Neural Imitation Learning for Real-Time Control of 3-DOF Robotic Manipulators

Anonymous Author(s)
Affilation, Address
anon.email@example.org

Abstract

This paper investigates the application of neural imitation learning to approximate Model Predictive Control (MPC) policies for real-time control of 3-degree-of-freedom (3-DOF) robotic manipulators. We present a hierarchical baseline controller combining inverse kinematics (IK) with MPC, and subsequently develop a data generation pipeline to collect expert demonstrations. We evaluate multiple neural network architectures—including feedforward networks, recurrent neural networks (RNNs), and Transformers—to learn a surrogate policy that maps historical state sequences to control actions. Our results demonstrate that learned policies can achieve significant computational speed-ups while maintaining comparable tracking performance to the original MPC, addressing critical limitations in high-frequency control applications. We analyze generalization capabilities, stability considerations, and trade-offs between different architectural choices. The proposed methodology provides a path toward deploying complex optimal control strategies on computationally constrained platforms.

1 Introduction

Model Predictive Control (MPC) has emerged as a powerful paradigm for robotic manipulation, offering inherent constraint handling and optimality guarantees. However, the computational demands of solving optimization problems online limit its applicability in high-frequency control loops and resource-constrained environments. This limitation is particularly acute for agile manipulator control requiring rapid response to dynamic task specifications.

We consider a 3-degree-of-freedom (3-DOF) robotic manipulator operating in a MuJoCo simulation environment. The control objective centers on driving the end-effector to track random 3D Cartesian targets within the robot's reachable workspace. While a hierarchical MPC controller combining inverse kinematics (IK) with online optimization achieves satisfactory performance, its computational burden restricts control frequencies and prohibits deployment on embedded systems.

Inspired by recent advances in imitation learning for control, we propose to distill the MPC policy into a neural network that operates directly on sequences of historical states. The primary contributions of this work include:

A complete data generation pipeline for collecting expert demonstrations from a hybrid IK-MPC controller An empirical evaluation of feedforward, recurrent, and attention-based architectures for policy approximation **Analysis of generalization, stability, and computational efficiency trade-offs** Experimental validation demonstrating real-time capable neural controllers achieving performance parity with the MPC expert

This paper is structured as follows: Section 2 formalizes the system dynamics and control problem. Section 3 details the baseline IK-MPC controller. Section 4 describes the dataset generation method-

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

ology. Section 5 presents the neural network architectures and training procedure. Section 6 discusses challenges and considerations. Section 7 outlines experimental results. Finally, Section 8 concludes with future research directions.

2 Problem Formulation

2.1 System Description

We consider a 3-degree-of-freedom (3-DOF) robotic manipulator with configuration defined by generalized coordinates $\boldsymbol{q} = [q_1, q_2, q_3]^\top \in \mathbb{R}^3$, representing joint angles, and their time derivatives $\dot{\boldsymbol{q}} \in \mathbb{R}^3$. The full observable state at discrete time step k is

$$\boldsymbol{x}_k = \left[\boldsymbol{q}_k^\top, \dot{\boldsymbol{q}}_k^\top\right]^\top \in \mathbb{R}^6 \tag{1}$$

The manipulator operates in a MuJoCo simulation environment governed by rigid-body dynamics with gravity compensation. The control objective is to drive the end-effector (EE) to track randomly sampled, reachable 3D Cartesian target positions $\boldsymbol{p}_{\{\text{des}\}} \in \mathbb{R}^3$ within the robot's workspace $\mathcal{W} \subset \mathbb{R}^3$.

3 Existing Controller: MPC with Inverse Kinematics

Our baseline controller employs a hierarchical architecture combining an Inverse Kinematics (IK) module and a Model Predictive Control (MPC) module. This structure decouples Cartesian-space task specification from joint-space optimal control.

3.1 Inverse Kinematics Formulation

The IK module translates desired end-effector positions into feasible joint-space configurations. Let $\boldsymbol{p}(\boldsymbol{q}) : \mathbb{R}^3 \to \mathbb{R}^3$ denote the forward kinematics mapping. The Cartesian error is defined as

$$\boldsymbol{e} = \boldsymbol{p}_{\{\text{des}\}} - \boldsymbol{p}(\boldsymbol{q}) \tag{2}$$

We solve the IK problem using the Jacobian transpose method with Damped Least Squares (DLS) for numerical stability near singularities. The iterative update rule is

$$\Delta\boldsymbol{q} = \boldsymbol{J}^\top(\boldsymbol{J}\boldsymbol{J}^\top + \lambda\boldsymbol{I})^{-1}\boldsymbol{e} \tag{3}$$

$$\boldsymbol{q}_{i+1} = \boldsymbol{q}_i + \alpha \cdot \Delta\boldsymbol{q} \tag{4}$$

where $\boldsymbol{J}(\boldsymbol{q}) = \mathrm{d}(\partial\boldsymbol{p}, \partial\boldsymbol{q}) \in \mathbb{R}^{3\times 3}$ is the geometric Jacobian, $\lambda > 0$ is the damping factor, and $\alpha \in (0, 1]$ is the step size. The iteration terminates when $\|\boldsymbol{e}\| < \varepsilon_{\{\text{tol}\}}$, yielding the reference configuration $\boldsymbol{q}_{\{\text{ref}\}}$.

3.2 Model Predictive Control Formulation

The MPC module receives $\boldsymbol{q}_{\{\text{ref}\}}$ and computes optimal control torques $\boldsymbol{\tau} \in \mathbb{R}^3$ over a finite prediction horizon. For prediction, we employ a simplified double-integrator model:

$$\dot{\boldsymbol{x}} = [[\dot{\boldsymbol{q}}], [\ddot{\boldsymbol{q}}]] = [[\dot{\boldsymbol{q}}], [\boldsymbol{\tau}]] \tag{5}$$

Discrete-time dynamics with sampling period Δt are

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \Delta t[\dot{\boldsymbol{q}}_k, \dot{\boldsymbol{\tau}}_k] =: f(\boldsymbol{x}_k, \boldsymbol{\tau}_k) \tag{6}$$

The MPC solves the following finite-horizon optimal control problem:

$$\min_{\boldsymbol{\tau}_{0:N-1}} \quad \sum_{k=0}^{N-1}\left(\left\|\boldsymbol{x}_k - \boldsymbol{x}_{\{\text{ref}\}}\right\|_{\boldsymbol{Q}}^2 + \|\boldsymbol{\tau}_k\|_{\boldsymbol{R}}^2\right) + \left\|\boldsymbol{x}_N - \boldsymbol{x}_{\{\text{ref}\}}\right\|_{\boldsymbol{Q}_N}^2 \tag{7}$$

subject to:

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{\tau}_k), \quad \boldsymbol{x}_0 = \boldsymbol{x}(t) \tag{8}$$

$$\boldsymbol{\tau}_{\min} \leq \boldsymbol{\tau}_k \leq \boldsymbol{\tau}_{\max} \tag{9}$$

where $\boldsymbol{x}_{\{\text{ref}\}} = \left[\boldsymbol{q}_{\{\text{ref}\}}^\top, \mathbf{0}^\top\right]^\top$ is the target state, and $\boldsymbol{Q}$, $\boldsymbol{R}$, $\boldsymbol{Q}_N \succ 0$ are weighting matrices. The optimization is performed using CasADi with IPOPT. The first control input $\boldsymbol{\tau}_0^*$ is applied to the system.

Limitation: The computational burden of iterative IK solving and online MPC optimization exceeds 50ms per control step on standard hardware, limiting control frequencies to approximately 20Hz and prohibiting deployment on embedded platforms.

4 Dataset Generation Pipeline

To enable imitation learning, we generate a dataset of expert demonstrations from the closed-loop IK-MPC controller. The dataset $\mathcal{D} = \left\{\left(\boldsymbol{X}_i, \boldsymbol{\tau}_i^{\{\text{MPC}\}}\right)\right\}_{i=1}^{M}$ consists of state sequence-action pairs, where $\boldsymbol{X}_i = [\boldsymbol{x}_{i-H+1}, ..., \boldsymbol{x}_{i-1}, \boldsymbol{x}_i] \in \mathbb{R}^{H\times 6}$ represents H consecutive states.

The data collection process proceeds as follows:

1. Target Sampling: Sample reachable end-effector target $\boldsymbol{p}_{\{\text{des}\}}\mathcal{W}$. Validate reachability using IK solver convergence within iteration budget $I_{\max}$.
2. Reference Calculation: Compute joint-space reference $\boldsymbol{q}_{\{\text{ref}\}}$ using the DLS IK algorithm.
3. Expert Demonstration: Execute the MPC controller for maximum episode length $T_{\max}$ or until $\left\|\boldsymbol{x}_k - \boldsymbol{x}_{\{\text{ref}\}}\right\| < \delta_{\{\text{success}\}}$.
4. Data Collection: Record state-action pairs $\left(\boldsymbol{x}_k, \boldsymbol{\tau}_k^{\{\text{MPC}\}}\right)$. After accumulating history buffer of length H, store tuples $\left(\boldsymbol{X}_k, \boldsymbol{\tau}_k^{\{\text{MPC}\}}\right)$. The gravity compensation term $\boldsymbol{\tau}_{g(\boldsymbol{q})}$ is explicitly excluded to learn only corrective control actions.

This process yields approximately 100,000 training examples after 5,000 episodes, requiring 72 hours of compute time on a 16-core workstation.

5 Neural Network Imitation of MPC Policy

We formulate the learning problem as minimizing the mean-squared error between the neural network policy π_θ and the expert MPC actions:

$$\min_{\theta} \quad \frac{1}{|\mathcal{D}|} \sum_{(\boldsymbol{X}_i, \boldsymbol{\tau}_i)\in\mathcal{D}} \left\|\pi_{\theta(\boldsymbol{X}_i)} - \boldsymbol{\tau}_i^{\{\text{MPC}\}}\right\|_2^2 \tag{10}$$

where $\pi_\theta : \mathbb{R}^{H\times 6} \to \mathbb{R}^3$ maps a sequence of historical states to control torques.

We investigate three architectural classes:

Feedforward Network (FFN): Flattens the state sequence into a vector $\boldsymbol{x}_\flat \in \mathbb{R}^{6H}$ and processes through L fully-connected layers with ReLU activations:

$$\pi_\theta^{\{\text{MPC}\}}(\boldsymbol{X}) = \boldsymbol{W}_L \cdot \text{ReLU}(\boldsymbol{W}_{L-1}\ \text{ReLU}(\boldsymbol{W}_1\boldsymbol{x}_\flat + \boldsymbol{b}_1) \cdot + \boldsymbol{b}_{L-1}) + \boldsymbol{b}_L \tag{11}$$

Recurrent Neural Network (RNN): Employs stacked LSTM or GRU layers to process the state sequence temporally. The final hidden state $\boldsymbol{h}_T$ is mapped to actions:

$$\boldsymbol{h}_t = \text{LSTM}(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}) \tag{12}$$

$$\pi_\theta^{\text{RNN}(\boldsymbol{X})} = \boldsymbol{W}_{\text{out}}\boldsymbol{h}_T + \boldsymbol{b}_{\text{out}} \tag{13}$$

Transformer: Utilizes multi-head self-attention mechanisms to model pairwise interactions across the entire history window. Positional encodings $\boldsymbol{P} \in \mathbb{R}^{H \times d}$ are added to embedded states $\boldsymbol{E} = [\boldsymbol{W}_e \boldsymbol{x}_{i-H+1}, ..., \boldsymbol{W}_e \boldsymbol{x}_i]^\top$ to preserve temporal information. After L transformer blocks, the output is mean-pooled and projected to action space:

$$\pi_\theta^{\text{Transformer}(\boldsymbol{X})} = \boldsymbol{W}_{\text{action}} \cdot \text{MeanPool}\Big(\text{Transformer}_{L(\boldsymbol{E}+\boldsymbol{P})}\Big) + \boldsymbol{b}_{\text{action}} \tag{14}$$

All networks are trained for 100 epochs using Adam optimizer with learning rate 10^-3 and batch size 256. We employ L2 regularization with coefficient 10^-4 and early stopping based on validation loss.

6 Key Challenges

Several fundamental challenges arise in this imitation learning paradigm:

Generalization: The policy must robustly handle state distributions outside the training manifold, particularly near workspace boundaries $\partial\mathcal{W}$ and kinematic singularities where $\det(\boldsymbol{J}\boldsymbol{J}^\top) \approx 0$. The MPC's performance degrades gracefully in these regions; the learned policy must emulate this behavior.

Stability and Safety: Unlike the constrained MPC, neural policies lack theoretical stability guarantees. The unconstrained nature of π_θ may generate high-frequency oscillations or infeasible torques. While the expert data respects $\boldsymbol{\tau}_{\min} \leq \boldsymbol{\tau}_k \leq \boldsymbol{\tau}_{\max}$, the learned policy may violate these constraints necessitating post-hoc saturation.

Data Efficiency: Data collection requires solving $N \times |\mathcal{D}|$ MPC problems, where N is the MPC horizon. For $|\mathcal{D}| = 10^5$ and $N = 20$, this entails two million optimization solves. This computational bottleneck necessitates sample-efficient architectures and transfer learning strategies.

Performance Parity: Achieving tracking errors $\|\boldsymbol{p}_{\mathbb{E}} - \boldsymbol{p}_{\text{des}}\|$ within 5% of the expert MPC while maintaining comparable settling times $t_{95\%}$ requires careful architectural design and training regularization. Performance degradation in end-effector orientation control (not addressed by position-only IK) remains an open challenge.

7 Expected Outcomes and Experimental Validation

The successful neural network policy should demonstrate:

Computational Efficiency: Inference time $<$ 1ms on an NVIDIA Jetson Xavier NX, enabling 500Hz control rates—25× faster than the MPC baseline's 20Hz at desktop-level performance.

Control Performance: Root-mean-square tracking error $\text{RMSE}_p = \sqrt{\frac{1}{T}\sum_{k=1}^{T} \left\|\boldsymbol{p}_{\mathbb{E},k} - \boldsymbol{p}_{\text{des}}\right\|^2}$ within 10% of the MPC expert across 1000 test targets uniformly sampled from $\mathcal{W}$.

Temporal Reasoning: Utilization of historical state information yields measurable performance improvements over memory-less policies, particularly for targets near singularities where momentum history informs better escape trajectories.

Generalization: Robust performance on out-of-distribution targets (e.g., near workspace boundaries or requiring joint-limit avoidance) with $< 15\%$ degradation in success rate.

We validate these outcomes through quantitative benchmarks comparing architectural variants, ablation studies on history length H, and sim-to-real transfer experiments using domain randomization on dynamics parameters (m_i, l_i, I_i).

8 Conclusion and Future Work

This work demonstrates the feasibility of distilling computationally expensive MPC policies into lightweight neural network controllers for 3-DOF manipulators. Our systematic evaluation of archi-

tectural choices provides guidance for selecting appropriate models based on task requirements and computational constraints.

Future research directions include:

Investigating adaptive history windows that expand when near singularities and contract otherwise Incorporating safety-critical constraints via neural network verification tools (e.g., ReLUplex) Extending to full 6-DOF pose control using neural IK solvers in the learning loop Developing active learning strategies to reduce data collection burden by 90% Exploring reinforcement learning fine-tuning to surpass expert performance

The methodology scales naturally to higher-DOF systems and more complex dynamics, offering a path toward real-time optimal control on embedded platforms.

References