal is to predict multiple initial solutions so that the downstream optimizer can find a solution close to the global optima, i.e., $J(\hat{\boldsymbol{x}}^\star; \boldsymbol{\psi}) \approx J(\boldsymbol{x}^\star; \boldsymbol{\psi})$. Training a neural network directly for this objective is not feasible in general. Instead, we propose proxy training objectives that combine two terms: a regression term that encourages outputs

to be close to the global optimum, e.g., $\mathcal{L}_{\text{reg}}(\hat{\boldsymbol{x}}_k^{\text{init}}, \boldsymbol{x}^\star) = \|\hat{\boldsymbol{x}}_{\text{init}} - \boldsymbol{x}^\star\|$, where $\|\cdot\|$ is a distance metric; along with a *diversity* term that promotes outputs being different from each other, thereby covering various regions of the solution space. An illustrative example is in Sect. IV-D. In the following, we present three simple training strategies promoting diversity and preventing mode collapse. We discuss alternative formulations, with probabilistic modeling and reinforcement learning, in Sect. VII.

Pairwise distance loss. A simple method to encourage the model's outputs to differ from each other is to penalize the pairwise distance between all outputs. The overall loss combines this dispersion-promoting term with the regression loss,

$$\mathcal{L}_{\text{PD}} = \frac{1}{K}\sum_{k=1}^{K}\mathcal{L}_{\text{reg}}(\hat{\boldsymbol{x}}_k^{\text{init}}, \boldsymbol{x}^\star) + \alpha_K \frac{1}{K}\sum_{k=1}^{K}\mathcal{L}_{\text{PD},k}(\hat{\boldsymbol{x}}_k^{\text{init}}),$$

$$\mathcal{L}_{\text{PD},k} = \frac{1}{K-1}\sum_{\substack{k'=1 \\ k'\neq k}}^{K}\|\hat{\boldsymbol{x}}_k^{\text{init}} - \hat{\boldsymbol{x}}_{k'}^{\text{init}}\|,$$

where α_K is a hyperparameter that balances the trade-off between accuracy and dispersion.

Winner-takes-all loss. A more interesting way to encourage multimodality is to select the best-predicted output at training time and only minimize the regression loss for this specific prediction,

$$\mathcal{L}_{\text{WTA}} = \min_k\{\mathcal{L}_{\text{reg}}(\hat{\boldsymbol{x}}_k^{\text{init}}, \boldsymbol{x}^\star)\}.$$

Intuitively, the model only needs one of its outputs to be close to the ground truth, while the other predictions are not penalized for deviating, potentially aligning with different regions of the underlying distribution. Similar losses have been used, e.g., in multiple-choice learning [43]. One advantage of this approach is that it is hyperparameter-free.

Mixture loss. Lastly, we consider a combination of the previous two approaches to potentially enhance performance, as it provides some measure of dispersion we can tune,

$$\mathcal{L}_{\text{MIX}} = \min_k\left\{\mathcal{L}_{\text{reg}}(\hat{\boldsymbol{x}}_k^{\text{init}}, \boldsymbol{x}^\star) + \alpha_K \Phi\left(\mathcal{L}_{\text{PD},k}(\hat{\boldsymbol{x}}_k^{\text{init}})\right)\right\},$$

here, Φ is an upper-bounded function, such as $\min$ or $\tanh$, designed to limit the contribution of the pairwise distance term.

Beyond the losses above, **MISO** could be integrated with other training paradigms, such as reinforcement learning or probabilistic modeling, directions left for future work (Sect. VII).

D. Illustrative example

To illustrate the advantage of using a single model with multimodal outputs compared to regression models or ensembles of regressors, we examine a straightforward one-dimensional optimization problem aimed at minimizing the cost function $J(x)$ shown in Fig. 2 (left). The function features two global minima, denoted as **A** and **C**, with a local minimum located between them at **B**.

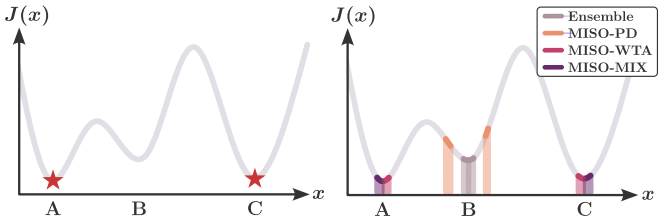


Fig. 2. **Left**: The cost $J(x)$ with global minima at **A** and **C** and a local minimum at **B**. **Right**: Different methods' predictions - demonstrating why explicitly promoting multimodality is important.

Applying our learning framework to this simple problem, the dataset of optimal solutions includes instances of **A** and **C**. A single-output regression model has no means to distinguish the two modes and inevitably learns to predict the mean of examples in the dataset, somewhere near **B**. Consequently, the local optimizer is likely to converge to the suboptimal local minimum at **B**. Constructing an ensemble of such models to generate multiple initial solutions does not mitigate this issue, as each ensemble member tends to be biased toward the mean of the two modes near **B**. We implemented the optimization problem and showed the predictions for different training strategies in Fig. 2 (right). Indeed, an ensemble of single-output predictors fails to predict a global optimum, while our multi-output predictor succeeds with winner-takes-all and mixture losses.

While the problem considered here is purposefully simplistic, the existence of local minima is the key challenge in most optimization problems.

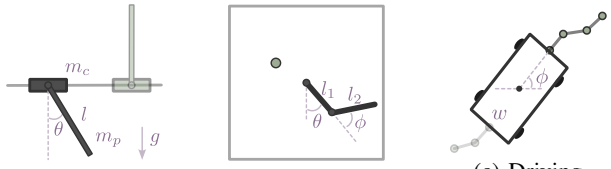
E. Application to optimal control

MISO is applicable to a broad class of sequential optimization problems; however, for the sake of evaluation, we focus on optimal control problems. Optimal control has a wide range of applications, e.g., in robotics, autonomous driving, and many other domains with strict runtime requirements, and due to the complexity induced by constraints and non-convex costs, local optimization algorithms are highly sensitive to the initial solution.

In optimal control the optimization variable $\boldsymbol{x}$ represents a trajectory defined as a sequence of states and control inputs over discrete time steps: $\boldsymbol{\tau} = \{\boldsymbol{s}_t, \boldsymbol{u}_t\}_{t=1:H}$. Here, $\boldsymbol{s}_t \in \mathcal{S}$ and $\boldsymbol{u}_t \in \mathcal{U}$ denote the state and control input at time step $t \in \mathbb{Z}^+$, and $H \in \mathbb{Z}^+$ is the optimization horizon. The constraints involve adhering to the system dynamics $\boldsymbol{f}_d(\boldsymbol{s}_{t+1}, \boldsymbol{s}_t, \boldsymbol{u}_t) = \boldsymbol{0}$, starting from an initial state $\boldsymbol{s}_0 = \boldsymbol{s}_{\text{curr}}$, where $\boldsymbol{s}_{\text{curr}}$ represents the system's current state. The problem instance parameters $\boldsymbol{\psi}$ encompass the initial state $\boldsymbol{s}_0$, and other domain-specific variables that parameterize the objective function or constraints, such as target states, reference trajectories, obstacle positions, friction coefficients, temperature, etc.

A specific property of optimal control problems is that the relationship between optimization variables, states $\boldsymbol{s}_t$ and controls $\boldsymbol{u}_t$, are defined by the dynamics constraint $\boldsymbol{f}_d$; and the initial state $\boldsymbol{s}_0$ is given. Therefore, a sequence of controls uniquely defines an (initial) solution. We can leverage this

property by learning to predict only a sequence of controls instead of the full optimization variable of state-control sequences. Further, one can define the training loss over either control, state, or state-control sequences and backpropagate gradients through the dynamics constraint as long as it is differentiable. In our experiments, we use state-control loss by default as we found it to improve both our and baseline learning methods.



(a) Cart-pole (b) Reacher (c) Driving

Fig. 3. Optimal control tasks used in our experiments.

V. Experimental setup

Tasks. We evaluated our method on the three robot control benchmark tasks shown in Fig. 3, each employing a distinct local optimization algorithm. **Cart-pole.** This task involves balancing a pole upright while moving a cart toward a randomly selected target position [44], using a first-order box Differential Dynamic Programming (DDP) optimizer [29]. **Reacher.** In this task, a two-link planar robotic arm needs to reach a target placed at a random positon [45], using a Model Predictive Path Integral (MPPI) optimizer [46]. **Autonomous Driving.** Based on the nuPlan benchmark [47], this task focuses on trajectory tracking in complex urban environments by following a reference trajectory generated by a Predictive Driver Model (PDM) planner [48], using the Iterative Linear Quadratic Regulator (iLQR) optimizer [49].

Baselines. We compare **MISO** to a range of alternative methods to provide single or multiple initial solutions. For a single initial solution, we considered: **Warm-start**, the default method that uses the optimizer output from the last problem instance; **Regression**, a single-output regression model (the $K = 1$ version of **MISO**); **Oracle Proxy**, optimization with unlimited runtime, which we also used to generate our training data. For methods that generate multiple initial solutions, we considered: **Warm-start with perturbations**, which extends the warm-start approach by adding Gaussian noise to the optimizer output from the last problem instance; **Regression with perturbations**, where Gaussian noise is introduced to the predictions of the single-output regression model; **Multi-output regression**, a naive multi-output regression model without a diversity-promoting objective; and **Ensemble**, which trains multiple single-output neural networks with different random initializations. Finally, we assessed variants of our proposed method with the different training losses discussed in Sect. IV-C: **MISO-PD**, **MISO-WTA**, and **MISO-MIX**.

Evaluation settings. We employ two evaluation modes. (i) **One-off**, where the optimization task is treated as an isolated problem with the objective of finding the minimum of a given function. This mode serves as the default configuration for training neural networks, where data is replayed to the model, and the optimizer's solution is recorded but not executed. Methods are assessed by the mean cost of the optimizer's output over problem instances. (ii) **Sequential**, which involves solving a series of related optimization problems, executing each proposed solution, and starting the subsequent optimization from the resulting state. This setting simulates real-world conditions where the optimizer continuously interacts with a dynamic environment in a closed loop. We evaluate performance by taking the mean cost over problems in a sequence, and then the mean over sequences.

To account for the additional time required to predict initial solutions, we assumed that all models perform inference in under 0.85ms, which was the case for all methods on both CPU and GPU, except for the ensemble. In the autonomous driving task, we then reduced the runtime allocated to the optimizer accordingly.

Implementation details. To generate the training data, we first create a set of problem instances by sequentially executing the optimizer initialized with the default warm-start strategy. The problem instances are then fed again to an "oracle" version of the optimizer with a significantly increased runtime limit, and the resulting solutions are recorded. After training, evaluation is performed on a separate set of problem instances that is disjoint from the training data (sampled from the same distribution). All experiments are conducted on an Intel Core i9-13900KF CPU and an NVIDIA RTX 4090 GPU.

VI. Results

Our main results for optimization with different initial solutions are reported in Table I and Table II for single optimizer and multiple optimizers settings, respectively. Figure 4 shows the effect of the number of predicted initial solutions. Figure 5 provides qualitative results.

Single optimizer. In the single-optimizer setting, Table I, we first observe that even one learned initialization outperforms heuristic solutions (regression vs. warm-start), in almost all settings, and in particular in the most challenging autonomous driving task. We then examine the impact of generating multiple initial solutions. Perturbations-based methods show some improvement over their single-initialization counterparts in most cases, and ensembles of independently learned models perform consistently better than single models. Finally, our proposed multi-output methods demonstrate substantial improvements over all baselines because they can learn to predict diverse multimodal initial solutions. Specifically, **MISO-WTA** or **MISO-MIX** achieve the lowest mean costs across all tasks. Considering the pairwise distance term alone proves insufficient to ensure adequate diversity, whereas incorporating it with **MISO-WTA** often boosts performance, yet, its effectiveness varies. The pairwise distance coefficient α_K requires task-specific tuning (values ranged from -0.1 to -0.01 across tasks), which underscores the challenge of selecting optimal hyperparameters; the winner-takes-all loss, being hyperparameter-free, offers a more robust alternative. As expected, improvements are consistently larger in the more important sequential optimization setting, where errors over time compound.

TABLE I

RESULTS FOR THE SINGLE OPTIMIZER SETTING. THE MEAN COST (± STANDARD ERROR) OF SOLUTIONS FOUND BY THE SINGLE OPTIMIZER USING DIFFERENT INITIAL SOLUTIONS ACROSS TASKS AND EVALUATION SETTINGS.

		One-Off Optimization			Sequential Optimization		
Method	K	**Reacher**	**Cart-pole**	**Driving**	**Reacher**	**Cart-pole**	**Driving**
Single Optimizer							
Warm-start	1	13.48 ±0.88	11.69 ±0.84	283.86 ±37.91	13.48 ±0.88	11.69 ±0.84	283.86 ±37.91
Regression	1	13.40 ±0.88	11.19 ±0.80	74.23 ±7.69	19.56 ±0.52	6.18 ±0.47	70.62 ±7.38
Warm-start w. perturb	32	13.46 ±0.88	11.64 ±0.83	145.01 ±23.01	14.71 ±0.93	16.29 ±0.44	164.75 ±22.84
Regression w. perturb	32	13.38 ±0.88	11.16 ±0.80	67.69 ±8.01	15.28 ±0.58	5.74 ±0.47	66.75 ±6.56
Multi-output regression	32	13.41 ±0.88	11.21 ±0.80	70.25 ±8.75	18.49 ±0.55	6.62 ±0.45	78.74 ±8.99
Ensemble	32	13.39 ±0.88	10.94 ±0.79	47.22 ±4.71	8.40 ±0.40	3.55 ±0.34	52.59 ±4.81
MISO-PD	32	13.41 ±0.88	11.22 ±0.80	66.06 ±7.48	19.20 ±0.49	6.07 ±0.45	71.90 ±7.90
MISO-WTA	32	13.36 ±0.88	**10.48 ±0.77**	**30.17 ±2.24**	2.72 ±0.21	0.83 ±0.06	**30.75 ±2.15**
MISO-MIX	32	**12.74 ±0.86**	**10.48 ±0.77**	33.95 ±2.39	**2.44 ±0.20**	**0.79 ±0.04**	33.38 ±2.21
Oracle Proxy	1	13.43 ±0.88	11.01 ±0.80	41.94 ±4.31	6.88 ±0.58	4.54 ±0.71	26.52 ±2.00

TABLE II

RESULTS FOR THE MULTIPLE OPTIMIZERS SETTING. MEAN COST ± STANDARD ERROR) OF SOLUTIONS FOUND BY MULTIPLE OPTIMIZERS USING DIFFERENT INITIAL SOLUTIONS ACROSS TASKS AND EVALUATION SETTINGS.

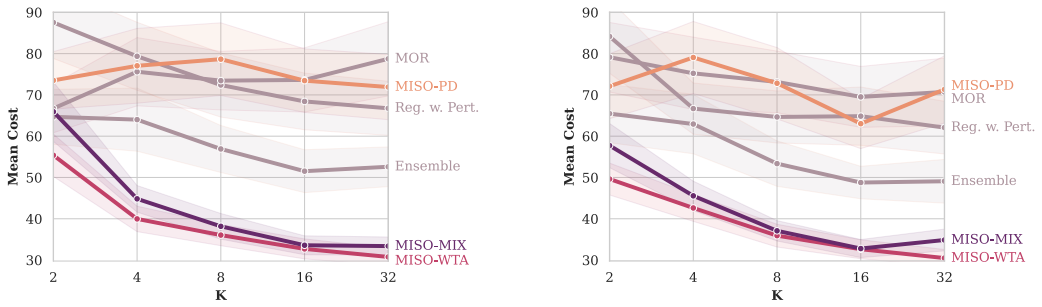
		One-Off Optimization			Sequential Optimization		
Method	K	**Reacher**	**Cart-pole**	**Driving**	**Reacher**	**Cart-pole**	**Driving**
Multiple Optimizers							
Warm-start w. perturb	32	13.41 ±0.88	10.93 ±0.79	155.53 ±24.33	5.89 ±0.50	6.68 ±0.59	162.13 ±34.10
Regression w. perturb	32	13.34 ±0.88	11.12 ±0.80	64.88 ±6.84	3.53 ±0.27	5.36 ±0.48	62.07 ±6.39
Multi-output regression	32	13.34 ±0.88	11.21 ±0.80	70.29 ±9.13	3.31 ±0.26	6.38 ±0.43	70.71 ±8.18
Ensemble	32	13.34 ±0.88	10.65 ±0.78	45.44 ±4.64	3.08 ±0.23	2.21 ±0.20	49.08 ±5.29
MISO-PD	32	13.34 ±0.88	11.22 ±0.80	67.62 ±7.58	3.42 ±0.27	6.09 ±0.47	71.33 ±8.13
MISO-WTA	32	13.34 ±0.88	**10.29 ±0.76**	**30.87 ±2.30**	2.21 ±0.16	0.76 ±0.05	**30.48 ±2.07**
MISO-MIX	32	**12.72 ±0.86**	**10.29 ±0.76**	33.52 ±2.35	**1.56 ±0.14**	**0.63 ±0.02**	34.85 ±2.64

Multiple optimizers. When considering the multiple-optimizers setting, we observe the same trend. Learning-based methods outperform heuristic ones, and multi-output approaches yield further enhancements. As expected, the use of multiple optimizers leads to consistently better results compared to the single-optimizer setting due to increased exploration of the solution space.

Scaling with the number of initial solutions. Figure 4 shows that our method scales effectively and consistently with the number of predicted initial solutions K, and outperforms other approaches across varying values of K. Regarding computational cost, the multiple-optimizers setting increases total computation proportionally to K, though this can be mitigated through parallelization. Importantly, unlike ensemble approaches where inference time scales with K (each model requires a separate forward pass), our multi-output model maintains near-constant inference time (<0.85ms on both CPU and GPU), making it practical for real-time applications. We further evaluate mode diversity, and find that, in line with our conclusions, all **MISO** outputs remain useful even when K increases.

Performance guarantees. To evaluate the guarantees discussed in Sect. IV-B, we added the warm-start initial solution to the set of candidates of **MISO-WTA**and **MISO-MIX**. We observed in all problem instances the cost of the best initial solution to be equal or lower than the cost of the warm start in the single optimizer setting; and the cost of the final solution to be lower or equal than for the warm start in the multiple optimizer setting. The mean costs remained mostly similar to **MISO**, and improved slightly in some cases.

Qualitative results. Figure 5 (left) depicts the optimizer's output trajectories with different initial solutions for the autonomous driving task. In this scenario, the high-level planner abruptly alters the reference path, which could happen, e.g., because of a newly detected pedestrian. The change in reference path makes the previous solution (warm-start) a poor initialization, and the optimizer converges to a local minimum that minimizes control effort but is far from the desired path. Regression and model ensemble also fail to predict a good initial solution. In contrast, **MISO-WTA** adapts to this sudden reference change and closely follows the reference path. Figure 5 (right) depicts **MISO**'s initial solutions for the cart-pole task. The different outputs capture different modes of the solution space (maintaining balance while moving, swinging leftward, and swinging rightward), showing **MISO**'s ability to generate diverse and multimodal solutions.



(a) Single Optimizer (b) Multiple Optimizers

Fig. 4. Mean cost of the driving task (Sequential Optimization) for varying K values. The shaded regions indicate the standard error.

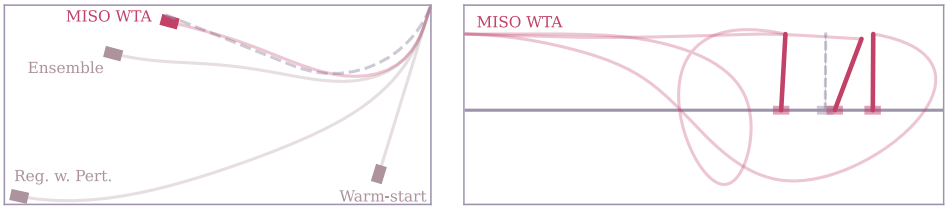


Fig. 5. Single Optimizer for Driving **(left)** and Cart-Pole **(right)**. On the left, we show each method's adaptation when the high-level planner abruptly modifies the reference path. On the right, we illustrate multiple trajectories predicted by **MISO-WTA**

Additional experiments. Under the same runtime budget, explicit convex QP solvers (CVXPY backends) and standard multi-init schemes (grid/LHS) were not competitive with **MISO**.

Summary. Overall, our methods significantly outperform the other baselines in both settings. The consistent superiority of the **MISO-MIX** and **MISO-WTA** methods across different tasks and configurations underscores the advantages of using learning-based multi-output strategies for generating initial solutions. These findings demonstrate that promoting diversity among multiple initializations is crucial for improving optimization outcomes, especially when combined with multiple optimizers.

VII. Conclusions and Future Work

We introduced Learning Multiple Initial Solutions (**MISO**), a novel framework for learning multiple diverse initial solutions that significantly enhance the reliability and efficiency of local optimization algorithms across various settings. Extensive experiments in optimal control demonstrated that our method consistently outperforms baseline approaches and scales efficiently with the number of initializations.

Limitations. Our approach is not without limitations. First, to train a useful model, we rely on the coverage and quality of the training data, as the method does not directly interact with the optimizer or the underlying objective function. Second, the underlying assumption of our regression loss is that initial solutions closer to the global optimum increase the likelihood of successful optimization may not hold in complex optimization landscapes with intricate constraints. Third, in highly complex optimization problems where each solution constitutes a high-dimensional and intricate structure, accurately learning initial solution candidates can become exceedingly challenging, potentially diminishing the effectiveness of our approach. Finally, while we demonstrate generalization to unseen problem instances within each task, cross-task transfer (e.g., training on one control problem and applying to a different one) is not evaluated and remains an open question.

Future work. There are several promising directions for future research. To address the aforementioned limitations, one may simply incorporate the optimization objective into the model training loss, thus creating a direct link to the final optimization goal. Alternatively, using reinforcement learning (RL) to train **MISO** is a particularly exciting opportunity. By framing the problem in an RL context, e.g., where the reward is the negative cost of the optimizer's final solution, models would be directly trained to maximize the probability of the optimizer finding the global optima and may learn to specialize to the specific optimizer. One challenge would be computational, as RL would require running the optimizer numerous times during training.

Other extensions of our approach include probabilistic modeling, e.g., Gaussian mixture models, variational autoencoders, or diffusion models; however, inference overhead is a key concern—for instance, diffusion models require $\sim$378ms per sequence compared to $<$1ms for our multi-output regressor, making them impractical for real-time control. Future work may explore alternative selection functions, such as risk measures or criteria based on stability, robustness, exploration, or other domain-specific metrics; as well as using a heterogeneous set of parallel optimizers. Finally, we are excited about various possible applications in optimal control and beyond, where sequences of similar optimization problems need to be solved, for example, localization and mapping in robotics, financial optimization, traffic routing optimization, or even training neural networks with different initial weights, e.g., for meta-learning.

REFERENCES

[1] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Veh.*, vol. 1, pp. 33–55, 2016.

[2] Y. Ye, H. Pei, B. Wang, P.-Y. Chen, Y. Zhu, J. Xiao, and B. Li, "Reinforcement-learning based portfolio management with augmented asset movement prediction states," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, 2020.

[3] S. Mugel, C. Kuchkovsky, E. Sánchez, S. Fernández-Lorenzo, J. Luis-Hita, E. Lizaso, and R. Orús, "Dynamic portfolio optimization with real datasets using quantum processors and quantum-inspired tensor networks," *Phys. Rev. Res.*, vol. 4, p. 013006, 2022.

[4] H. Michalska and D. Q. Mayne, "Robust receding horizon control of constrained nonlinear systems," *IEEE Trans. Automat. Contr.*, vol. 38, pp. 1623–1633, 2002.

[5] P. O. Scokaert, D. Q. Mayne, and J. B. Rawlings, "Suboptimal model predictive control (feasibility implies stability)," *IEEE Trans. Automat. Contr.*, vol. 44, pp. 648–654, 2002.

[6] S. Sun, Z. Cao, H. Zhu, and J. Zhao, "A survey of optimization methods from a machine learning perspective," *IEEE Trans. Cybern.*, vol. 50, pp. 3668–3681, 2019.

[7] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2010.

[8] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, *et al.*, "Learning dexterous in-hand manipulation," *Int. J. Robot. Res.*, vol. 39, pp. 3–20, 2020.

[9] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, *et al.*, "Learning to navigate in complex environments," *arXiv preprint arXiv:1611.03673*, 2016.

[10] T. Chen, X. Chen, W. Chen, H. Heaton, J. Liu, Z. Wang, and W. Yin, "Learning to optimize: A primer and a benchmark," *J. Mach. Learn. Res.*, vol. 23, pp. 1–59, 2022.

[11] J. Sacks and B. Boots, "Learning to optimize in model predictive control," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2022.

[12] A. O. Fajemisin, D. Maragno, and D. den Hertog, "Optimization with constraint learning: A framework and survey," *Eur. J. Oper. Res.*, vol. 314, pp. 1–14, 2024.

[13] I. Lenz, R. A. Knepper, and A. Saxena, "Deepmpc: Learning deep latent features for model predictive control." in *Robot.: Sci. Syst.*, vol. 10, 2015.

[14] N. Wahlström, T. B. Schön, and M. P. Deisenroth, "From pixels to torques: Policy learning with deep dynamical models," *arXiv preprint arXiv:1502.02251*, 2015.

[15] A. Tamar, G. Thomas, T. Zhang, S. Levine, and P. Abbeel, "Learning from the hindsight plan—episodic mpc improvement," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2017.

[16] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, "Learning latent dynamics for planning from pixels," in *Int. Conf. Mach. Learn. (ICML)*, 2019.

[17] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2018.

[18] X. Xiao, T. Zhang, K. Choromanski, E. Lee, A. Francis, J. Varley, S. Tu, S. Singh, P. Xu, F. Xia, *et al.*, "Learning model predictive controllers with real-time attention for real-world navigation," *arXiv preprint arXiv:2209.10780*, 2022.

[19] T. C. Johnson, C. Kirches, and A. Wachter, "An active-set method for quadratic programming based on sequential hot-starts," *SIAM J. Optim.*, vol. 25, pp. 967–994, 2015.

[20] T. Marcucci and R. Tedrake, "Warm start of mixed-integer programs for model predictive control of hybrid systems," *IEEE Trans. Automat. Contr.*, vol. 66, pp. 2433–2448, 2020.

[21] V. Nair, S. Bartunov, F. Gimeno, I. Von Glehn, P. Lichocki, I. Lobov, B. O'Donoghue, N. Sonnerat, C. Tjandraatmadja, P. Wang, *et al.*, "Solving mixed integer programs using neural networks," *arXiv preprint arXiv:2012.13349*, 2020.

[22] N. Sonnerat, P. Wang, I. Ktena, S. Bartunov, and V. Nair, "Learning a large neighborhood search algorithm for mixed integer programs," *arXiv preprint arXiv:2107.10201*, 2021.

[23] D. Bertsimas and B. Stellato, "The voice of optimization," *Mach. Learn.*, vol. 110, pp. 249–277, 2021.

[24] K. Baker, "Learning warm-start points for ac optimal power flow," in *IEEE Int. Workshop Mach. Learn. Signal Process. (MLSP)*, 2019.

[25] S. Kang, X. Xu, J. Sarva, L. Liang, and H. Yang, "Fast and certifiable trajectory optimization," *arXiv preprint arXiv:2406.05846*, 2024.

[26] S. W. Chen, T. Wang, N. Atanasov, V. Kumar, and M. Morari, "Large scale model predictive control with neural networks and primal active sets," *Automatica*, vol. 135, p. 109947, 2022.

[27] T. Wang and J. Ba, "Exploring model-based planning with policy networks," *arXiv preprint arXiv:1906.08649*, 2019.

[28] T. S. Lembono, A. Paolillo, E. Pignat, and S. Calinon, "Memory of motion for warm-starting trajectory optimization," *IEEE Robot. Autom. Lett.*, vol. 5, pp. 2594–2601, 2020.

[29] B. Amos, I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter, "Differentiable mpc for end-to-end planning and control," *Adv. Neural Inf. Process. Syst.*, vol. 31, 2018.

[30] S. East, M. Gallieri, J. Masci, J. Koutnik, and M. Cannon, "Infinite-horizon differentiable model predictive control," *arXiv preprint arXiv:2001.02244*, 2020.

[31] P. Karkus, B. Ivanovic, S. Mannor, and M. Pavone, "Diffstack: A differentiable and modular control stack for autonomous vehicles," in *Conf. Robot Learn. (CoRL)*, 2023.

[32] R. Sambharya, G. Hall, B. Amos, and B. Stellato, "Learning to warm-start fixed-point optimization algorithms," *J. Mach. Learn. Res.*, vol. 25, pp. 1–46, 2024.

[33] C. Diehl, T. Klosek, M. Krueger, N. Murzyn, T. Osterburg, and T. Bertram, "Energy-based potential games for joint motion forecasting and control," *arXiv preprint arXiv:2312.01811*, 2023.

[34] A. Hottung, M. Mahajan, and K. Tierney, "Polynet: Learning diverse solution strategies for neural combinatorial optimization," *arXiv preprint arXiv:2402.14048*, 2024.

[35] N. Grinsztajn, D. Furelos-Blanco, S. Surana, C. Bonnet, and T. Barrett, "Winner takes it all: Training performant rl populations for combinatorial optimization," *Adv. Neural Inf. Process. Syst.*, vol. 36, pp. 48 485–48 509, 2023.

[36] M.-K. Bouzidi, Y. Yao, D. Goehring, and J. Reichardt, "Learning-aided warmstart of model predictive control in uncertain fast-changing traffic," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2024.

[37] J. T. Betts and W. P. Huffman, "Trajectory optimization on a parallel processor," *J. Guid. Control Dyn.*, vol. 14, pp. 431–439, 1991.

[38] B. Sundaralingam, S. K. S. Hari, A. Fishman, C. Garrett, K. Van Wyk, V. Blukis, A. Millane, H. Oleynikova, A. Handa, F. Ramos, *et al.*, "curobo: Parallelized collision-free minimum-jerk robot motion generation," *arXiv preprint arXiv:2310.17274*, 2023.

[39] H. Huang, B. Sundaralingam, A. Mousavian, A. Murali, K. Goldberg, and D. Fox, "Diffusionseeder: Seeding motion optimization with diffusion for rapid motion planning," *arXiv preprint arXiv:2410.16727*, 2024.

[40] O. De Groot, L. Ferranti, D. M. Gavrila, and J. Alonso-Mora, "Topology-driven parallel trajectory optimization in dynamic environments," *IEEE Trans. Robot.*, 2024.

[41] L. Barcelos, T. Lai, R. Oliveira, P. Borges, and F. Ramos, "Path signatures for diversity in probabilistic trajectory optimisation," *Int. J. Robot. Res.*, vol. 43, pp. 1693–1710, 2024.

[42] P. Otta, O. Santin, and V. Havlena, "Measured-state driven warm-start strategy for linear mpc," in *Eur. Control Conf. (ECC)*, 2015.

[43] A. Guzman-Rivera, D. Batra, and P. Kohli, "Multiple choice learning: Learning to produce multiple structured outputs," *Adv. Neural Inf. Process. Syst.*, vol. 25, 2012.

[44] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Trans. Syst., Man, Cybern.*, pp. 834–846, 2012.

[45] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. d. L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, *et al.*, "Deepmind control suite," *arXiv preprint arXiv:1801.00690*, 2018.

[46] G. Williams, A. Aldrich, and E. Theodorou, "Model predictive path integral control using covariance variable importance sampling," *arXiv preprint arXiv:1509.01149*, 2015.

[47] H. Caesar, J. Kabzan, K. S. Tan, W. K. Fong, E. Wolff, A. Lang, L. Fletcher, O. Beijbom, and S. Omari, "nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles," *arXiv preprint arXiv:2106.11810*, 2021.

[48] D. Dauner, M. Hallgarten, A. Geiger, and K. Chitta, "Parting with misconceptions about learning-based vehicle motion planning," in *Conf. Robot Learn. (CoRL)*, 2023.

[49] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems," in *Int. Conf. Inform. Control, Autom. Robot.*, vol. 2, 2004.