

MASTER THESIS

Thesis submitted in fulfillment of the requirements for the degree of Master of Science in Engineering at the University of Applied Sciences Technikum Wien - Degree Program Robotics Engineering

Physics-Informed Inverse Robot Dynamics with Residual LSTM Modeling

By: Moritz Dönges, BSc

Student Number: 2310331024

Supervisors: Michael Schebek, MSc

FH-Prof. Dr.techn. Mohamed Aburaia,
MSc

Vienna, January 31, 2026

Declaration

“As author and creator of this work to hand, I confirm with my signature knowledge of the relevant copyright regulations governed by higher education acts (see Urheberrechtsgesetz /Austrian copyright law as amended as well as the Statute on Studies Act Provisions / Examination Regulations of the UAS Technikum Wien as amended).

I hereby declare that I completed the present work independently and that any ideas, whether written by others or by myself, have been fully sourced and referenced. I am aware of any consequences I may face on the part of the degree program director if there should be evidence of missing autonomy and independence or evidence of any intent to fraudulently achieve a pass mark for this work (see Statute on Studies Act Provisions / Examination Regulations of the UAS Technikum Wien as amended).

I further declare that up to this date I have not published the work to hand nor have I presented it to another examination board in the same or similar form. I affirm that the version submitted matches the version in the upload tool.“

Vienna, January 31, 2026

Signature

Kurzfassung

In der kollaborativen Robotik erfordern Online-Nutzlasterkennung und sichere Interaktion genaue Inversdynamikmodelle bei eingeschränkter Sensorik. Diese Arbeit untersucht das physikinspirierte Lernen der Gelenkaktuation aus propriozeptiven Signalen, wenn nur Motorstrommessungen verfügbar sind. Es wird eine zweistufige Pipeline vorgeschlagen: Ein Deep Lagrangian Network (DeLaN) lernt ein mechanikkonsistentes Basismodell für die Motorströme aus Gelenkpositionen, -geschwindigkeiten und -beschleunigungen, und ein LSTM-Sequenzmodell wird auf dem Residuum trainiert, um geschichtsabhängige Effekte abzubilden. Dazu werden ein Trajektorien-basiertes Vorverarbeitungs- und Split-Protokoll sowie ein Best-Model-Selektionsschema eingeführt, das Validierungsgenauigkeit und Seed-Stabilität ausbalanciert. Experimente auf den IEEE-DataPort-Datensätzen UR3e/UR10e quantifizieren, wie die Trajektorienabdeckung Konvergenz und Generalisierung beeinflusst, und benchmarken den Ansatz gegen die publizierte lineare Identifikations-Baseline. Die Ergebnisse zeigen, dass eine höhere Anzahl von Trajektorien die Variabilität über Datensatz-Seeds reduziert und dass das Residual-LSTM den verbleibenden Motorstromfehler des DeLaN-Baselinesystems systematisch verringert. Bei benchmark-großen Splits (50,000 Trainings- und 5,000 ungesehene Test-Samples) übertrifft das vorgeschlagene DeLaN+LSTM-Modell die Baseline unter Last deutlich und verbessert die Performance auf dem größeren Roboter, während die Baseline im unbelasteten UR3e-Fall weiterhin am stärksten ist. Die Arbeit stellt eine durchgängige, containerisierte Implementierung und ein Reporting-Protokoll für reproduzierbares, physikstrukturiertes Inversdynamik-Lernen im Motorstrombereich bereit.

Schlagworte: DeLaN, LSTM, Motor Strom Schätzung, Newton/Euler, UR3 Robot, Inverse Kinematik

Abstract

In collaborative robotics, online payload estimation and safe interaction require accurate inverse-dynamics models under limited sensing. This thesis investigates physics-informed learning of joint actuation from proprioceptive signals when only motor-current measurements are available. A two-stage pipeline is proposed: a Deep Lagrangian Network (DeLaN) learns a mechanics-consistent baseline motor-current model from joint positions, velocities, and accelerations, and an LSTM sequence model is trained on the residual to capture history-dependent effects. A trajectory-level preprocessing and split protocol, together with a best-model selection scheme that balances validation accuracy and seed stability, is introduced. Experiments on the IEEE DataPort UR3e/UR10e datasets quantify how trajectory coverage affects convergence and generalisation and benchmark the approach against the published linear identification baseline. The results show that increasing the number of trajectories reduces variability across dataset seeds and that the residual LSTM systematically decreases the remaining motor-current error of the DeLaN baseline. On benchmark-scale splits (50,000 training and 5,000 unseen test samples), the proposed DeLaN+LSTM model substantially outperforms the baseline under load and improves performance on the larger robot, while the baseline remains strongest on the unloaded UR3e condition. The work provides an end-to-end, containerised implementation and reporting protocol for reproducible physics-structured inverse-dynamics learning in the motor-current domain.

Keywords: DeLaN, LSTM, Motor Current Estimation, Newton/Euler, UR5 Robot, Inverse Kinematic

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Problem Description	2
1.3	Aim of Work	4
2	State of the Art	5
2.1	Research Strategy	5
2.2	Literature	7
2.3	Limitations of the Current State of the Art	12
2.4	Related Work	15
2.5	Deep Lagrangian Networks	15
2.6	Long-Short-Term-Memory	15
3	Methods	17
3.1	Stage 1: Structured inverse dynamics for robot + fixed gripper	17
3.2	Stage 2: Sequence model for residual joint torques	18
3.3	Dataset of Collaborative Robots	20
3.4	Data preprocessing and dataset construction	21
3.5	DeLaN training setup	23
3.6	Long-Short-Term-Memory Training Setup	25
4	Experimental Setup	27
4.1	DeLaN + LSTM - Learning Curve Stroy	27
4.1.1	K-Domination	27
4.1.2	Best Model Approach	29
5	Experimental Results	34
5.1	Learning Curve Results	34
5.1.1	K-Domination Results	34
5.1.2	Best Model Approach Results	38
5.2	Performance Evaluation to Baseline	48
6	Implementation	52
6.1	Execution Environment (Docker)	53
6.2	Implementation of the Two-Stage Pipeline	53
6.3	Experimental Orchestration (Sweeps)	54
7	Discussion	54

8 Summary and Outlook	58
Bibliography	60
List of Figures	64
List of Tables	67
List of source codes	68
A Kinematic and Dynamic Background of Robot Manipulation and Environment Interaction	69
B Query Categories-Index Terms	70
C Concept Graph	75

1 Introduction

This chapter provides the common thread of the work and positions it within the broader field of robotic manipulation and human-robot collaboration. First the Motivation of the study is developed, followed by the Problem Description and the resulting Aim of the Work. Subsequent chapters present the State of the Art, a formal Problem Statement, the Related Work and proposed methods, the experimental setup and evaluation, a discussion of the results and their implications, and an outlook on future research directions.

1.1 Motivation

The motivation for this work is classified in a context and a use case, where the context outlines the growing role of industrial and collaborative manipulators, while the use case specifies a concrete manipulation scenario that requires accurate online identification of robot and payload parameters.

As the robotics industry grows year over year, so does the number of robots operating around the world. It is estimated that there were approximately 3.4 million industrial robots in use worldwide in 2023 [1]. At the same time, the number of newly installed industrial robots has been increasing steadily since 2014. Between 2021 and 2024, around 541 000 new industrial robots were installed per year [2]. Within this landscape, collaborative robots (cobots) represent about 10.5% of the industrial robot market, with 57 040 new units deployed in 2023, and annual cobot installations since 2020, 2022, and 2023 reaching roughly 50 000 units per year. Importantly, these cobots are expected to complement rather than replace traditional industrial robots [3].

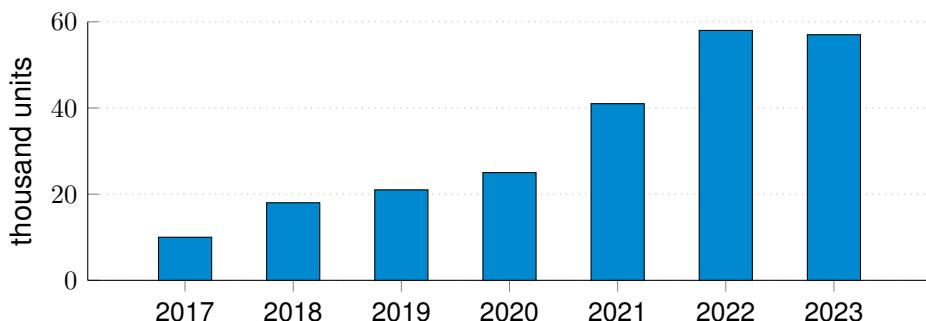


Figure 1: Global annual installations of collaborative robots from 2017 to 2023 (in thousand units). Figure modified from [3].

The growing deployment of, and increasing collaboration with, robots imposes stringent requirements on safety and performance. As tasks become more complex and humans and

robots share workspaces more closely, two closely related problems become central: safe manipulation of payloads and safe physical human-robot interaction [4–6]. Addressing both problems requires accurate knowledge of the inertial parameters of the manipulated object together with consistent estimation of the robot’s dynamic state and interaction forces [7–10]. A collaborative robot must therefore maintain an internal representation of the mass-inertia properties of the payload or tool it manipulates and of the forces exchanged with its environment. This dynamic awareness is a prerequisite for compliant, contact-rich behaviour and for precise, high-performance manipulation in close proximity to humans [11–15].

The considerations above motivate a concrete use case in which a collaborative robotic arm must manipulate previously unseen objects in a shared workspace. A vision system can provide geometric information such as shape and dimensions of the payload, but it does not directly reveal its mass, center of mass (CoM), or inertia tensor. For safe and precise execution of contact-rich tasks, however, these inertial properties are indispensable.

In practice, the only viable way to obtain this information during operation is to exploit the robot’s own sensor data, such as joint positions, velocities and accelerations, motor currents/torques, and optionally wrist force/torque measurements. From these signals, one can estimate both the robot’s rigid-body parameters and the inertial properties of the attached payload. This leads to the dual identification problem of robot dynamic parameter identification (RDPI) and payload dynamic parameter identification (PDPI).

The targeted application scenario comprises typical industrial and collaborative tasks such as pick-and-place, human-assisted manipulation, and precise tool use. In all these cases, RDPI and PDPI must be performed online so that the controller maintains an up-to-date model of the combined robot-payload dynamics and the resulting contact forces. Robust online identification methods are therefore a key enabling technology for safe human-robot collaboration and high-performance manipulation with arbitrary payloads and tools.

1.2 Problem Description

The following kinematic and dynamic background of robot manipulation analyses why endowing a robotic manipulator with awareness of its own dynamics, payload, and tools is mathematically demanding and cannot be achieved by simple calculation or direct measurement alone. The inertial properties of a rigid body are collected in the standard 10-dimensional parameter vector

$$\phi^T = \begin{bmatrix} m & mc_x & mc_y & mc_z & J_{xx} & J_{xy} & J_{xz} & J_{yy} & J_{yz} & J_{zz} \end{bmatrix} \in \mathbb{R}^{10}, \quad (1)$$

which enters the Newton-Euler equations

$$\begin{bmatrix} \mathbf{f} \\ \boldsymbol{\tau} \end{bmatrix} = m \begin{bmatrix} \mathbf{I}_{3 \times 3} & -[\mathbf{c}]^\times \\ [\mathbf{c}]^\times & \mathbf{J}_s \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \boldsymbol{\alpha} \end{bmatrix} + \begin{bmatrix} m[\boldsymbol{\omega}]^\times [\boldsymbol{\omega}]^\times \mathbf{c} \\ [\boldsymbol{\omega}]^\times \mathbf{J}_s \boldsymbol{\omega} \end{bmatrix}, \quad (2)$$

so that the wrench $(\mathbf{f}, \boldsymbol{\tau})$ depends nonlinearly on the motion $(\mathbf{a}, \boldsymbol{\alpha}, \boldsymbol{\omega})$ but linearly on ϕ .

For the equipment rigidly attached to the tool flange (gripper/tool, with or without payload/load) we define an effective rigid-body parameter vector

$$\phi_{\text{eff}} = \begin{cases} \phi_{\text{tool}}, & \text{no load,} \\ \phi_{\text{tool}} + \phi_{\text{load}}, & \text{with load,} \end{cases} \quad (3)$$

which acts on top of the nominal robot dynamics. In contrast, with a clean flange (no tool/no load) only the robot parameters ϕ_{robot} contribute to the system dynamics.

The robot structure itself is described by its own parameter vector ϕ_{robot} , which enters the standard joint-space rigid-body dynamics. We denote this contribution by τ_{robot} (clean flange),

$$\tau_{\text{robot}} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) + \tau_f(\dot{\mathbf{q}}), \quad (4)$$

where $\tau_f(\dot{\mathbf{q}})$ models joint-level non-idealities such as Coulomb and viscous friction, possible Stribeck effects, and drive-train phenomena like backlash.

The wrench generated by the effective rigid body at the flange induces an additional joint-space torque

$$\tau_{\text{ext}} = \mathbf{J}^T(\mathbf{q}) \vec{F}_{\text{ext}}(\phi_{\text{eff}}), \quad (5)$$

where $\mathbf{J}(\mathbf{q})$ is the end-effector Jacobian. In the clean-flange case (no tool/no load), \vec{F}_{ext} reduces to purely external interaction forces with the environment (e.g. contacts or collisions).

The motor torques are therefore

$$\tau_{\text{motor}} = \tau_{\text{robot}} + \tau_{\text{ext}}(\phi_{\text{eff}}), \quad (6)$$

and for brushless DC actuators with torque constant k_t one obtains the current-torque relation

$$\tau_{\text{motor}} = k_t \mathbf{I} \Rightarrow \mathbf{I} = \frac{\tau_{\text{robot}} + \tau_{\text{ext}}(\phi_{\text{eff}})}{k_t}. \quad (7)$$

If a force/torque sensor is mounted at the flange, the measured wrench can be written, using the relations derived in the Appendix A, as

$$\vec{F}_{\text{measured}} = Y(\mathbf{a}, \boldsymbol{\alpha}, \boldsymbol{\omega}) \phi_{\text{eff}} + \vec{F}_{\text{bias}} + \vec{n}, \quad (8)$$

where $Y(\cdot)$ is the 6×10 Newton-Euler regressor matrix defined in the Appendix B. It is linear in the inertial parameter vector ϕ_{eff} , but depends nonlinearly on the motion variables $(\mathbf{a}, \boldsymbol{\alpha}, \boldsymbol{\omega})$. The terms \vec{F}_{bias} and \vec{n} denote sensor bias and noise, respectively. The motion variables $(\mathbf{a}, \boldsymbol{\alpha}, \boldsymbol{\omega})$ are in turn determined by the joint state and motor torques through the nonlinear dynamics(4)-(7).

From an identification viewpoint, this creates two tightly coupled challenges. First, all available measurements (joint currents, positions, velocities and flange wrench) depend on the combined dynamics of robot, tool and load via the nonlinear relationships (4)-(8), so the contribution of the load parameters ϕ_{load} cannot be isolated by simple computation or direct measurement. Second, accurate payload or load dynamic parameter identification (PDPI) presupposes an equally accurate compensation of the underlying robot-tool dynamics, including unmodelled effects such as friction and joint transmission nonlinearities. Together, these aspects make dynamic awareness of payload, tool and robot a mathematically demanding inverse problem rather than a straightforward calculation from geometric or sensor data.

1.3 Aim of Work

The SoA review highlights two persistent challenges for dynamic awareness in collaborative manipulation. First, accurate inverse-dynamics and friction compensation are often obtained in calibration-centric workflows that depend on carefully designed excitation and repeated identification runs. Second, hard-to-model effects (e.g., frictional memory and transmission nonlinearities) are frequently absorbed into lumped residual terms or task-specific models. Recent physics-informed architectures such as Deep Lagrangian Networks (DeLaN) provide a mechanics-consistent model class for inverse dynamics [16, 17] and have been extended towards motor-side supervision when joint-torque measurements are unavailable [18]. Complementarily, deep sequence models, in particular LSTMs, have been shown to be effective residual learners for dynamics and force/torque estimation from proprioceptive histories [19, 20].

Aligned with the limitations identified in Chapter 2, the aim of this thesis is to develop and evaluate a physics-informed, sequence-model-based inverse-dynamics pipeline that learns a single nominal actuation model from encoder and motor-current data, while separating structured rigid-body dynamics from history-dependent residual effects. This nominal model is intended to provide a foundation for tool/gripper compensation and subsequent payload dynamic parameter identification (PDPI) as formulated in Section ???. Although the structured inverse-dynamics formulation can, in principle, be interpreted in the end-effector wrench measurement frame via Jacobian mapping, the evaluation in this thesis is conducted purely in joint space in the measured actuation domain of motor currents. Concretely, the work pursues the following objectives:

- Learn a DeLaN-based baseline predictor of per-joint motor currents from joint kinematics, using a parameterisation that respects key mechanical structure (e.g., positive-definite inertia) and serves as a consistent nominal model [16, 17].
- Freeze the baseline and learn the remaining motor-current residual dynamics with an LSTM sequence model over a finite history window, capturing effects not explained by the instantaneous state alone [19, 20].
- Establish a trajectory-level split and model-selection protocol, and benchmark DeLaN and DeLaN+LSTM against the dataset baseline on the IEEE DataPort trajectories, reporting per-joint RMSE in current units [A] [21].

Accordingly, the contributions of this work are a motor-current-domain formulation of a physics-structured DeLaN baseline. Combined with an LSTM residual model into a single inverse-dynamics predictor and an evaluation and best-model selection procedure, that quantifies accuracy and stability under trajectory-level splits. An implementation that reproduces preprocessing, training, and evaluation end-to-end within a containerised pipeline.

RQ1 How accurately does a physics-structured DeLaN baseline predict per-joint motor currents on held-out trajectories, and how does trajectory coverage affect stability and generalisation (evaluated by per-joint motor-current RMSE in A on validation/test splits, reported as median \pm IQR across seeds)?

RQ2 To what extent does an LSTM residual model reduce the remaining motor-current prediction error, and how do feature choice and history length influence accuracy and overfitting behaviour (evaluated by residual and combined motor-current RMSE in A on the test split, and by the overfit indicator $\mathcal{L}_{\text{val}}/\mathcal{L}_{\text{train}}$)?

RQ3 How does the proposed DeLaN+LSTM pipeline compare to the classical model-based identification baseline on the IEEE DataPort benchmark datasets and operating conditions [21] (evaluated by per-joint motor-current RMSE in A on the 5,000-sample held-out test split and relative RMSE change versus the baseline)?

2 State of the Art

2.1 Research Strategy

The literature search was organised around five content clusters C_1, \dots, C_5 and the goal/context term sets C_{mt} and C_{ct} . The clusters capture the main methodological families, while C_{mt} and C_{ct} constrain the queries to estimation-related objectives in robotic manipulation:

- C_1 = Classical / Observers
- C_2 = Gaussian Process (GP)
- C_3 = Deep Sequence Models (MLP / GRU / TCN / Transformer / LSTM)
- C_4 = Physics-Informed / Differentiable
- C_5 = Surveys
- C_T = Goal & Domain Terms
 - C_{mt} = Estimation & Modeling Terms
 - C_{ct} = Robotics Context Terms

The detailed index terms associated with each set are listed in Appendix B. For each content cluster C_i , a family of queries Q_i was constructed by combining (disjunctions of) its index terms with estimation & modelling terms from C_{mt} and robotics context terms from C_{ct} . Figure 2 illustrates this logic schematically as a generalised set intersection over the three term groups.

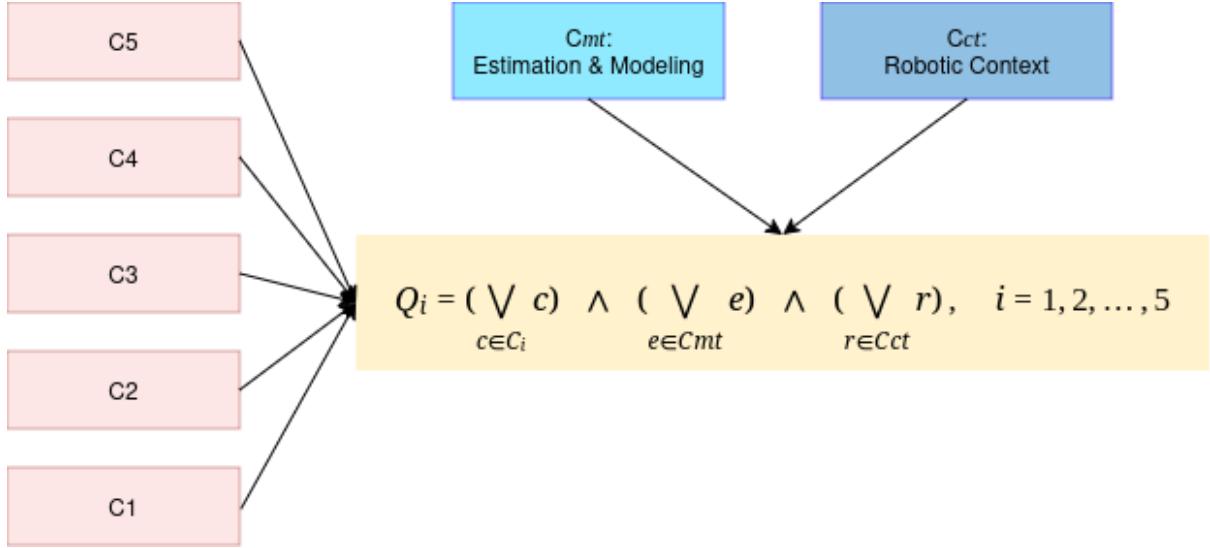


Figure 2: Query logic used to categorise the SoA papers. Each category Q_i is formed by combining content clusters C_i with estimation & modelling terms C_{mt} and robotics context terms C_{ct} . The combined representation C and query set Q are formed by the union of their respective subsets.

This process yielded 36 papers that are directly relevant to robot and payload dynamics, interaction force estimation and related identification problems. Table 1 summarises how these works are distributed across the five query categories and counts, for each category, how many papers address payload dynamics, robot rigid-body dynamics, and contact-force estimation (papers that treat payload and rigid-body dynamics contribute to both columns). The corresponding relations are visualised in the concept graph in Fig. 4.

Table 1: Overview of query results by category.

Query	Relev. SoA	Payload	Rigid-body	Contact Force
Q_1 = Classical / Observers	17	10	8	3
Q_2 = Gaussian Process (GP)	4	0	1	3
Q_3 = Deep Sequence Models	8	4	3	1
Q_4 = Physics-Informed / Diff.	5	0	5	0
Q_5 = Surveys	2	—	—	—
Total	36	14	17	7

2.2 Literature

In the context of robotic manipulator dynamics and payload identification, existing methods largely fall into the four methodological families reproduced by Lutter et al. [16] and illustrated in Figure 3.

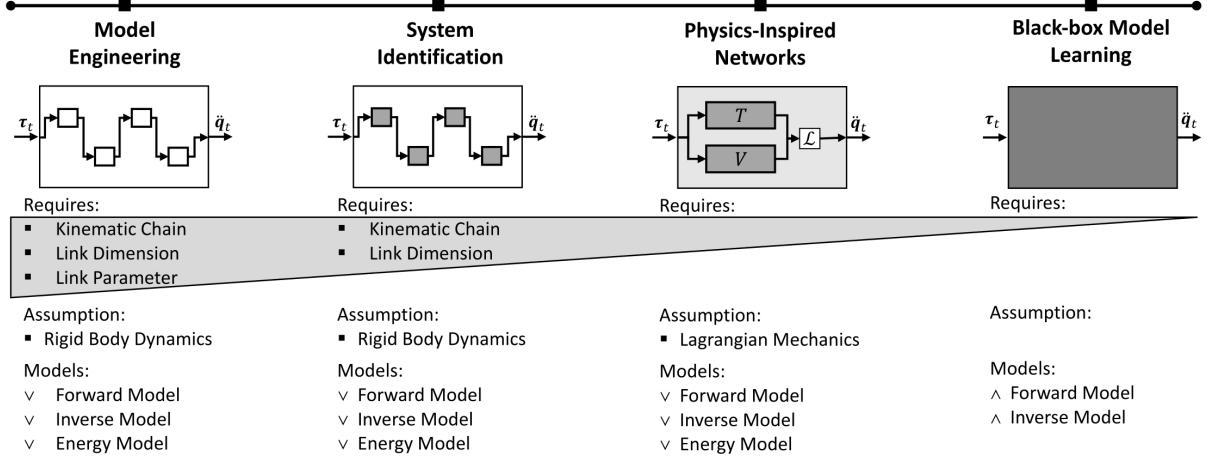


Figure 3: Overview of modelling paradigms for robot dynamics (reproduced by Lutter et al. [16]). The four panels correspond to the approaches discussed in this SoA: classical rigid-body model engineering and system identification (Category Q1), physics-inspired networks such as DeLaN/PINNs (Category Q4), and black-box model learning with deep networks (Category Q3). Gaussian-process residual models (Category Q2) sit between system identification and black-box learning.

The Category Q1 ‘Classical / Observers’ groups classical model-based methods for robot and payload dynamics and interaction force estimation, mostly based on linearly parameterised rigid-body dynamics (RBD) and LS/WLS-type Newton-Euler regressors, sometimes combined with observers and Kalman filters [4–8, 11–15, 22–27]. Across these works, three main lines of research can be distinguished in payload dynamic parameter identification (PDPI) using force/torque (FT) sensing [4, 6, 11, 14], robot and payload dynamic parameter identification in joint or motor-current space without FT sensors [12, 13, 15, 23, 25, 26], and observer-based sensorless force/torque estimation and online payload identification using proprioceptive data [5, 7, 8, 22, 24, 27]. Across Q1, mass is usually identified accurately, CoM moderately well, and inertia emerges as the hardest quantity to estimate robustly [4, 6, 11, 14, 26].

A first group of methods performs PDPI directly in the FT frame [4, 6, 11, 14]. They typically exploit static poses to identify the payload mass and centre of mass, and then use dedicated dynamic excitation trajectories together with LS or TLS-type Newton-Euler regressors to estimate the inertia tensor. Representative works demonstrate that, given a sufficiently informative excitation and an FT sensor rigidly mounted at the flange, payload mass can be recovered very accurately and CoM can be estimated with reasonable precision, even for heavy payloads [4, 11]. However, inertia estimates are systematically more fragile, especially under cobot-typical safety constraints with short trajectories and low accelerations and in several cases quantitative ground truth for CoM and inertia is missing or only partially available (validation is often given in terms of residual gravitational/inertial wrench after compensation rather than direct parameter error) [4, 6, 11, 14]. Moreover, these approaches require additional FT hardware and considerable experimental effort in the form of carefully designed calibration motions.

A second group tackles robot dynamic parameter identification (RDPI) and PDPI in joint space or motor-current space without FT sensors [12, 13, 15, 23, 25, 26]. Here, fully or partially decoupled identification schemes are designed to separate gravitational, frictional and inertial effects, often using families of S-curve or Fourier trajectories executed both with and without payload [12, 13, 15, 23]. Double-weighted WLS and optimisation-enhanced LS methods achieve very accurate joint-torque prediction and good agreement with CAD-based payload models, confirming that classical LS/NE pipelines, for example by Swevers et al. [15] remain a strong baseline for RDPI and PDPI [12, 13, 15, 23, 25, 26]. At the same time, these methods are typically executed in dedicated calibration phases, rely on repeated execution of long and highly exciting trajectories and payload parameters are updated only between such identification runs. They therefore provide an excellent commissioning tool and basis model, but do not by themselves endow the robot with continuous online awareness of payload changes during manipulation tasks.

A third line of work focuses on sensorless estimation of external joint torques and end-effector wrenches using observers and filters [7, 8, 22, 24, 27]. Momentum observers, higher-order sliding-mode observers, adaptive Kalman filters and high-order finite-time observers use a nominal RBD model together with controller torques and joint measurements to reconstruct external forces, sometimes with probabilistic covariance information. These methods achieve good performance in collision detection, binary contact decisions and execution monitoring, and some approaches augment classical friction models with learned nonlinear terms such as

neural-network Stribeck approximations [7, 8, 27]. Nevertheless, their accuracy depends critically on the quality of the underlying RBD and friction models, and residual force errors remain significant in highly dynamic phases or around velocity reversals [7, 24, 27]. Importantly, most observer-based schemes treat payloads and tools as fixed parts of the nominal model or as lumped disturbances, and do not explicitly estimate payload parameters.

More recent contributions bridge RDPI/PDPI and observer-based estimation by using proprioceptive data to identify payload parameters online [5, 26]. Momentum-observer-based schemes and parameter-difference methods compute external joint torques as residuals between measured and model-based torques and apply LS/RLS Newton-Euler regressors to recover payload mass, CoM and, in some cases, inertia during regular robot operation. These works demonstrate that accurate online PDPI is possible without FT sensors, provided that a reasonably accurate base robot model, friction compensation and sufficiently exciting motions are available [5, 26]. At the same time, they underline several structural limitations: inertia remains the hardest quantity to identify robustly. Nonlinear friction, backlash and transmission effects must be modelled or learned carefully (with several authors explicitly noting residual error peaks near motion reversal due to unmodelled friction [15, 25]). and the resulting estimators not providing a unified, continuously updated representation of robot and load.

The Category Q2 ‘Gaussian Process (GP)’ groups Gaussian Process (GP) and GP-hybrid methods for inverse dynamics and sensorless contact estimation. In all cases, a GP is trained offline as a residual or surrogate model on top of a nominal rigid-body dynamics (RBD) description, and then deployed online for torque or disturbance prediction [28, 29]

Two studies use GPs to improve joint-space contact-force estimation. In [28], an enhanced GP learns the residual dynamics between an Euler-Lagrange model and measured torques. This residual is injected into a decoupling disturbance observer and Kalman filter to obtain external joint torques and end-effector forces. The follow-up [30] extends this to a GP-adaptive disturbance Kalman filter, where the disturbance covariance is adjusted based on the GP output. Both papers show reduced estimation error and faster convergence than purely model-based observers, but require a reasonably accurate nominal model, high-quality proprioception and extensive non-contact training data. Payload and tool effects are absorbed into a single residual term.

A third work combines GP inverse dynamics with learning-based contact detection [29]. A GP predicts non-contact motor torques from joint states, and the residual between measured and GP-predicted torques is passed to a convolutional neural network that classifies contact vs. no-contact during assembly tasks, achieving high classification accuracy on scripted collisions. The method remains task-specific and does not recover physical interaction forces or payload parameters.

Finally, [31] compares GPs and neural networks for inverse dynamics when the inputs include physics-inspired features (e.g. nominal RBD torques). Embedding such structure improves data efficiency and prediction accuracy and GPs are competitive for moderate dataset sizes.

Deep Sequence Models comprises deep-learning approaches for inverse dynamics, force estimation and payload identification. Most methods either learn residual dynamics on top of a nominal rigid-body model or learn a direct mapping from joint histories to payload parameters or contact indicators, using LSTM/GRU-type sequence models, feed-forward networks, CNNs or ensemble methods [10, 19, 20, 32–36].

A first group focuses on deep residual inverse dynamics on top of a nominal RBD model. In [19], the authors use the public Franka Panda dataset and the Gaz et al. model to compute joint-torque residuals between data and RBD prediction, and train a bootstrapped LSTM ensemble (BLL-LSTM) on sequences of (q, \dot{q}, \ddot{q}) to predict these residuals. The ensemble clearly improves torque prediction over Gaussian processes and single models on held-out dataset splits, but remains a purely offline, dataset-based study that assumes a reasonably accurate full robot model. Similarly, [20] trains LSTMs to map base FT-wrench and joint states to end-effector tip forces, and joint states to joint torques, on a 7-DoF Panda. The LSTMs outperform MLPs, 1D convolutions and a DeLaN baseline in both simulation and real experiments, but are evaluated in a task-specific setting and rely on FT hardware and simulation-generated ground-truth forces.

A second group uses deep models to infer end-effector wrench or contact directly from proprioception. The adaptive sparse GRNN force observer in [32] maps joint positions, velocities, accelerations and motor currents to 6D wrench on a UR5 teleoperation system, using an FT sensor only for supervision. It achieves strong soft/stiff collision force estimation and outperforms GP and MLP baselines. In [33], a CNN is trained in IsaacGym with domain randomisation to detect and localise link-environment contacts for a 7-DoF Panda using only link velocities and pose errors (no torque or FT sensing). The network reaches around 98 % accuracy in sim-to-real contact localisation, but does not estimate wrench magnitude or payload dynamics.

The third line of work addresses payload dynamic parameter identification (PDPI) with deep networks and ensembles. A learning-based method for the OpenMANIPULATOR-X in [10] combines a nominal RBD model and camera pose estimation with an MLP that processes joint states and velocity sign to estimate per-joint torque contributions. A subsequent LS step recovers payload mass and CoM of known objects, with average errors of about 9 % in mass and 18 % in CoM. For collaborative cobots,[35] proposes a batch ensemble of weak learners (NNs or decision trees) that directly map (q, \dot{q}, τ) to payload parameters for a library of 77 payloads along a fixed excitation trajectory on a Franka robot. The method achieves good sim-to-real transfer and clearly reduces mass/CoM error compared to RLS, but still requires a dedicated excitation path per payload. This is generalised in [34, 36], which introduce incremental ensemble learning for PDPI along arbitrary task paths. The initial incremental ensemble [34] adapts weak learners online based on Euclidean distance in feature space but suffers from catastrophic forgetting on previously seen trajectories. The follow-up work [36] adds a bag-based classifier that routes new path segments to the most relevant weak learner and spawns new learners as required, thereby preserving accuracy on old paths while adapting to new ones and eliminating the explicit excitation trajectory. The price is a growing ensemble size and the fact that the individual learners remain small feed-forward networks without explicit temporal structure.

Physics-informed and differentiable dynamics models group Category Q4, where deep networks are structured by Lagrangian or state-space physics and trained on joint data to predict torques or currents. The central goal in all works is accurate robot dynamic parameter identification (RDPI) and inverse dynamics prediction for fixed robots. Payload dynamics and interaction forces are not estimated explicitly.

A first line of work builds on Deep Lagrangian Networks (DeLaN) and related continuous-time models. The survey and benchmark in [16] compares structured DeLaN/HNN models against black-box neural networks on low-DoF systems and a 4-DoF WAM arm, showing that physics-informed architectures achieve lower normalised errors and much longer valid prediction horizons than unconstrained networks. However, these models assume conservative dynamics without contacts, and friction is either neglected or handled separately. Extending this idea, [18] learns an “extended DeLaN” that incorporates motor couplings and friction on a UR10e arm, using joint positions, velocities and accelerations together with motor currents. The network learns Lagrangian terms plus actuator/friction parameters and predicts motor currents with high accuracy in simulation and on hardware, outperforming the original DeLaN and a feed-forward baseline, yet still under fixed tool/payload and contact-free conditions.

A second line of work uses physics-informed neural networks (PINNs) as refinements of classical LS/NE identification. In [37], base parameters are first obtained by a Newton-Euler regressor. A decomposed PINN then minimises a hybrid loss combining data residuals and the rigid-body dynamics equations, reducing joint-torque RMSE compared to LS alone. Building on this, [38] proposes a friction-inclusive PINN for multi-joint industrial robots without joint torque sensors, combining Lagrangian dynamics with an explicit Stribeck friction model and a history-based residual network that learns remaining errors over a time window. The resulting hybrid model achieves very strong joint-torque (or current) prediction across several joints and outperforms DeLaN-type and LS baselines. Finally, [39] introduces an H-PINN for a single collaborative robot joint, embedding the joint’s state-space dynamics into an RNN cell and jointly estimating physical parameters and state transitions, with highly accurate joint-level dynamics prediction in simulation and experiments.

Taken together, model-based methods can deliver high-quality RDPI and PDPI, as well as useful sensorless interaction-force estimates, but typically only under carefully controlled excitation of dedicated identification trajectories [4, 6, 11–13, 15, 23–27]. From the perspective of this work, the main gaps are the lack of a unified, online notion of dynamic awareness that covers robot and payload; the persistent difficulty of reliably identifying and exploiting payload inertia in cobot-safe regimes; and the sensitivity of existing approaches to friction and transmission nonlinearities [5, 7].

Gaussian Processes show up as effective residual models for unmodelled robot dynamics and can enhance sensorless contact estimation [28–31]. However, the GP is always a lumped compensator: there is no sequence-aware handling of backlash or history-dependent friction, and no unified dynamic representation in the measurement frame that could serve as a precise, task-agnostic basis for tool/gripper compensation.

The literature shows that deep models can substantially improve torque and force estimation and can achieve competitive PDPI, including sim-to-real transfer for collaborative robots. At the same time, existing works either depend on accurate nominal models and FT supervision, or operate as largely black-box payload regressors, and none provide a unified deep sequence model that simultaneously delivers joint-torque prediction, tool/payload compensation and interaction-force awareness during general manipulation tasks.

Across Physics-Informed and Differential Models, physics-informed deep models consistently improve inverse-dynamics prediction and friction compensation compared to purely black-box networks, while using only encoder and motor data [16, 18, 37–39]. At the same time, they are trained on carefully designed excitation trajectories and then deployed as fixed models, without explicit treatment of changing payloads or contact forces, and some architectures become quite complex when scaling beyond low-DoF setups or single joints. Thus, physical informed and differential models provides strong structured baselines for fixed robot dynamics, but does not yet realise an online, unified dynamic awareness of robot, tool and payload with explicit force estimation, as targeted in this work.

Overall, the literature spans the full spectrum in Fig. 3: from classical model engineering and system identification, through GP residual models, to black-box deep learning and physics-inspired networks. Classical rigid body dynamic observer and model based methods provide strong RDPI/PDPI under carefully designed calibration trajectories, but remain sensitive to friction and payload changes and rarely yield a unified, online notion of the effective rigid body. Gaussian process and deep black-box models are robust as residual models to cover unmodelled dynamics and contact, yet largely treat friction, payload and interaction forces as a single residual and do not produce a physics-consistent wrench model in the measurement frame. Physics-informed DeLaN/PINN approaches in the fourth category bridge these extremes by embedding Lagrangian structure and achieving joint-torque prediction from proprioception alone, but are typically trained for fixed tools and evaluated only in joint space. This thesis therefore positions itself in the “physics-inspired networks” quadrant of Fig. 3, combining a DeLaN-style backbone with an LSTM residual model to obtain a single joint-space inverse-dynamics model that also serves as a reliable nominal wrench model in the measurement frame and a basis for PDPI.

2.3 Limitations of the Current State of the Art

The reviewed literature reveals several recurring limitations that motivate the approach developed in this thesis:

- **Calibration-centric identification and limited continuity across tasks.** Accurate RDPI/PDPI results are commonly obtained in dedicated calibration phases using carefully designed excitation trajectories and repeated with/without-payload experiments [4, 10–13, 15, 23, 26, 34–36]. Even online variants presuppose a sufficiently accurate nominal model and friction compensation [5, 26]. This leaves a gap for methods that provide a single, continuously usable nominal dynamics model during general manipulation tasks.

- **Incomplete treatment of payload inertia and the effective rigid body.** In Q1, payload mass (and often CoM) can be estimated accurately, but payload inertia is frequently weakly excited, ill-conditioned, or only partially validated, especially in cobot-safe regimes [4, 6, 11, 14, 26]. Learning-based approaches in Categories Q2–Q4 largely assume fixed tools/payloads and therefore do not maintain an explicit, continuously updated effective rigid-body representation [18, 20, 28, 31, 32, 37, 38].
- **Entangled treatment of friction, transmission, and contact effects.** Observer-based schemes and LS/NE identification remain sensitive to imperfect friction and transmission models. Residual errors often peak around velocity reversals and during highly dynamic motion phases [7, 22–25, 27]. GP and deep residual models can reduce these errors but commonly absorb friction, backlash, and contact effects into a single disturbance term [19, 20, 28–30, 38], which complicates separating payload-induced dynamics from interaction-induced effects in a physics-consistent way.
- **Limited integration of temporal residual learning with physics-structured models in the actuation domain.** Deep sequence models are frequently used either as black-box regressors from motion histories to torques/forces or as residual predictors on top of rigid-body baselines [19, 20, 32, 33]. Physics-informed DeLaN/PINN models improve inverse-dynamics prediction and can leverage motor-side measurements when joint-torque sensing is unavailable [16–18, 38], but are typically trained and evaluated as fixed predictors. This motivates a pragmatic two-stage architecture that combines a physics-consistent inverse-dynamics backbone with a learned history-dependent residual, trained from encoder and motor-current data, to serve as a nominal model for downstream tool/payload compensation and PDPI.



Figure 4: The Concept graph summarising the reviewed literature. Green nodes represent method categories Q1-Q4 with node size proportional to the number of papers in each category. Blue nodes denote the main estimation goals scaled by how many papers address each goal. Red nodes correspond to individual references R1-R34. Their diameter is proportional to the citation count (clipped at the maximum). Directed edges indicate that a given reference belongs to a method category and contributes to one or more estimation goals. R1-R34 table in Appendix C

2.4 Related Work

This thesis is most closely related to physics-informed inverse-dynamics models that combine Lagrangian structure with learning-based compensation in settings without direct joint-torque supervision. Wu et al. [18] propose an extended DeLaN formulation that incorporates motor-side effects and couplings and learns dynamics directly from electrical motor signals. This motivates current-domain supervision when only motor currents are available. Hu et al. [38] develop a friction-inclusive, PINN-style dynamics model that combines a structured backbone with an explicit friction model and a sequential residual learner (TCN), which is conceptually closest to the two-stage “structured baseline + temporal residual” design adopted in this thesis. Finally, the DeLaN framework and benchmark presented by Lutter et al. [16] provides the methodological baseline for physics-inspired dynamics learning and serves as the code-level starting point for the DeLaN implementation used in this work.

2.5 Deep Lagrangian Networks

Deep Lagrangian Networks (DeLaN) are physics-inspired neural models for rigid-body dynamics that embed Lagrangian mechanics directly into the network structure. Instead of learning a black-box mapping from state to actuation, DeLaN parameterises the system energy and derives the dynamics via the Euler–Lagrange equations, which yields models that are physically plausible by construction and provide access to multiple consistent quantities (energy, forward dynamics, inverse dynamics) from the same set of parameters [16]. Concretely, DeLaN represents the Lagrangian $\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}) = T(\mathbf{q}, \dot{\mathbf{q}}) - V(\mathbf{q})$ by learning a configuration-dependent inertia matrix and a potential-energy term, and then evaluates the implied inverse dynamics $\hat{\tau}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ through analytic differentiation. To respect key structural properties, common DeLaN implementations enforce symmetry and positive definiteness of the learned inertia (e.g., via a Cholesky-factor parameterisation), while non-conservative effects are modelled through an additional friction term or residual component [16, 38]. DeLaN models are well suited to dynamics-learning problems that are strongly constrained by mechanics.

2.6 Long-Short-Term-Memory

Long short-term memory (LSTM) networks are recurrent neural networks designed to learn temporal dependencies in sequential data. An LSTM processes a time series step by step and maintains an internal cell state as a dedicated memory, whose content is controlled by multiplicative gates (forget, input, output). This gating mechanism enables the model to selectively retain or overwrite information over long horizons, which is crucial when dynamics depend on history rather than solely on the instantaneous state.

In robotics, LSTMs are therefore frequently used as sequence-to-vector or sequence-to-sequence regressors to infer forces and torques from histories of proprioceptive signals. Kružić et al. [20] demonstrate this idea for a 7-DoF manipulator by training deep networks (including an LSTM) to estimate end-effector forces and joint torques from measured motion signals, without requiring an explicit analytical dynamics model. Tao et al. [19] apply an LSTM as a residual predictor in a hybrid inverse-dynamics compensation scheme, where the recurrent model learns the unmodelled error terms (e.g., due to friction and other uncertainties) on top of a rigid-body baseline and thereby reduces the remaining torque residuals.

3 Methods

This chapter formalises the proposed two-stage inverse-dynamics pipeline and its learning objectives. It first introduces the physics-inspired DeLaN backbone and the residual LSTM correction model, and then derives the combined predictor used throughout the thesis. Finally, the chapter specifies the dataset interface in terms of input/output signal vectors and dimensions, and clarifies the offline training versus online execution setting adopted in the experimental pipeline. To motivate the perspective of this work towards gripper/tool manipulation and compensation, as well as payload dynamic parameter identification, the structured inverse-dynamics formulation is also related to the end-effector wrench measurement frame. However, this thesis evaluates the developed method purely in joint space via motor-current modelling.

3.1 Stage 1: Structured inverse dynamics for robot + fixed gripper

In a first step, we learn a nominal inverse-dynamics model for the robot-gripper system without payload and without contact. From the synchronised dataset we obtain

$$\{\mathbf{q}_k, \dot{\mathbf{q}}_k, \ddot{\mathbf{q}}_k, \mathbf{I}_k, \vec{F}_{\text{meas},k}\}_{k=1}^N,$$

where $\mathbf{q}_k, \dot{\mathbf{q}}_k, \ddot{\mathbf{q}}_k \in \mathbb{R}^n$ are joint position, velocity and acceleration, $\mathbf{I}_k \in \mathbb{R}^n$ are motor currents, and $\vec{F}_{\text{meas},k} \in \mathbb{R}^6$ is the measured flange wrench in the sensor frame S . For brushless DC motors with torque constant k_t [40], the measured motor torques, for the main area of operation range are

$$\tau_{\text{motor},k} = k_t \mathbf{I}_k. \quad (9)$$

The nominal robot-gripper dynamics are parameterised by a Deep Lagrangian Network (DeLaN) [17] with parameters θ for the conservative dynamics and ψ for friction and other non-conservative terms. The network implements a Lagrangian

$$\mathcal{L}_\theta(\mathbf{q}, \dot{\mathbf{q}}) = \frac{1}{2} \dot{\mathbf{q}}^\top \mathbf{M}_\theta(\mathbf{q}) \dot{\mathbf{q}} - V_\theta(\mathbf{q}), \quad (10)$$

with positive-definite inertia matrix $\mathbf{M}_\theta(\mathbf{q})$ (e.g. represented via a Cholesky-factor network) and potential $V_\theta(\mathbf{q})$ represented by a neural network.[16–18, 38] Using the Euler-Lagrange equations yields the conservative joint torques

$$\tau_{\text{cons}}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}; \theta) = \mathbf{M}_\theta(\mathbf{q}) \ddot{\mathbf{q}} + \mathbf{C}_\theta(\mathbf{q}, \dot{\mathbf{q}}) \dot{\mathbf{q}} + \mathbf{G}_\theta(\mathbf{q}), \quad (11)$$

where \mathbf{C}_θ and \mathbf{G}_θ are implicitly defined by \mathcal{L}_θ [17, 38].

Joint friction and other non-conservative effects, for example joint backlash or friction and heat losses, are captured by Long-Short-Term-Memory Network further described in Subsection 3.2, Section 3.6

The DeLaN torque prediction is the are the conservative parts of the inverse dynamic system, so that for sample k

$$\hat{\tau}_{\text{DeLaN}}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}; \theta) = \boldsymbol{\tau}_{\text{cons}}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}; \theta). \quad (12)$$

The parameter θ is trained offline by minimising a joint-space regression loss [16]

$$\mathcal{L}_{\text{DeLaN}}(\theta, \psi) = \frac{1}{N} \sum_{k=1}^N \|\hat{\tau}_{\text{DeLaN}}(\mathbf{q}_k, \dot{\mathbf{q}}_k, \ddot{\mathbf{q}}_k; \theta) - \boldsymbol{\tau}_{\text{motor},k}\|_2^2. \quad (13)$$

Note that this training objective uses only joint states and motor torques. The force/torque sensor is not required for fitting the DeLaN model.

After training, the DeLaN parameters are frozen and the model serves as a data-driven nominal inverse-dynamics model of the robot-gripper system.

3.2 Stage 2: Sequence model for residual joint torques

In the second stage, we model history-dependent effects that are not captured by the structured DeLaN model, such as backlash and nonlinear friction. Using the same robot-gripper dataset (still without payload and without contact), we first compute the joint-space residual torques

$$\mathbf{r}_{\tau,k} = \boldsymbol{\tau}_{\text{motor},k} - \hat{\tau}_{\text{DeLaN},k}. \quad (14)$$

Let H denote the sequence length (number of time steps in the history window). For each time index $k \geq H$ we construct an input sequence

$$\mathbf{x}_k = \left[\mathbf{q}_{k-H+1:k}, \dot{\mathbf{q}}_{k-H+1:k}, \ddot{\mathbf{q}}_{k-H+1:k}, \hat{\tau}_{\text{DeLaN},k-H+1:k} \right], \quad (15)$$

where $\mathbf{q}_{a:b}$ denotes the stacked joint vectors $(\mathbf{q}_a, \dots, \mathbf{q}_b)$, and analogously for $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$ and $\hat{\tau}_{\text{DeLaN}}$. This construction stacks the last H joint states together with the corresponding DeLaN torque predictions into a single sequence feature vector \mathbf{x}_k .

An LSTM with parameters φ maps this sequence to a residual-torque prediction

$$\hat{\mathbf{r}}_{\tau,k} = f_{\text{LSTM}}(\mathbf{x}_k; \varphi) \in \mathbb{R}^n. \quad (16)$$

The LSTM is trained to minimise the mean-squared error between predicted and true residual torques,

$$\mathcal{L}_{\text{LSTM}}(\varphi) = \frac{1}{N_H} \sum_{k=H}^N \|\hat{\mathbf{r}}_{\tau,k} - \mathbf{r}_{\tau,k}\|_2^2, \quad (17)$$

where $N_H = N - H + 1$ is the number of valid sequences.

The combined joint-space model for the robot-gripper system is then

$$\hat{\boldsymbol{\tau}}_{\text{RG},k} = \hat{\boldsymbol{\tau}}_{\text{DeLaN},k} + \hat{\boldsymbol{r}}_{\tau,k}. \quad (18)$$

For evaluation in the sensor frame, the corresponding combined flange wrench is obtained via the Jacobian mapping

$$\hat{\vec{F}}_{\text{RG},k} = {}^S J(\mathbf{q}_k)^{-\top} \hat{\boldsymbol{\tau}}_{\text{RG},k}. \quad (19)$$

Since both stages are trained exclusively on data without payload and without environment contact, $\hat{\boldsymbol{\tau}}_{\text{RG},k}$ and $\hat{\vec{F}}_{\text{RG},k}$ represent a high-fidelity, history-aware model of the nominal robot-gripper dynamics. In later stages, deviations between this model and the measured joint torques or flange wrenches can be attributed to the effective rigid-body contribution of additional payloads and contacts.

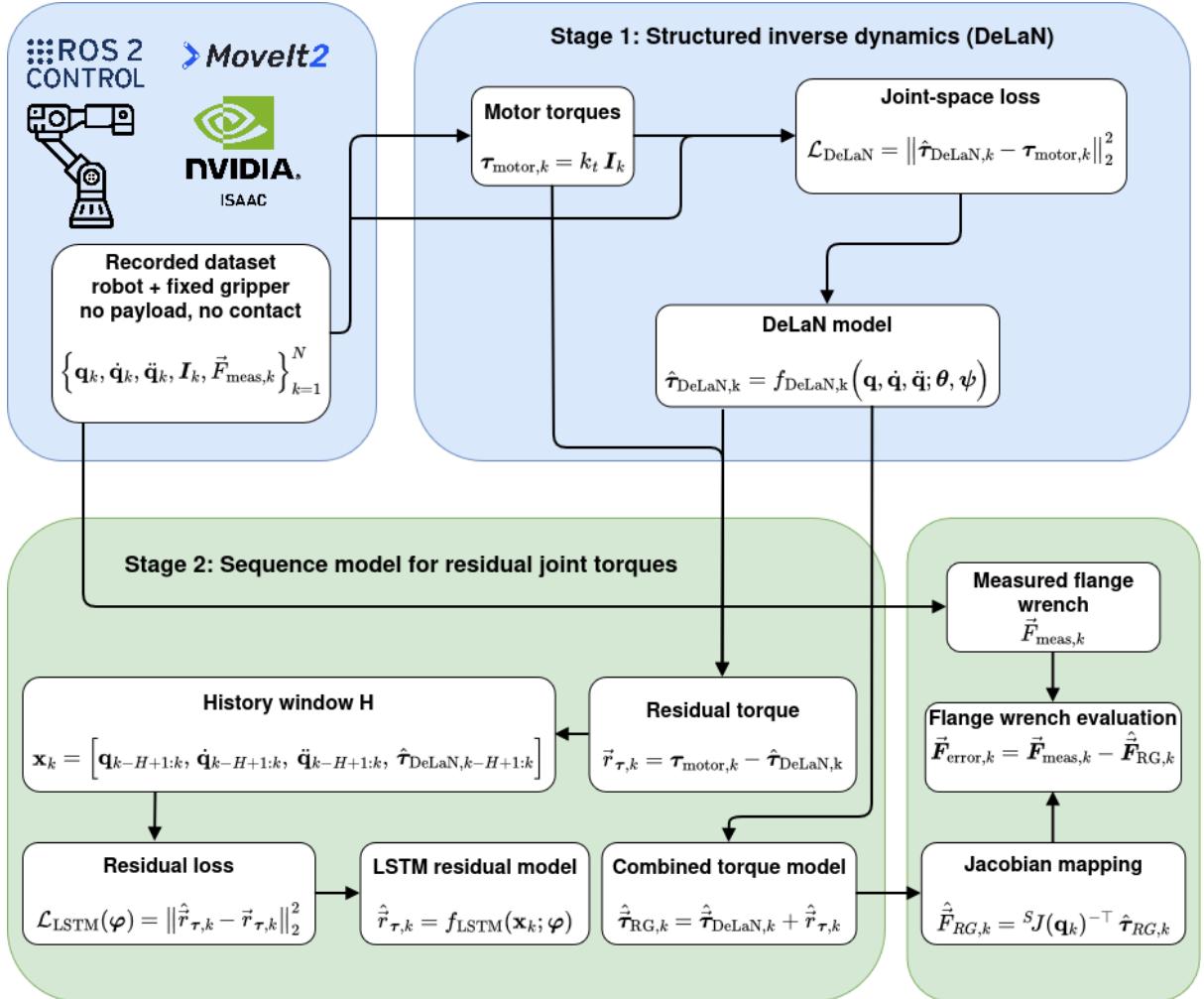


Figure 5: Two-stage learning pipeline for the nominal robot-gripper dynamics. Stage 1 learns a structured inverse-dynamics model (DeLaN) in joint space from encoder and motor-current data and is trained by regressing motor torques. In Stage 2, a recurrent sequence model (LSTM) takes joint-state histories and DeLaN torque predictions as input and learns residual joint torques over a fixed history window. The combined joint-space model is then mapped through the Jacobian to obtain the nominal flange wrench in the force/torque sensor frame, which is compared against the measured wrench for evaluation.

3.3 Dataset of Collaborative Robots

All experiments are based on the publicly available “Dataset of Collaborative Robots for Energy Consumption Modeling” released via IEEE DataPort [41] and documented in [21]. The dataset contains measurements from two Universal Robots platforms (UR3e and UR10e) recorded both without load and with an external payload.

Each log sample provides a time stamp t and a trajectory identifier, and includes joint-space signals (joint positions \mathbf{q} , joint velocities $\dot{\mathbf{q}}$) together with electrical measurements (per-joint

motor currents \mathbf{i} , motor voltages, as well as robot-level current and voltage). In addition, the dataset provides end-effector quantities such as Cartesian position and the measured wrench (force and moment) at the end effector. In the remainder of this thesis, motor current is treated as the central measured actuation signal.

To excite the robot dynamics across a wide range of operating conditions, the robots execute sinusoidal joint motions with varying amplitudes, frequencies, and initial conditions [21]:

$$q_i(t) = q_{i0} + A_i \cos(2\pi f_i t + \varphi_i), \quad (20)$$

where q_i , q_{i0} , A_i , f_i and φ_i denote the desired joint position, initial position, amplitude, oscillation frequency, and phase of joint i , respectively. The experiments were conducted for UR3e and UR10e under two load conditions: without load and with an attached payload (hammer 1.5 kg and RobotIQ 2F-85 gripper 1 kg) [21].

The signals are recorded at 100 Hz. For each robot (and load condition), the dataset provides 50,000 samples for training and 5,000 samples for testing [21]. Since the underlying dynamics are time-invariant, the published dataset is formed by combining multiple shorter recordings into one consistent dataset, without treating discontinuities as separate experiments [21].

Since the dataset provides motor current measurements, we formulate training objectives and evaluation metrics in the measured actuation domain as per-joint motor currents $\mathbf{i}_{\text{motor}}$ in A. When needed for interpretation, current-domain errors can be mapped to equivalent motor-domain errors via the (previously introduced) constant conversion factor k_t , by multiplying with k_t . Consistent with the pipeline overview in Fig. 5, we omit the conversion $\tau_{\text{motor}} = k_t \mathbf{i}_{\text{motor}}$ and the rightmost Jacobian-based flange-wrench mapping block, and evaluate the approach directly as a combined motor-current model $\hat{\mathbf{i}}_{\text{comb}}$.

3.4 Data preprocessing and dataset construction

The preprocessing step converts the raw IEEE DataPort logs into a DeLaN-ready, trajectory-wise dataset. In our implementation, we use a wide CSV representation in which each row corresponds to one time step and contains all six joint channels, together with an explicit trajectory identifier. Concretely, the wide format provides a time stamp t_1 , per-joint kinematics q_1-q_6 and dq_1-dq_6 , per-joint motor currents $I_{q1}-I_{q6}$, and an integer ID column. The DeLaN preprocessing service maps this wide table into the long-format interface expected by the upstream codebase[16] (columns Time, Joint Name, Position, Velocity, Acceleration, Effort), propagating ID as trajectory_id. Thus, trajectory boundaries are taken directly from the dataset rather than being inferred by heuristic re-segmentation.

We consider the six revolute joints of the manipulator and use a fixed ordering $j \in \{1, \dots, 6\}$ consistent with the wide columns. For each row (frame) index f we read

$$t_f, \quad q_{f,j}, \quad \dot{q}_{f,j}, \quad I_{f,j},$$

where $I_{f,j}$ denotes the measured motor current. Stacking over joints yields the frame-wise vectors

$$\mathbf{q}_f, \quad \dot{\mathbf{q}}_f, \quad \mathbf{I}_f \in \mathbb{R}^6.$$

For compatibility with the DeLaN data interface, the current vector is stored as a generic actuation channel $\tau_f^{\text{eff}} := \mathbf{I}_f$.

Let $\mathcal{I} = \{1, \dots, N_{\text{traj}}\}$ denote the set of unique trajectory identifiers. For each $i \in \mathcal{I}$, we collect all frames with `trajectory_id` equal to i , sort them by time, and stack them into a variable-length trajectory of length T_i . To attenuate sensor noise while avoiding temporal misalignment, we apply a 4th-order Butterworth low-pass filter with cutoff frequency $f_c = 10$ Hz. The filter is applied in zero-phase form (forward–backward filtering), thereby preventing phase shifts in \mathbf{q} , $\dot{\mathbf{q}}$ and \mathbf{I} . Joint accelerations $\ddot{\mathbf{q}}$ are then derived from the filtered velocities via numerical differentiation and are filtered analogously, which aligns with the baseline preprocessing used for the dataset [21].

For trajectory i this yields a time series of length T_i with

$$\mathbf{t}^{(i)} = (t_1^{(i)}, \dots, t_{T_i}^{(i)}) \in \mathbb{R}^{T_i}, \quad (21)$$

$$\mathbf{Q}^{(i)} = \begin{bmatrix} \mathbf{q}_1^{(i)\top} \\ \vdots \\ \mathbf{q}_{T_i}^{(i)\top} \end{bmatrix} \in \mathbb{R}^{T_i \times 6}, \quad (22)$$

$$\dot{\mathbf{Q}}^{(i)} = \begin{bmatrix} \dot{\mathbf{q}}_1^{(i)\top} \\ \vdots \\ \dot{\mathbf{q}}_{T_i}^{(i)\top} \end{bmatrix} \in \mathbb{R}^{T_i \times 6}, \quad (23)$$

$$\ddot{\mathbf{Q}}^{(i)} = \begin{bmatrix} \ddot{\mathbf{q}}_1^{(i)\top} \\ \vdots \\ \ddot{\mathbf{q}}_{T_i}^{(i)\top} \end{bmatrix} \in \mathbb{R}^{T_i \times 6}, \quad (24)$$

$$\mathbf{T}^{(i)} = \begin{bmatrix} \boldsymbol{\tau}_1^{\text{eff},(i)\top} \\ \vdots \\ \boldsymbol{\tau}_{T_i}^{\text{eff},(i)\top} \end{bmatrix} \in \mathbb{R}^{T_i \times 6}. \quad (25)$$

In short, each trajectory i is represented by

$$(\mathbf{t}^{(i)}, \mathbf{Q}^{(i)}, \dot{\mathbf{Q}}^{(i)}, \ddot{\mathbf{Q}}^{(i)}, \mathbf{T}^{(i)}),$$

with shapes

$$\mathbf{t}^{(i)} \in \mathbb{R}^{T_i}, \quad \mathbf{Q}^{(i)}, \dot{\mathbf{Q}}^{(i)}, \ddot{\mathbf{Q}}^{(i)}, \mathbf{T}^{(i)} \in \mathbb{R}^{T_i \times 6}.$$

We randomly permute \mathcal{I} (with a dataset seed) and partition it into disjoint training and test index sets,

$$\mathcal{I}_{\text{train}} \cup \mathcal{I}_{\text{test}} = \mathcal{I}, \quad \mathcal{I}_{\text{train}} \cap \mathcal{I}_{\text{test}} = \emptyset,$$

such that a given trajectory is entirely in either the training or the test set. This avoids leakage of near-identical neighbouring samples across splits. For the learning-curve experiments, we optionally subsample a fixed number of trajectories K (seeded) before splitting, such that each (K, seed) corresponds to a fixed trajectory subset.

For efficient loading during model training, the trajectory-wise data are stored in a NumPy .npz archive with the following keys:

$$\text{train_labels}, \text{train_t}, \text{train_q}, \text{train_qd}, \text{train_qdd}, \text{train_tau}, \quad (26)$$

$$\text{test_labels}, \text{test_t}, \text{test_q}, \text{test_qd}, \text{test_qdd}, \text{test_tau}. \quad (27)$$

Each of the trajectory-wise arrays is stored as a one-dimensional object array. For example,

$$\text{train_q}[i] \cong \mathbf{Q}^{(i)} \in \mathbb{R}^{T_i \times 6},$$

and analogously for `train_qd`, `train_qdd` and `train_tau`. Thus, if there are N_{train} training trajectories, we have

$$\text{train_q} \in \mathbb{R}_{\text{object}}^{N_{\text{train}}}, \quad \text{train_q}[i] \in \mathbb{R}^{T_i \times 6},$$

and likewise for the test set.

In the current implementation the `Effort` column is used to populate $\mathbf{T}^{(i)}$; the conversion to motor torques via $\boldsymbol{\tau}_{\text{motor},k} = k_t \mathbf{I}_k$ is applied later in the training pipeline (cf. (6)).

On the DeLaN side, the trajectory-wise arrays are flattened into a single pool of samples for stochastic optimisation. Denoting concatenation along the time dimension by `vstack`(\cdot), the training set becomes

$$\tilde{\mathbf{Q}}_{\text{train}} = \text{vstack}(\text{train_q}[i]) \in \mathbb{R}^{N_{\text{train,tot}} \times 6}, \quad (28)$$

$$\dot{\tilde{\mathbf{Q}}}_{\text{train}} = \text{vstack}(\text{train_qd}[i]) \in \mathbb{R}^{N_{\text{train,tot}} \times 6}, \quad (29)$$

$$\ddot{\tilde{\mathbf{Q}}}_{\text{train}} = \text{vstack}(\text{train_qdd}[i]) \in \mathbb{R}^{N_{\text{train,tot}} \times 6}, \quad (30)$$

$$\tilde{\mathbf{T}}_{\text{train}} = \text{vstack}(\text{train_tau}[i]) \in \mathbb{R}^{N_{\text{train,tot}} \times 6}, \quad (31)$$

where

$$N_{\text{train,tot}} = \sum_{i \in \mathcal{I}_{\text{train}}} T_i$$

is the total number of training time steps across all trajectories. An analogous flattening is performed for the test set.

All arrays are cast to a floating-point dtype (e.g. `float32`) to avoid `dtype=object` issues in the JAX-based DeLaN implementation.

3.5 DeLaN training setup

Given the flattened dataset

$$\mathcal{D}_{\text{train}} = \{(\mathbf{q}_k, \dot{\mathbf{q}}_k, \ddot{\mathbf{q}}_k, \boldsymbol{\tau}_{\text{motor},k})\}_{k=1}^{N_{\text{train,tot}}},$$

the DeLaN model f_{DeLaN} is trained as described in Section 3. Here we summarise the main implementation details used in this work.

Instead of feeding the raw joint angles \mathbf{q}_k directly into the DeLaN subnetworks, we use a bounded, trigonometric feature map

$$\phi(\mathbf{q}_k) = \begin{bmatrix} \mathbf{q}_k \\ \sin(\mathbf{q}_k) \\ \cos(\mathbf{q}_k) \end{bmatrix} \in \mathbb{R}^{3n_{\text{dof}}}, \quad (32)$$

where the sine and cosine are applied element-wise. This follows the common practice in DeLaN/PINN-based models of encoding revolute joints through periodic features, which mitigates angle wrap-around and keeps the network inputs well scaled.

To prevent joints with large torque magnitudes from dominating the optimisation, we employ a per-joint normalisation in the inverse-dynamics loss. Let

$$\hat{\boldsymbol{\tau}}_{\text{DeLaN},k} = f_{\text{DeLaN}}(\mathbf{q}_k, \dot{\mathbf{q}}_k, \ddot{\mathbf{q}}_k; \boldsymbol{\theta}, \boldsymbol{\psi}) \quad (33)$$

denote the DeLaN torque prediction for sample k .

We compute empirical variances of the training torques

$$\boldsymbol{\sigma}_{\boldsymbol{\tau}}^2 = \text{Var}(\tilde{\mathbf{T}}_{\text{train}}) \in \mathbb{R}^6, \quad (34)$$

and define a diagonal weighting matrix

$$\mathbf{W}_{\boldsymbol{\tau}} = \text{diag}(\boldsymbol{\sigma}_{\boldsymbol{\tau}}^{-1}) \in \mathbb{R}^{6 \times 6}, \quad (35)$$

where the inverse is taken element-wise. The training objective can then be written as

$$\mathcal{L}_{\text{DeLaN}}(\boldsymbol{\theta}, \boldsymbol{\psi}) = \frac{1}{N_{\text{train,tot}}} \sum_{k=1}^{N_{\text{train,tot}}} \|\mathbf{W}_{\boldsymbol{\tau}}(\hat{\boldsymbol{\tau}}_{\text{DeLaN},k} - \boldsymbol{\tau}_{\text{motor},k})\|_2^2, \quad (36)$$

which corresponds to a per-joint normalisation of the squared torque error. In practice, additional factors derived from $\ddot{\mathbf{q}}$ may be included analogously, but the essential idea is that the loss is balanced across joints.

All training samples are stored in a replay buffer \mathcal{M} , which supports random-access mini-batch sampling. At each optimisation step, a mini-batch index set $S \subset \{1, \dots, N_{\text{train,tot}}\}$ with $|S| = B$ is drawn (e.g. by taking a random permutation of indices and slicing), and the corresponding batch

$$\mathcal{B} = \{(\mathbf{q}_k, \dot{\mathbf{q}}_k, \ddot{\mathbf{q}}_k, \boldsymbol{\tau}_{\text{motor},k})\}_{k \in S}$$

is used to evaluate $\mathcal{L}_{\text{DeLaN}}$ and its gradients. This renders the optimisation effectively i.i.d. over the flattened pool of samples. The original trajectory grouping is retained only for the train/test split and for later sequence-based models and visualisations.

Throughout the preprocessing and training pipeline, we assume a (approximately) constant sampling interval

$$\Delta t_k = t_{k+1} - t_k \approx \Delta t, \quad \forall k,$$

as is standard for robot control logs. A fixed sampling period is crucial for:

- interpreting measured joint velocities $\dot{\mathbf{q}}_k$ and accelerations $\ddot{\mathbf{q}}_k$ as consistent derivatives of \mathbf{q}_k ,
- applying finite-difference schemes or filtering to reconstruct $\dot{\mathbf{q}}_k$ and $\ddot{\mathbf{q}}_k$ from position data,
- ensuring that the DeLaN model learns a time-homogeneous mapping from $(\mathbf{q}_k, \dot{\mathbf{q}}_k, \ddot{\mathbf{q}}_k)$ to $\tau_{\text{motor},k}$.

In the present work, the raw controller velocities and accelerations are used directly, but enforcing a constant sampling time and applying dedicated filtering to $\dot{\mathbf{q}}_k$ and $\ddot{\mathbf{q}}_k$ constitutes a straightforward extension of the preprocessing pipeline.

3.6 Long-Short-Term-Memory Training Setup

After training Stage 1, the DeLaN parameters (θ, ψ) are frozen and the model is evaluated on all training samples $(\mathbf{q}_k, \dot{\mathbf{q}}_k, \ddot{\mathbf{q}}_k)$, $k = 1, \dots, N_{\text{train,tot}}$. This yields joint-torque predictions

$$\hat{\boldsymbol{\tau}}_{\text{DeLaN},k} = f_{\text{DeLaN}}(\mathbf{q}_k, \dot{\mathbf{q}}_k, \ddot{\mathbf{q}}_k; \theta, \psi), \quad (37)$$

which we stack into a matrix

$$\hat{\mathbf{T}}_{\text{DeLaN,train}} = \begin{bmatrix} \hat{\boldsymbol{\tau}}_{\text{DeLaN},1}^\top \\ \vdots \\ \hat{\boldsymbol{\tau}}_{\text{DeLaN},N_{\text{train,tot}}}^\top \end{bmatrix} \in \mathbb{R}^{N_{\text{train,tot}} \times 6}. \quad (38)$$

Using the corresponding motor torques $\boldsymbol{\tau}_{\text{motor},k}$, we define the joint-space residual torques for all training samples as

$$\boldsymbol{r}_{\tau,k} = \boldsymbol{\tau}_{\text{motor},k} - \hat{\boldsymbol{\tau}}_{\text{DeLaN},k}, \quad k = 1, \dots, N_{\text{train,tot}}, \quad (39)$$

and collect them in

$$\mathbf{R}_{\tau,\text{train}} = \begin{bmatrix} \boldsymbol{r}_{\tau,1}^\top \\ \vdots \\ \boldsymbol{r}_{\tau,N_{\text{train,tot}}}^\top \end{bmatrix} \in \mathbb{R}^{N_{\text{train,tot}} \times 6}. \quad (40)$$

Let H denote the history length (sequence length) used in Stage 2. From the flattened arrays $\tilde{\mathbf{Q}}_{\text{train}}$, $\dot{\tilde{\mathbf{Q}}}_{\text{train}}$, $\ddot{\tilde{\mathbf{Q}}}_{\text{train}}$, $\hat{\mathbf{T}}_{\text{DeLaN,train}}$ we then construct overlapping sequences with a sliding window. For each time index $k \geq H$ we define the LSTM input sequence

$$\mathbf{x}_k = [\mathbf{q}_{k-H+1:k}, \dot{\mathbf{q}}_{k-H+1:k}, \ddot{\mathbf{q}}_{k-H+1:k}, \hat{\boldsymbol{\tau}}_{\text{DeLaN},k-H+1:k}], \quad (41)$$

where, as in the main Methods section, $\mathbf{q}_{a:b}$ denotes the stacked joint vectors $(\mathbf{q}_a, \dots, \mathbf{q}_b)$ and analogously for $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$ and $\hat{\boldsymbol{\tau}}_{\text{DeLaN}}$. The corresponding target for each sequence is chosen as the residual torque at the final time step,

$$\mathbf{y}_k = \boldsymbol{r}_{\tau,k} \in \mathbb{R}^6. \quad (42)$$

Stacking all valid windows yields the Stage 2 training dataset

$$\mathbf{X}_{\text{train}}^{\text{LSTM}} = \{\mathbf{x}_k\}_{k=H}^{N_{\text{train,tot}}}, \quad (43)$$

$$\mathbf{Y}_{\text{train}}^{\text{LSTM}} = \{\mathbf{y}_k\}_{k=H}^{N_{\text{train,tot}}}, \quad (44)$$

with $N_H = N_{\text{train,tot}} - H + 1$ sequences in total. In implementation, $\mathbf{X}_{\text{train}}^{\text{LSTM}}$ is stored as a tensor of shape (N_H, H, d_{in}) , where $d_{\text{in}} = 4n_{\text{dof}}$ corresponds to $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \hat{\tau}_{\text{DeLaN}})$ per time step, and $\mathbf{Y}_{\text{train}}^{\text{LSTM}} \in \mathbb{R}^{N_H \times 6}$. A completely analogous construction is used for the test split.

We train an LSTM-based [42] residual model g_ϕ on the windowed dataset $(\mathbf{x}_k, \mathbf{y}_k)$. As in Stage 1, the goal is to prevent single joints from dominating the objective. To this end, we apply a train-only standardisation of both inputs and targets, and train the network to predict scaled residuals.

Let $\mathbf{X}_{\text{train}}^{\text{LSTM}} \in \mathbb{R}^{N_H \times H \times d_{\text{in}}}$ and $\mathbf{Y}_{\text{train}}^{\text{LSTM}} \in \mathbb{R}^{N_H \times 6}$ denote the windowed training set. We compute feature-wise input statistics across all windows and time steps,

$$\mu_{x,j} = \frac{1}{N_H H} \sum_{k=1}^{N_H} \sum_{t=1}^H X_{\text{train}}^{\text{LSTM}}[k, t, j], \quad \sigma_{x,j}^2 = \frac{1}{N_H H} \sum_{k=1}^{N_H} \sum_{t=1}^H (X_{\text{train}}^{\text{LSTM}}[k, t, j] - \mu_{x,j})^2, \quad (45)$$

for $j = 1, \dots, d_{\text{in}}$, and collect them into vectors $\boldsymbol{\mu}_x, \boldsymbol{\sigma}_x \in \mathbb{R}^{d_{\text{in}}}$. Analogously, we compute per-joint target statistics over windows,

$$\mu_{y,i} = \frac{1}{N_H} \sum_{k=1}^{N_H} Y_{\text{train}}^{\text{LSTM}}[k, i], \quad \sigma_{y,i}^2 = \frac{1}{N_H} \sum_{k=1}^{N_H} (Y_{\text{train}}^{\text{LSTM}}[k, i] - \mu_{y,i})^2, \quad (46)$$

for $i = 1, \dots, 6$, yielding $\boldsymbol{\mu}_y, \boldsymbol{\sigma}_y \in \mathbb{R}^6$. To avoid numerical issues, standard deviations below a small threshold ε are clamped to 1 component-wise. Defining diagonal scaling matrices

$$\mathbf{W}_x = \text{diag}(\boldsymbol{\sigma}_x^{-1}), \quad \mathbf{W}_y = \text{diag}(\boldsymbol{\sigma}_y^{-1}),$$

the standardised inputs and targets are then given by

$$\mathbf{x}_{k,t}^{\text{n}} = \mathbf{W}_x (\mathbf{x}_{k,t} - \boldsymbol{\mu}_x), \quad (47)$$

$$\mathbf{r}_{\tau,k}^{\text{s}} = \mathbf{W}_y (\mathbf{r}_{\tau,k} - \boldsymbol{\mu}_y), \quad (48)$$

with $\mathbf{r}_{\tau,k} = \mathbf{y}_k$. The same $(\boldsymbol{\mu}_x, \boldsymbol{\sigma}_x, \boldsymbol{\mu}_y, \boldsymbol{\sigma}_y)$ computed on the training split are used to scale the test split.

The residual model is implemented as a two-layer LSTM with dropout regularisation and a linear output layer,

$$\hat{\mathbf{r}}_{\tau,k}^{\text{s}} = g_\phi(\mathbf{x}_k^{\text{n}}) \in \mathbb{R}^6, \quad (49)$$

where \mathbf{x}_k^{n} denotes the standardised input window and $\hat{\mathbf{r}}_{\tau,k}^{\text{s}}$ the predicted scaled residual. Training minimises a mean-squared error on scaled residuals,

$$\mathcal{L}_{\text{LSTM}}(\phi) = \frac{1}{N_H} \sum_{k=1}^{N_H} \|\hat{\mathbf{r}}_{\tau,k}^{\text{s}} - \mathbf{r}_{\tau,k}^{\text{s}}\|_2^2, \quad (50)$$

which corresponds to a balanced, per-joint normalisation via σ_y . Optimisation is performed with Adam using shuffled mini-batches and an internal validation split. The best-performing model is selected by early stopping on the validation loss.

For reporting and for composing the final DeLaN+LSTM torque prediction, the scaled residual is mapped back to physical units via

$$\hat{\mathbf{r}}_{\tau,k} = \mathbf{W}_y^{-1} \hat{\mathbf{r}}_{\tau,k}^s + \boldsymbol{\mu}_y, \quad (51)$$

where $\mathbf{W}_y^{-1} = \text{diag}(\sigma_y)$.

4 Experimental Setup

This chapter specifies the experimental protocol used to evaluate the proposed two-stage DeLaN+LSTM pipeline on the IEEE DataPort collaborative-robot trajectories in the motor-current domain. All experiments operate on trajectory logs of joint positions \mathbf{q} , joint velocities $\dot{\mathbf{q}}$, and per-joint motor currents $\mathbf{i}_{\text{motor}}$ sampled at approximately 100 Hz; joint accelerations $\ddot{\mathbf{q}}$ are derived from the recorded state after per-trajectory preprocessing. To prevent leakage between trajectories, all train/validation/test splits are constructed at the trajectory level, and performance is reported as per-joint motor-current RMSE in A, aggregated across seeds where applicable. Two complementary studies are considered: a learning-curve sweep (K-domination) that varies the number of available trajectories K to quantify optimisation stability and generalisation in both stages, and a best-model study at fixed K that selects robust DeLaN and LSTM configurations based on validation performance before evaluating on the held-out test split.

4.1 DeLaN + LSTM - Learning Curve Stroy

This learning-curve study evaluates the two-stage pipeline as a function of the number of available trajectories K . As K increases, both the DeLaN backbone and the residual LSTM train more consistently across dataset seeds, yielding lower and more uniform per-joint motor-current errors; the residual LSTM provides an additional reduction of the remaining modelling error once the DeLaN baseline has stabilised.

4.1.1 K-Domination

The purpose of the K-domination study is to quantify how the number of available demonstration trajectories influences the complete two-stage pipeline (Stage 1 DeLaN and Stage 2 residual LSTM). To this end, we construct multiple training sets by drawing K trajectories from a fixed pool, train the full pipeline for each set under identical hyperparameters, and evaluate performance as a function of K .

The base dataset consists of 122 trajectories, each identified by a trajectory ID. For each trajectory, we use joint position \mathbf{q} , joint velocity $\dot{\mathbf{q}}$, and motor current \mathbf{i} . All logs are recorded at approximately 100 Hz. Since trajectory durations vary substantially (roughly 5–40 s), all filtering and preprocessing steps are applied per trajectory. The resulting DeLaN-ready dataset is constructed as described in Section 3.4.

From the trajectory pool, we draw subsets of size

$$K \in \{8, 16, 32, 48, 64, 86, 122\},$$

using three independent dataset seeds. For each (K, seed) , a fixed trajectory split is created with test fraction 0.2 and validation fraction 0.1 at the level of complete trajectories (trajectories are never split across subsets).

For each (K, seed) subset, we train five DeLaN initialisations (seeds $s \in \{0, \dots, 4\}$) and select the best DeLaN by validation error (validation motor-current RMSE, equivalently MSE). All DeLaN runs use the structured JAX implementation and a fixed hyperparameter preset `lutter_like_256` (Table 1 [16]): softplus activation, width 256, depth 2, mini-batch size 1024, learning rate 10^{-4} and weight decay 10^{-5} . Training is run for at most 200 epochs with early stopping (patience 10, monitored on validation MSE) to avoid overfitting and to ensure that changes in performance are attributable to K rather than excessive training time.

After selecting the best DeLaN, we freeze it and export residuals for each trajectory. Stage 2 uses a history length $H = 100$ and feature mode `full`. Each LSTM input time step concatenates $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \hat{\mathbf{i}}_{\text{DeLaN}})$. The residual LSTM is trained with two stacked LSTM layers (units 128), dropout 0.2, batch size 64, and a validation split of 0.1. Training runs for at most 120 epochs and employs early stopping on validation loss (patience 20, warm-up 10 epochs), again fixing hyperparameters across all K to isolate the effect of trajectory quantity.

Algorithm 1 K-domination experiment protocol for the DeLaN+LSTM pipeline

Require: Trajectory pool \mathcal{T} with $|\mathcal{T}| = 122$ at 100 Hz
Require: $K \in \{8, 16, 32, 48, 64, 86, 122\}$, dataset seeds $\mathcal{D} = \{0, 1, 2\}$
Require: DeLaN seeds $\mathcal{S}_{\text{DeLaN}} = \{0, \dots, 4\}$
Ensure: Aggregated learning curves and per-joint motor-current RMSE as a function of K
Low-pass filter each trajectory (4^{th} -order Butterworth, $f_c = 10$ Hz, zero-phase)
for all K **do**
 for all $d \in \mathcal{D}$ **do**
 Sample K trajectories from \mathcal{T} using seed d
 Split into train/val/test trajectories (fixed fractions, no within-trajectory splitting)
 for all $s \in \mathcal{S}_{\text{DeLaN}}$ **do**
 Train DeLaN with fixed hyperparameters \triangleright max 200 epochs; early stop (patience 10)
 Record train loss and validation motor-current MSE/RMSE
 Select best DeLaN by validation error and freeze its parameters
 Export residual motor currents per trajectory
 Build residual windows ($H = 100$) and train LSTM \triangleright seed = dataset seed d ; max 120 epochs; early stop (patience 20)
 Record train/validation loss and residual motor-current RMSE
 Aggregate across seeds: median \pm IQR learning curves and progress-aligned errors

For each K , we aggregate results across the three dataset seeds. Reported learning curves (train loss, validation loss) are shown as median curves with interquartile ranges (IQR, 25–75 percentile) across the 5 delan seeds per (K, seed) . For Stage 1, where five DeLaN initialisations are trained per (K, seed) , we first compute the median \pm IQR across DeLaN seeds for each dataset seed, and then aggregate these seed-wise median curves across dataset seeds. For time-dependent error visualisations, trajectories are aligned by normalised progress (mapping each trajectory time index to $[0, 1]$) and resampled to a fixed number of bins; median \pm IQR is then computed per progress bin.

4.1.2 Best Model Approach

The K-domination sweep shows that once a sufficiently large number of trajectories is available, the DeLaN training dynamics become comparable across K and the remaining variance is dominated by which trajectories end up in the train/validation/test split (dataset seed) and the network initialisation (DeLaN seed). This effect is visible in the collapse of the median learning curves for $K \geq 32$ (Figures 6 and 7), while the dispersion and occasional outliers remain seed-dependent, especially for the progress-aligned motor-current RMSE (Figure 8).

To obtain a reliable basis for the second experiment, we therefore fix $K = 84$ and select a DeLaN hyperparameter preset within the stable K regime. Since Algorithm 2 performs validation-based checkpoint selection, the hyperparameters should reduce seed sensitivity (tight IQR with few or no catastrophic runs), reach low validation error quickly and consistently (stable early-

stopping behaviour) and generalise such that the validation-based ranking correlates with test performance. Accordingly, we do not select the setting with the lowest median validation error alone, but the setting that achieves a favourable accuracy–robustness trade-off across dataset and initialisation seeds.

For each DeLaN preset h and dataset seed d , we aggregate validation RMSE across DeLaN seeds and compute a robustness-aware selection score

$$\text{score}(h, d) = \text{median}(\text{RMSE}_{\text{val}}) + \lambda \cdot \text{IQR}(\text{RMSE}_{\text{val}}) + P \cdot \rho(d, h), \quad (52)$$

We use $\lambda = 0.5$ and $P = 10$ (in current units [A]). Here, the median term captures typical validation accuracy, the IQR term penalises sensitivity to dataset and initialisation seeds, and $\rho(d, h)$ denotes the divergence rate (fraction of failed runs under h on dataset seed d , e.g., NaNs or missing metrics). We then aggregate $\text{score}(h, d)$ across dataset seeds and select the preset h^* with minimal median score. Lower values of $\text{score}(h, d)$ (and thus of its median across d) therefore indicate both high accuracy and high reliability.

In addition to the scalar score, we report two aggregated scatter plots for comparing presets h : median validation RMSE versus IQR of validation RMSE and median validation RMSE versus median test RMSE. In both cases, statistics are computed across DeLaN seeds for each dataset seed first, and then aggregated across dataset seeds, such that each point reflects the combined accuracy–robustness behaviour under the dominant seed variability identified by K-domination. The bottom-left region of the median–IQR plot corresponds to accurate and stable settings, while the validation–test plot directly assesses whether validation error is a reliable selector for test performance in the sense required by Algorithm 2.

Guided by the stable “Lutter-like” regime observed in the K-domination results for $K \geq 32$ (Figures 6–9), we evaluate the following five presets at fixed $K = 84$: The baseline preset is chosen to match the current best-practice DeLaN configuration reported in [16] (`lutter_like_256`), while the remaining presets vary a single axis (capacity, regularisation, or step size) to probe stability and generalisation.

1. **Baseline (Lutter-like):** SoftPlus, batch size 1024, learning rate 10^{-4} , weight decay 10^{-5} , width 256, depth 2 (`lutter_like_256`).
2. **Smaller capacity:** baseline with width 128, depth 2 (`lutter_like_128`); tests whether the 256-wide model is necessary or whether a smaller network improves stability.
3. **Deeper network:** baseline with width 256, depth 3 (`lutter_like_256_d3`); tests whether additional depth improves representation at the expense of seed stability.
4. **More regularisation:** baseline with weight decay 10^{-4} (`lutter_like_256_wd1e4`); tests whether stronger regularisation reduces seed variance and improves generalisation.
5. **Lower step size:** baseline with learning rate $5 \cdot 10^{-5}$ (`lutter_like_256_lr5e5`); tests whether more conservative optimisation improves numerical stability versus underfitting.

This best-model study thus fixes $K = 84$ and asks: within the stable regime, which hyperparameters provide the best accuracy–robustness trade-off across dataset splits and DeLaN initialisations?

Algorithm 2 Best-model selection for the DeLaN

Require: 4th-order Butterworth low-pass filter with cutoff frequency $f_c = 10 \text{ Hz}$

Require: UR3_Load0Dataset training pool with 50,000 samples, split into (70%, 10%, 20%) train/validation/test subsets for model selection

Require: Dataset seeds $\mathcal{D} = \{0, 1, 2, 3, 4\}$, DeLaN seeds $\mathcal{S}_{\text{DeLaN}} = \{0, 1, 2, 3, 4\}$, LSTM seeds $\mathcal{S}_{\text{LSTM}} = \{0, 1, 2, 3, 4\}$

Require: Hyperparameter presets $\mathcal{H}_{\text{DeLaN}}$ and $\mathcal{H}_{\text{LSTM}}$ with $|\mathcal{H}_\cdot| = 5$

Ensure: Best DeLaN checkpoint and best LSTM checkpoint for the fixed K

Stage 1: DeLaN model selection

Set $\lambda = 0.5$, $P = 10.0$ ▷ stability weight, divergence penalty

for all $h \in \mathcal{H}_{\text{DeLaN}}$ **do**

- Initialize fold summaries $\mathcal{F}_h \leftarrow \emptyset$
- for all** $d \in \mathcal{D}$ **do**

 - Initialize run buffers $\mathcal{R} \leftarrow \emptyset$, curves $\mathcal{C}_{\text{train}}, \mathcal{C}_{\text{val}} \leftarrow \emptyset$
 - for all** $s \in \mathcal{S}_{\text{DeLaN}}$ **do**

 - Train DeLaN on fold d with hyper-set h and seed s ▷ early stopping on val MSE
 - if** training crashed or NaN/Inf or missing metrics **then**

 - Mark diverged and store placeholder metrics

 - else**

 - Record curves $\text{train_loss}(e)$, $\text{val_mse}(e)$, motor-current RMSE over progress, motor-current RMSE per joint, and $\text{RMSE}_{\text{val}}, \text{RMSE}_{\text{test}}, e^*$
 - Append curves and run metrics to $\mathcal{C}_{\text{train}}, \mathcal{C}_{\text{val}}, \mathcal{R}$ ▷ includes motor-current RMSE progress and per-joint traces

 - Align curves to common length E_{max} and aggregate median \pm IQR
 - Compute $\tilde{r}_{\text{val}}(d, h), q_{\text{val}}(d, h), \tilde{r}_{\text{test}}(d, h)$ over non-diverged runs
 - Compute divergence rate $\rho(d, h) = \#\text{diverged}/|\mathcal{S}_{\text{DeLaN}}|$
 - Score $s(h, d) = \tilde{r}_{\text{val}}(d, h) + \lambda q_{\text{val}}(d, h) + P \rho(d, h)$
 - Choose best seed s^* with minimal RMSE_{val} among non-diverged runs
 - Save fold artifacts for (h, d) ▷ median \pm IQR curves; best-run plots; checkpoint (h, d, s^*)
 - Append fold summary $(\tilde{r}_{\text{val}}, q_{\text{val}}, \tilde{r}_{\text{test}}, \rho, s)$ to \mathcal{F}_h

Aggregate across dataset seeds for h :

$S_h = \text{median}_d(s(h, d)), Q_h = \text{IQR}_d(s(h, d)), x_h = \text{median}_d(\tilde{r}_{\text{val}}(d, h)), y_h^{\text{test}} = \text{median}_d(\tilde{r}_{\text{test}}(d, h))$

Save hyper-set summary for h and add one point to scatter datasets

Choose best hyper-set $h^* = \arg \min_h S_h$

Plot scatter A: x vs stability (median or IQR of q_{val} across d)

Plot scatter B: x vs y_h^{test} ▷ validation vs test alignment

Choose final DeLaN checkpoint: $d^* = \arg \min_d s(h^*, d)$, use saved (h^*, d^*, s^*)

Freeze DeLaN and export residuals

Freeze best DeLaN parameters and export residual motor currents per trajectory

After selecting and freezing the best DeLaN checkpoint (Algorithm 2), Stage 2 addresses the remaining modelling error by learning residual motor currents. For each time index k , the residual current is defined as

$$\mathbf{r}_{i,k} = \mathbf{i}_{\text{motor},k} - \hat{\mathbf{i}}_{\text{DeLaN},k}. \quad (53)$$

In contrast to Stage 1, where architecture and training are kept fixed, the LSTM performance can depend strongly on the available history length H and on which signals are provided as inputs. We therefore treat the residual learner as a dedicated best-model selection problem under a fixed DeLaN baseline.

The goal is to choose a configuration (f, H) and corresponding checkpoints such that the end-to-end prediction error of the combined motor-current model

$$\hat{\mathbf{i}}_{\text{RG},k} = \hat{\mathbf{i}}_{\text{DeLaN},k} + \hat{\mathbf{r}}_{i,k} \quad (54)$$

is minimised on held-out trajectories, while remaining robust to the dataset split induced by the dataset seed d and the LSTM initialisation seed. Since the LSTM operates on history windows, evaluation is performed only on indices with valid context ($k \geq H - 1$).

Algorithm 3 sweeps three window lengths $H \in \{50, 100, 150\}$ and four feature modes f . The feature modes are designed to isolate whether the residual motor-current dynamics are primarily explained by kinematics, by the DeLaN current baseline, or by their combination:

full $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \hat{\mathbf{i}}_{\text{DeLaN}})$; tests whether the residual benefits from both kinematics and the DeLaN baseline prediction.

i_hat $(\hat{\mathbf{i}}_{\text{DeLaN}})$; tests whether the residual is largely a systematic correction correlated with the DeLaN prediction alone.

state $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$; tests whether kinematics alone is sufficient to explain the residual, without access to the DeLaN baseline.

state_ihat $(\dot{\mathbf{q}}, \ddot{\mathbf{q}}, \hat{\mathbf{i}}_{\text{DeLaN}})$; tests whether absolute position can be omitted while retaining velocity/acceleration information conditioned on the DeLaN baseline prediction.

For each (d, f, H) , multiple LSTM seeds are trained with early stopping, and both residual-level metrics and combined-model motor-current RMSE are recorded.

To select a configuration that is both accurate and reliable, we rank (f, H) by a robustness-aware score based on the combined-model motor-current RMSE: for each dataset seed d , we aggregate across LSTM seeds using median and IQR, and add a divergence penalty for failed runs (NaN/Inf or missing metrics). The resulting per-seed score $s(d, f, H)$ is then aggregated across dataset seeds, and the final choice is $(f^*, H^*) = \arg \min_{(f, H)} S_{f, H}$ as defined in Algorithm 3.

Algorithm 3 Best-model selection for the LSTM residual model

Require: Dataset seeds \mathcal{D} , feature modes $\mathcal{F} = \{\text{full}, \text{state}, \text{i_hat}, \text{state_ihat}\}$, window lengths $\mathcal{H} = \{50, 100, 150\}$

Require: LSTM seeds $\mathcal{S}_{\text{LSTM}}$, stability weight $\lambda = 0.5$, divergence penalty $P = 10.0$

Require: Residual datasets exported from the best DeLaN for each $d \in \mathcal{D}$

Ensure: Best LSTM checkpoint and best (f, H) configuration

Build LSTM datasets (once per (d, f, H))

for all $d \in \mathcal{D}$ **do**

- Load residual dataset for d
- for all** $f \in \mathcal{F}$ **do**

 - for all** $H \in \mathcal{H}$ **do**

 - Build LSTM window NPZ for (d, f, H) ▷ features by f , window length H

LSTM training and per-seed combined evaluation

for all $d \in \mathcal{D}$ **do**

for all $f \in \mathcal{F}$ **do**

for all $H \in \mathcal{H}$ **do**

- Initialize run buffers $\mathcal{R} \leftarrow \emptyset$, curves $\mathcal{C}_{\text{train}}, \mathcal{C}_{\text{val}} \leftarrow \emptyset$
- for all** $\ell \in \mathcal{S}_{\text{LSTM}}$ **do**

 - Train LSTM on NPZ(d, f, H) with seed ℓ ▷ early stopping; save best checkpoint
 - if** training crashed or NaN/Inf or missing metrics **then**

 - Mark diverged

else

- Record curves $\text{train_loss}(e), \text{val_loss}(e)$ and e^*
- Record residual metrics $\text{RMSE}_{\text{res}}, \text{MSE}_{\text{res}}$
- Run combined evaluation with DeLaN baseline for d and this LSTM
- Record $\text{RMSE}_{\text{rg}}, \text{gain}, \text{gain_ratio}$

Append curves and run metrics to $\mathcal{C}_{\text{train}}, \mathcal{C}_{\text{val}}, \mathcal{R}$

Align curves to common length E_{max} and aggregate median \pm IQR

Compute medians and IQRs over non-diverged runs:

$\tilde{r}_{\text{res}}(d, f, H), q_{\text{res}}(d, f, H), \tilde{r}_{\text{rg}}(d, f, H), q_{\text{rg}}(d, f, H), \tilde{g}(d, f, H), q_g(d, f, H)$

Compute divergence rate $\rho(d, f, H) = \#\text{diverged}/|\mathcal{S}_{\text{LSTM}}|$

Score $s(d, f, H) = \tilde{r}_{\text{rg}}(d, f, H) + \lambda q_{\text{rg}}(d, f, H) + P \rho(d, f, H)$

Save config artifacts for (d, f, H) ▷ median \pm IQR curves and metrics; score

Aggregate across dataset seeds (per (f, H))

for all $f \in \mathcal{F}$ **do**

for all $H \in \mathcal{H}$ **do**

- Compute summary across d :
- $S_{f,H} = \text{median}_d(s(d, f, H)), Q_{f,H} = \text{IQR}_d(s(d, f, H)), \tilde{r}_{\text{rg}}(f, H) = \text{median}_d(\tilde{r}_{\text{rg}}(d, f, H)), \tilde{g}(f, H) = \text{median}_d(\tilde{g}(d, f, H))$

Save per- (f, H) summary and add to boxplots / progress-curve aggregation

Final selection

Choose best $(f^*, H^*) = \arg \min_{(f,H)} S_{f,H}$

Report boxplots and progress-normalized RMSE curves for best configs

5 Experimental Results

This chapter reports experimental results for the proposed two-stage pipeline, covering the effect of the number of available trajectories (K) on DeLaN and residual LSTM training dynamics and accuracy. A quantitative comparison to the dataset baseline and a best-model evaluation on “with load” and “without load” trajectories.

5.1 Learning Curve Results

This section reports learning-curve experiments in which the number of available trajectories K is varied to quantify how trajectory coverage affects optimisation dynamics and motor-current prediction accuracy for both stages of the pipeline. Across the presented losses and RMSE metrics, smaller K values exhibit higher dispersion across dataset seeds and more pronounced error variability, while larger K values yield more consistent convergence behaviour and lower typical RMSE. The final comparison figures additionally report the per-joint test RMSE for DeLaN versus the combined DeLaN+LSTM predictor for representative runs.

5.1.1 K-Domination Results

Stage 1 (DeLaN): learning dynamics Figures 6 and 7 summarise the Stage 1 optimisation as a function of the number of trajectories K . Small trajectory sets lead to substantially higher training loss and validation error, and also to markedly larger variability across dataset seeds. In contrast, for $K \geq 32$ the curves are closely clustered over the full training horizon and exhibit similar convergence behaviour.

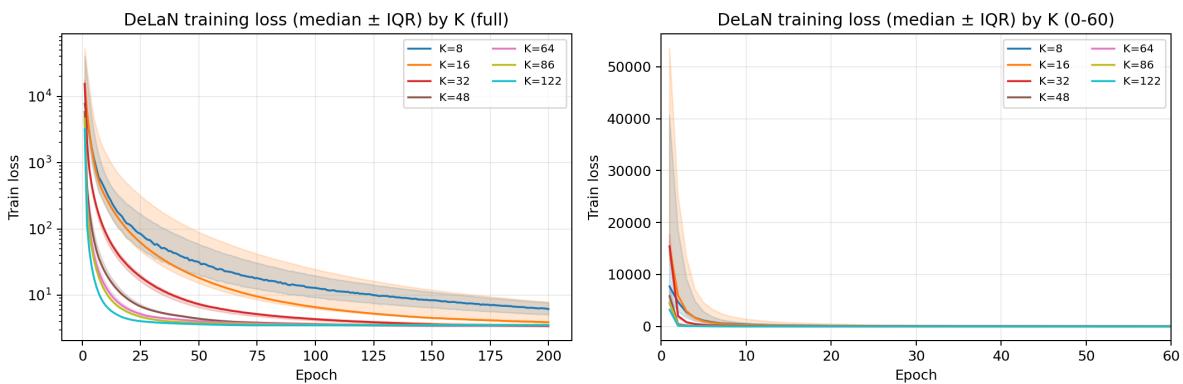


Figure 6: DeLaN training loss by K shown as median \pm IQR across dataset seeds (with seed-wise aggregation across DeLaN initialisations).

Figure 6 shows that the training loss decreases rapidly during the first epochs for all K and then continues to decrease more gradually. Across the full training horizon, smaller K values remain at higher loss levels and exhibit wider IQR bands than larger K values.

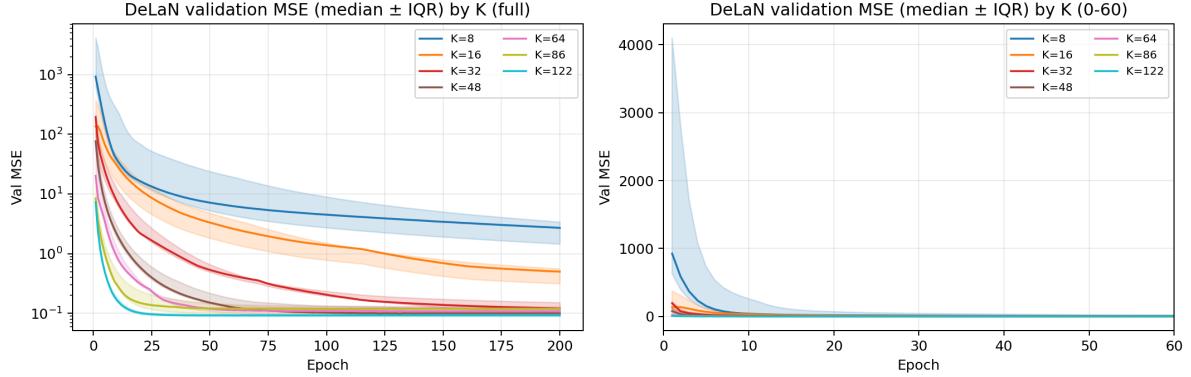


Figure 7: DeLaN validation MSE (motor current) by K shown as median \pm IQR across dataset seeds (with seed-wise aggregation across DeLaN initialisations).

Figure 7 reports the corresponding validation MSE. Validation MSE decreases for all K and separates clearly by trajectory count: smaller K values attain higher validation errors and wider interquartile ranges, whereas the largest settings concentrate at lower validation MSE.

Figure 8 reports the motor-current RMSE along the normalised trajectory progress. The smallest setting ($K = 8$) exhibits pronounced error spikes and large IQR, with peaks exceeding 3 A in the early part of the motion. Increasing K substantially reduces both the typical error level and its variability. For larger K , the median curves remain close to each other and the IQR bands narrow over most of the progress range.

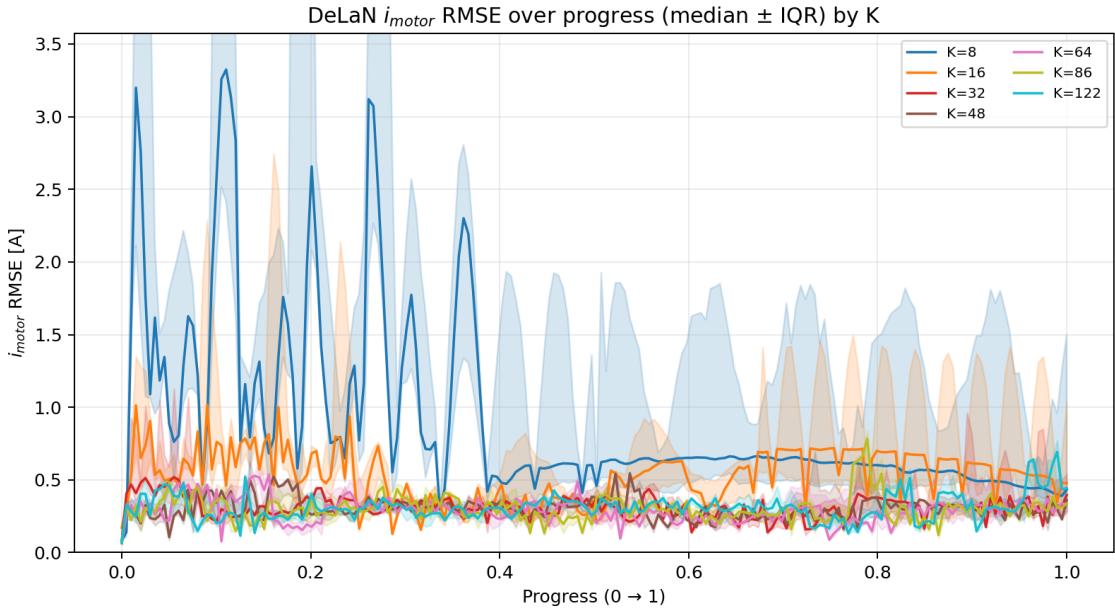


Figure 8: DeLaN motor-current RMSE over normalised progress ($0 \rightarrow 1$) by K shown as median \pm IQR (shaded bands).

Figure 9 reports per-joint motor-current RMSE for all evaluated values of K . For $K = 8$, the error is dominated by joints 3 and 5, where the median RMSE exceeds 3 A. For intermediate K (like $K = 16$), joint 4 exhibits a pronounced increase in RMSE relative to the other joints. For larger K , the per-joint medians concentrate within a narrower range and the IQR bars decrease across joints.

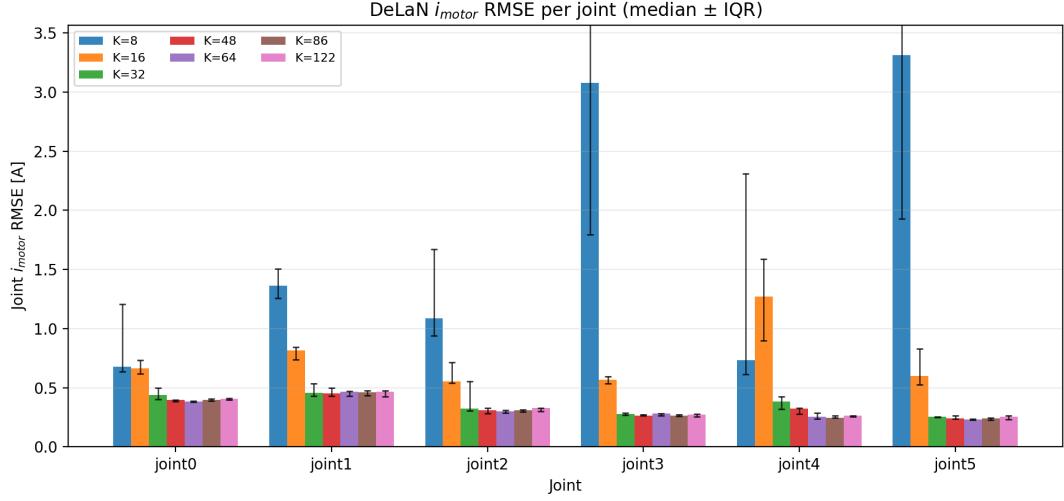


Figure 9: DeLaN motor-current RMSE per joint (median \pm IQR) for all evaluated values of K (bars) with IQR error bars.

Figures 10 and 11 show the Stage 2 training and validation loss of the residual LSTM. While the training loss decreases rapidly for all K , the validation loss shows a clear dependence on the number of trajectories: larger K yields lower validation loss and narrower IQR bands across the training horizon.

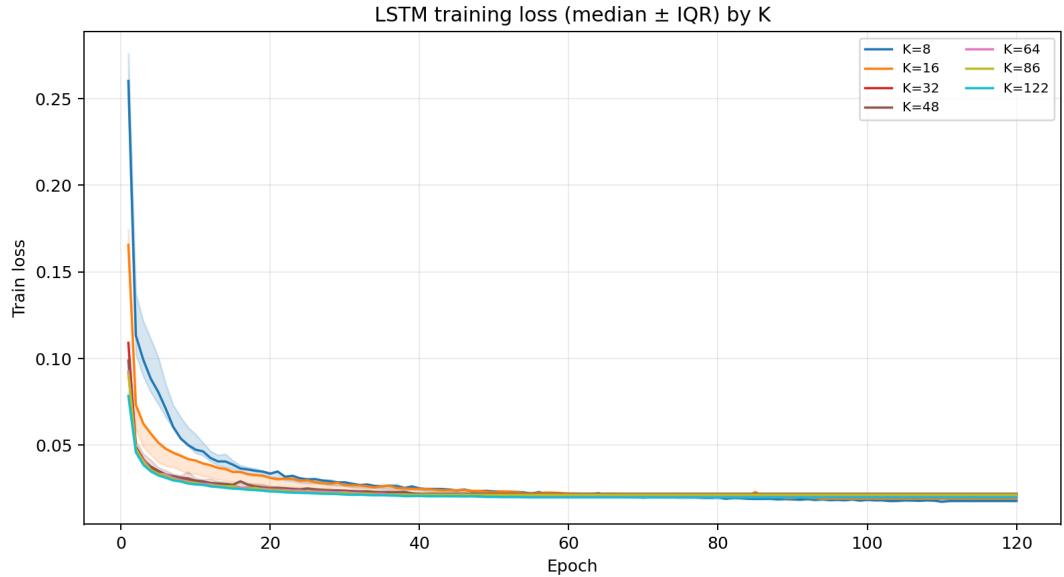


Figure 10: LSTM training loss by K shown as median \pm IQR across dataset seeds ($H = 100$, feature mode full).

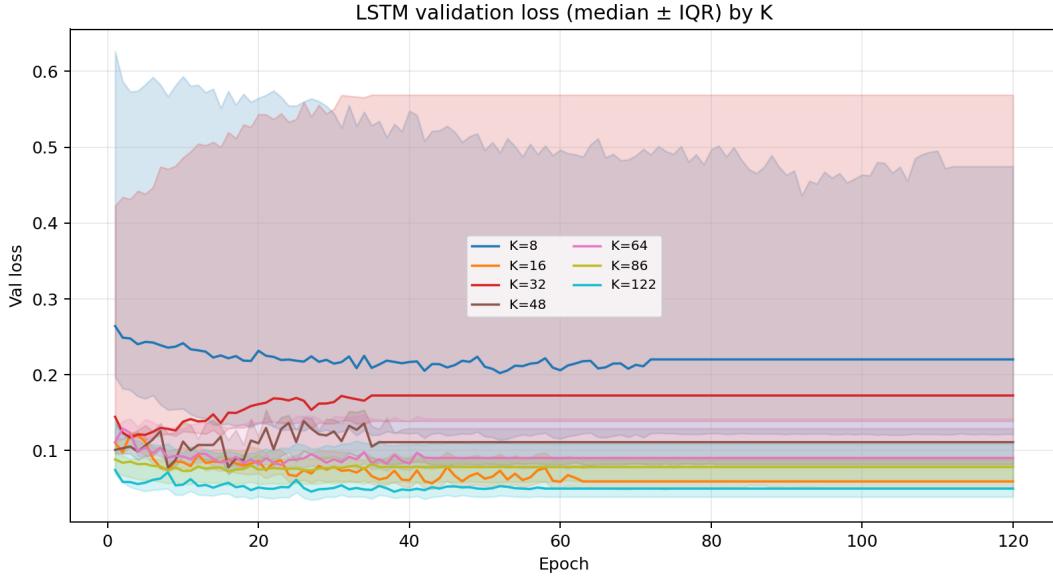


Figure 11: LSTM validation loss by K shown as median \pm IQR across dataset seeds ($H = 100$, feature mode `full`).

Figure 13 shows the residual motor-current RMSE over normalised trajectory progress, and Figure 12 summarises per-joint residual errors. The smallest setting ($K = 8$) exhibits large residual spikes and wide IQR bands, whereas larger K values concentrate at substantially smaller residual RMSE levels across progress and across joints.

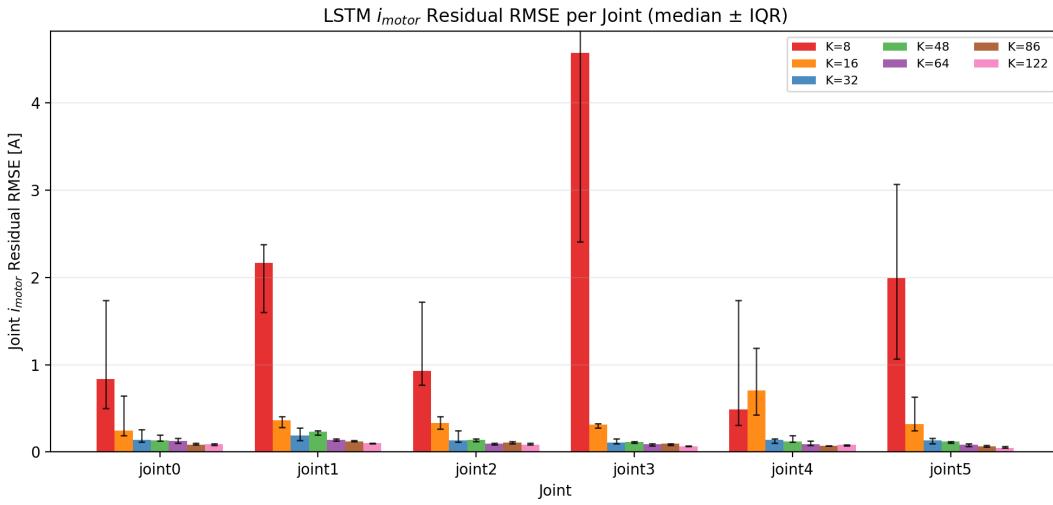


Figure 12: LSTM residual motor-current RMSE per joint (median \pm IQR) for selected values of K ($H = 100$, feature mode `full`).

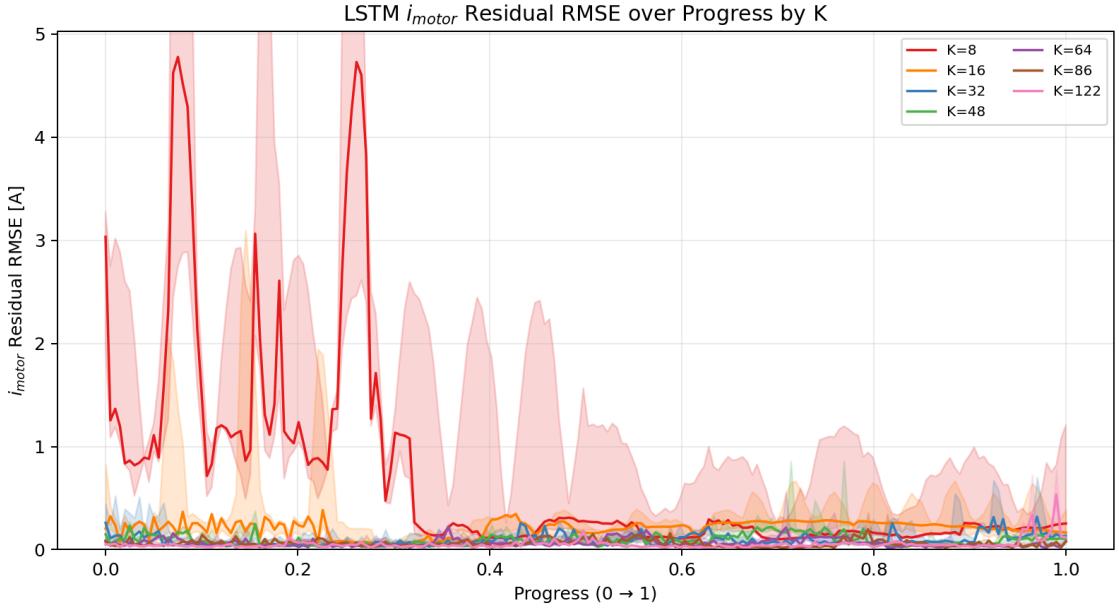


Figure 13: LSTM residual motor-current RMSE over normalised progress ($0 \rightarrow 1$) by K shown as median \pm IQR (shaded bands) ($H = 100$, feature mode `full`).

Figure 14 reports per-joint motor-current RMSE on the test split for a representative evaluation (valid indices $k \geq H - 1$). Across all joints, the combined DeLaN+LSTM predictor attains lower RMSE than the DeLaN baseline, with the largest absolute differences observed on joints 0–2.

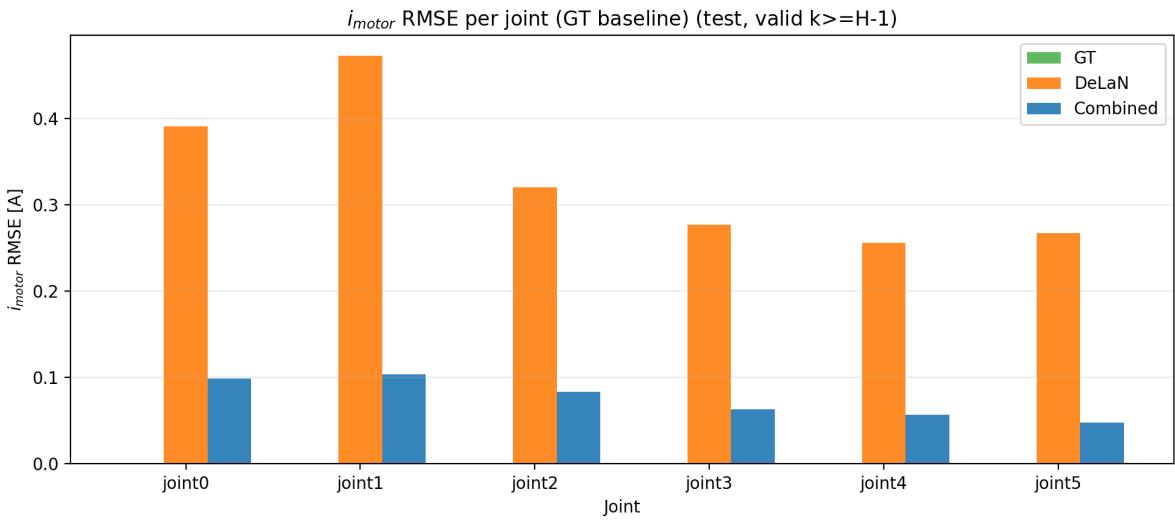


Figure 14: Representative per-joint motor-current RMSE on the test split comparing DeLaN and the combined DeLaN+LSTM predictor (valid indices $k \geq H - 1$, $H = 100$).

5.1.2 Best Model Approach Results

Figures 15–18 summarise the DeLaN best-model sweep at fixed $K = 84$ on the 50,000-sample UR3_Load0Dataset training pool. All metrics in this subsection are computed on this pool,

which is split into 70% training, 10% validation, and 20% test subsets for model selection. Results are aggregated across dataset seeds with seed-wise aggregation across DeLaN initializations.

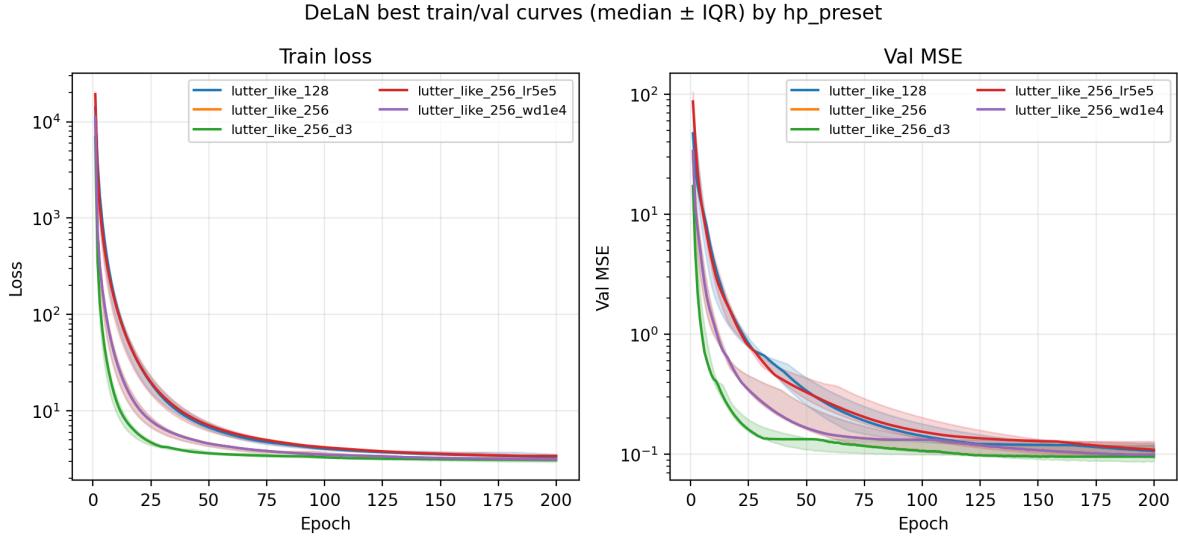


Figure 15: DeLaN training loss (left) and validation MSE in motor-current units (right) shown as median \pm IQR across seeds for each hyperparameter preset.

Figure 15 shows that all presets exhibit a rapid initial decrease in training loss and validation MSE, followed by a slower decay over the remaining epochs. Across presets, the curves separate consistently both in training loss and in the final validation MSE plateau.

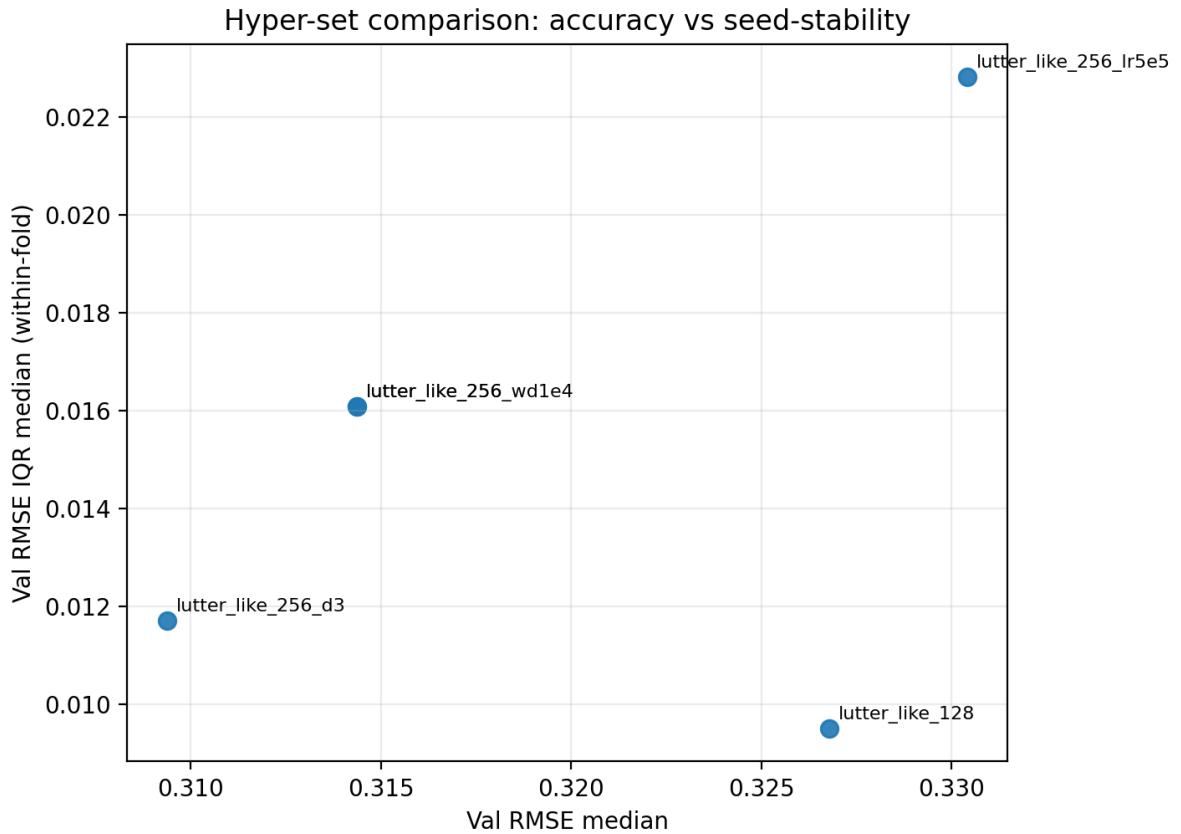


Figure 16: Hyperparameter comparison: median validation motor-current RMSE versus seed-stability measured by the median within-split IQR of validation RMSE (each point denotes one DeLaN hyperparameter preset. Statistics aggregated across dataset seeds with seed-wise aggregation across DeLaN initialisations).

Figure 16 reports one point per preset in the plane of (median validation RMSE, median within-split IQR of validation RMSE). The points span a validation RMSE range of approximately 0.309–0.330 A and a stability range of approximately 0.009–0.023 A.

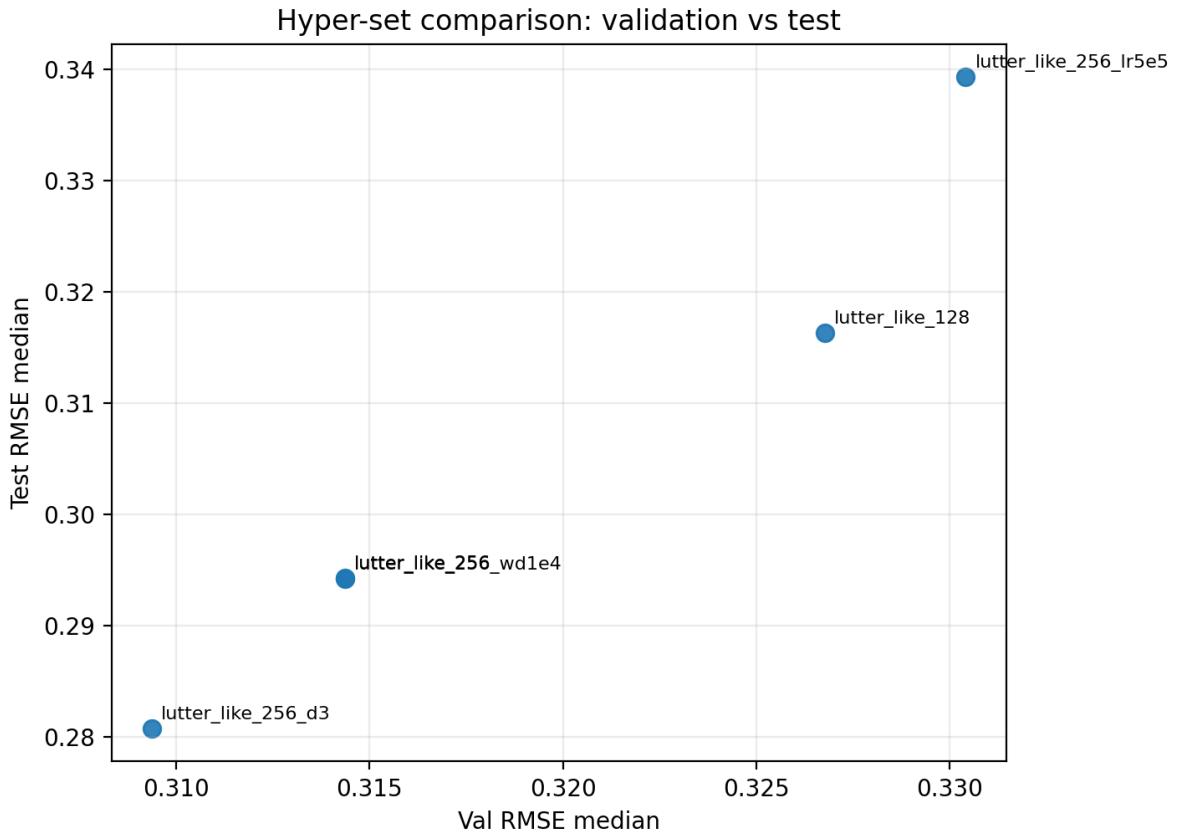


Figure 17: Hyperparameter comparison: median validation motor-current RMSE versus median test motor-current RMSE (each point denotes one DeLaN hyperparameter preset. Statistics aggregated across dataset seeds with seed-wise aggregation across DeLaN initialisations).

Figure 17 shows the corresponding (median validation RMSE, median test RMSE) pairs for each preset, with the observed points spanning roughly 0.309–0.330 A on the validation axis and 0.280–0.340 A on the test axis.

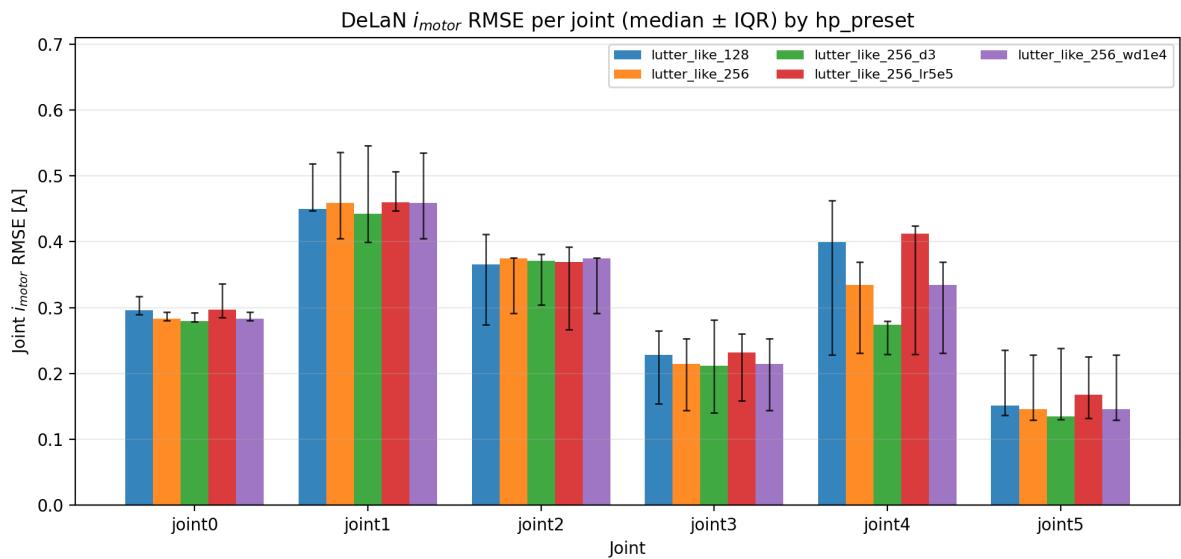


Figure 18: DeLaN motor-current RMSE per joint (median \pm IQR) grouped by hyperparameter preset.

Figure 18 reports per-joint motor-current RMSE by preset. Across presets, joint 1 exhibits the largest median RMSE, followed by joints 2 and 4, whereas joint 5 attains the lowest median RMSE. The IQR bars are largest on joints 1 and 4.

Figures 19–23 summarise the LSTM best-model sweep at fixed $K = 84$ based on the residual datasets exported from the selected DeLaN baseline.

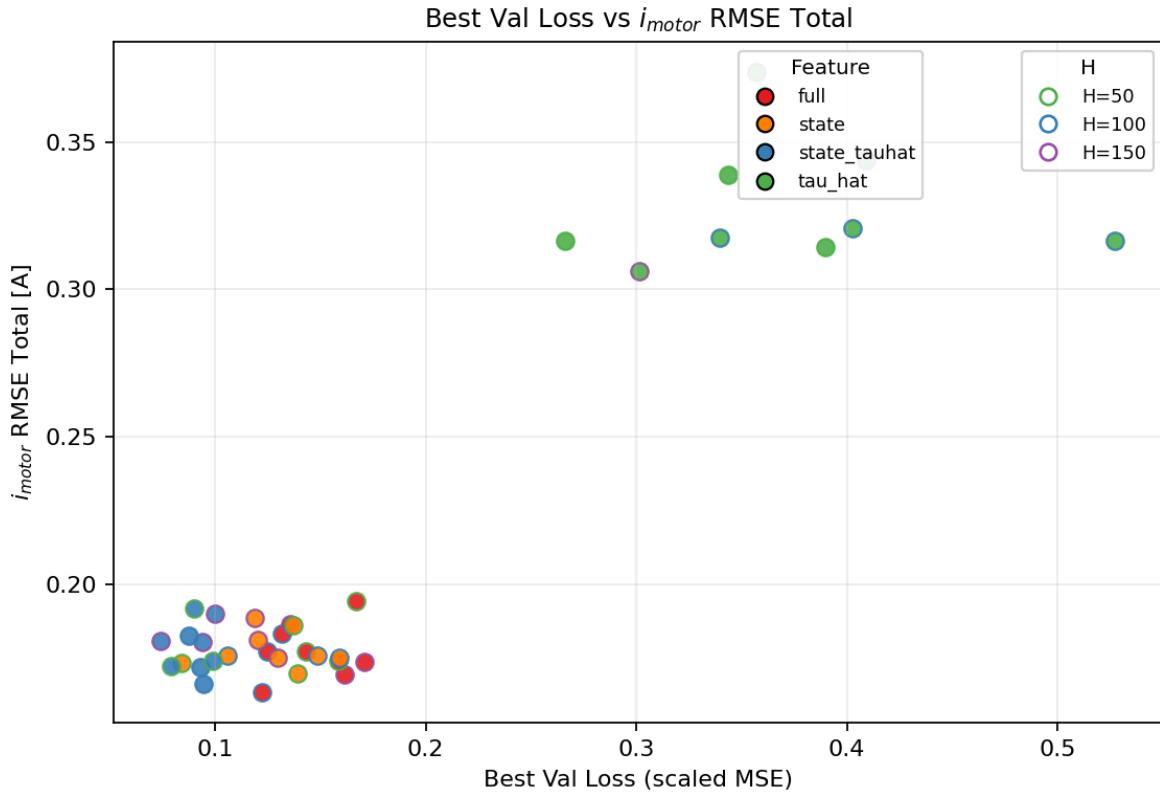


Figure 19: Best validation loss versus total motor-current RMSE on the test split for the LSTM residual model (marker colour denotes feature mode, marker outline denotes history length H).

Figure 19 shows that most configurations cluster at low test RMSE (approximately 0.165–0.195 A) and low best validation loss (approximately 0.07–0.18), while a smaller set of configurations attains markedly higher test RMSE values around 0.31–0.34 A.

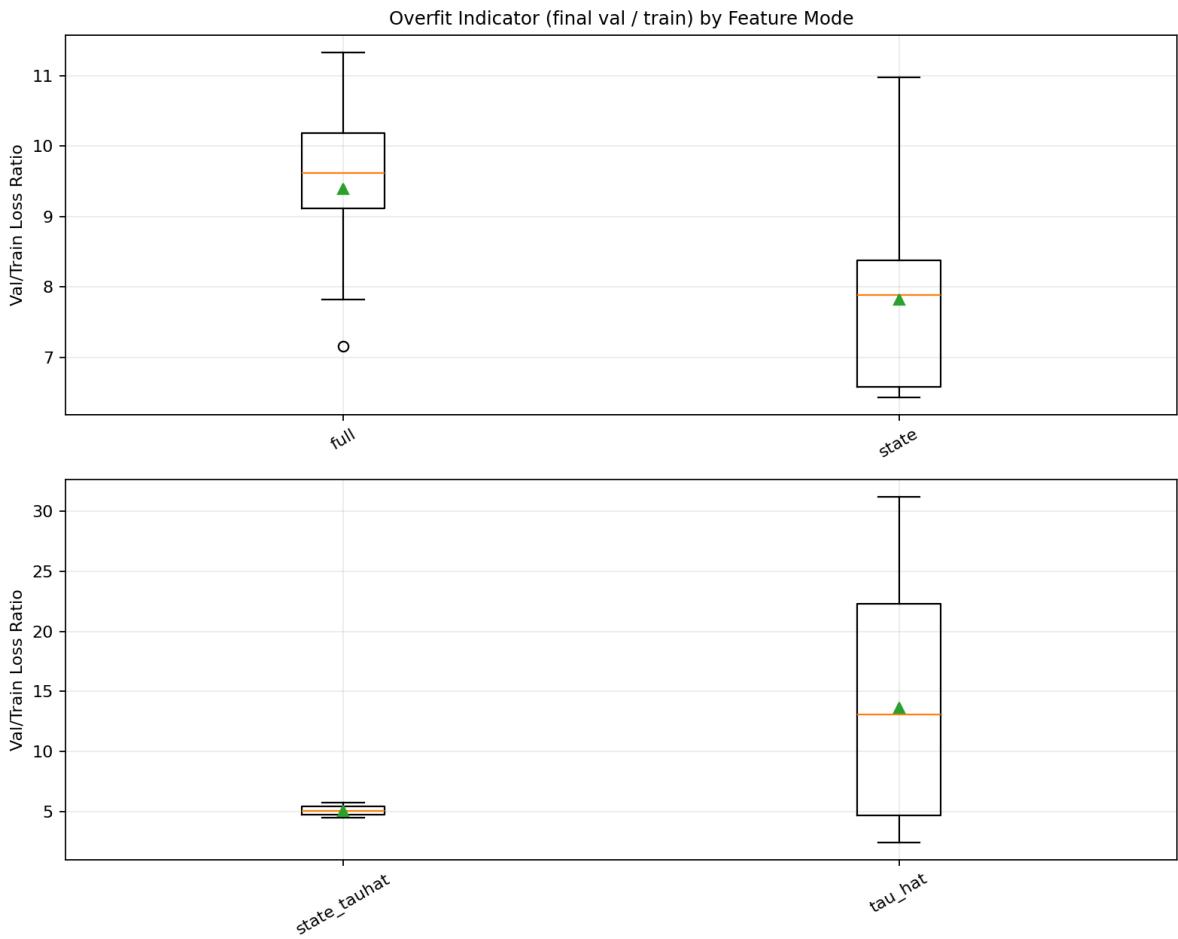


Figure 20: Overfit indicator (final validation loss / final training loss) by feature mode shown as boxplots across runs (green triangles indicate means, panels use different y -axis ranges for readability).

Figure 20 reports the distribution of the final validation-to-training loss ratio across feature modes. Across the lower-range panel, the medians lie around 5–10 depending on the mode, while the high-range panel includes ratios extending beyond 30.

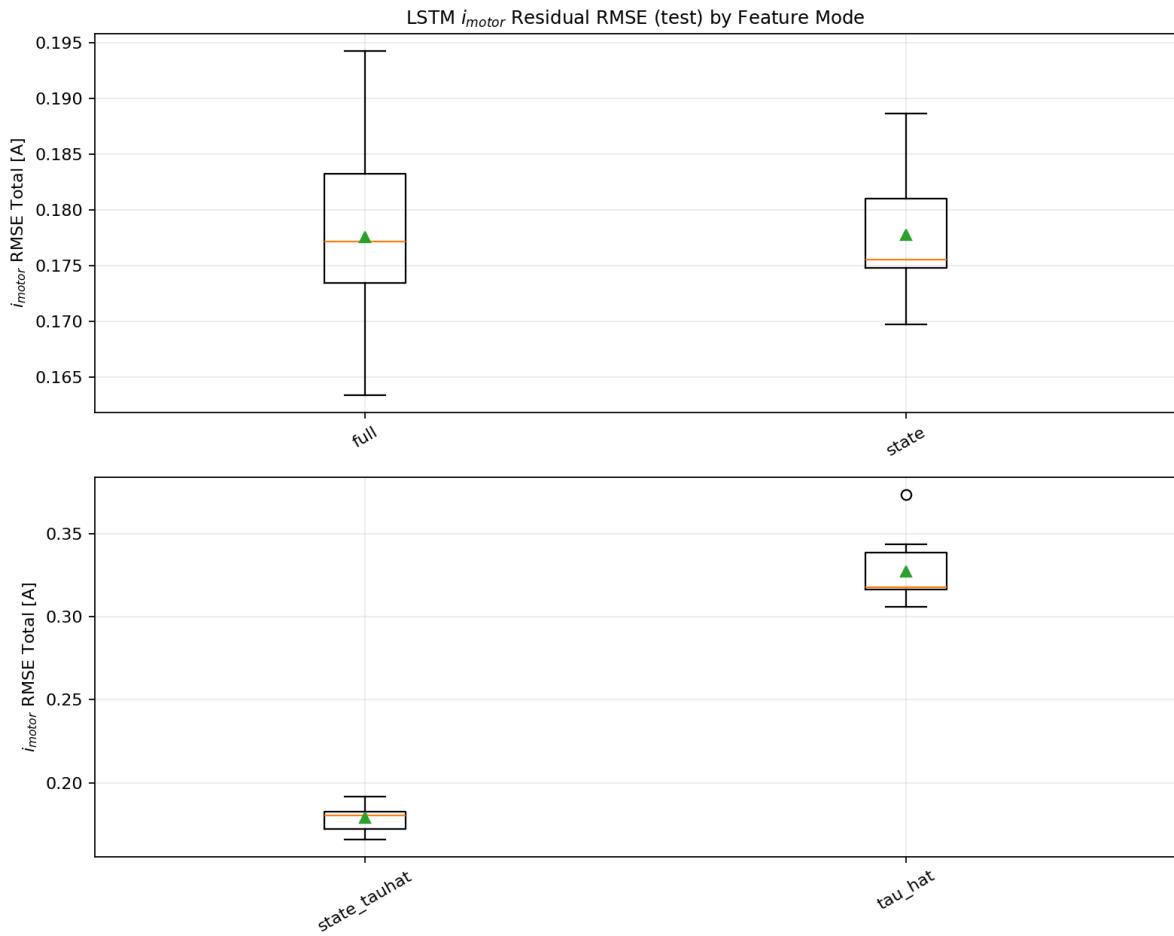


Figure 21: LSTM residual motor-current RMSE on the test split by feature mode shown as boxplots across runs (green triangles indicate means, panels use different y -axis ranges for readability).

Figure 21 reports residual RMSE on the test split by feature mode. In the lower-range panel, the residual RMSE distributions are concentrated around 0.17–0.19 A, whereas the high-range panel contains configurations with residual RMSE around 0.31–0.37 A.

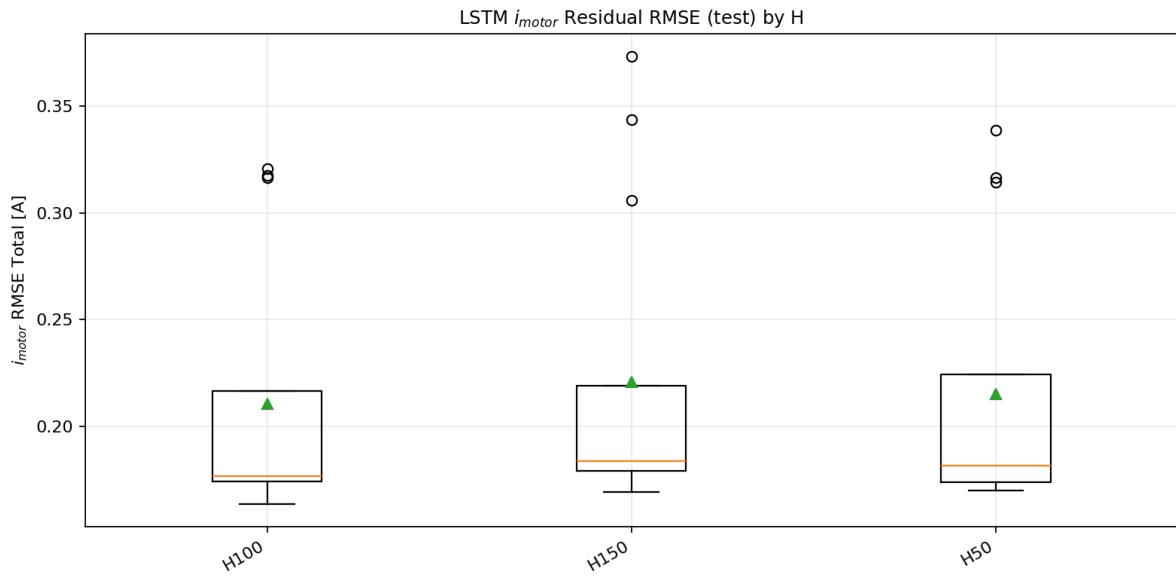


Figure 22: LSTM residual motor-current RMSE on the test split shown as boxplots across history lengths H (green triangles indicate means).

Figure 22 shows residual RMSE grouped by history length H . Across $H \in \{50, 100, 150\}$, the medians lie in a similar range and the boxplots exhibit comparable spread, with outliers present for all three history lengths.

i_{motor} residual: GT vs LSTM (test, valid $k \geq H-1$)

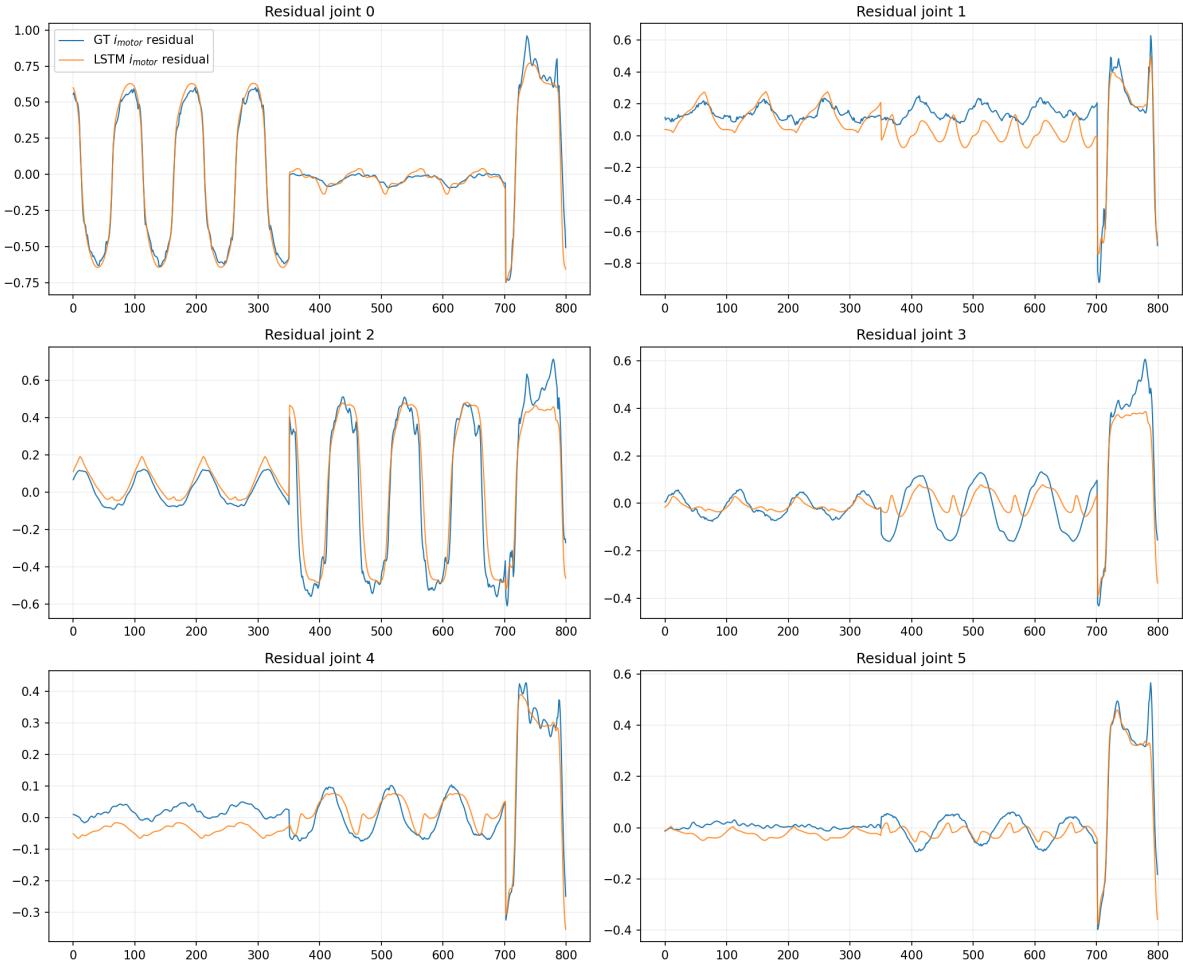


Figure 23: Representative residual motor-current traces: ground-truth residual versus LSTM residual prediction on the test split (valid indices $k \geq H - 1$, $H = 150$).

Figure 23 provides a representative residual overlay for all joints, showing the ground-truth residual and the corresponding LSTM prediction over time.

Figure 24 reports the per-joint motor-current RMSE on the test split for the selected DeLaN baseline and for the combined DeLaN+LSTM predictor (valid indices $k \geq H - 1$, $H = 150$). Across joints, the DeLaN bars range from approximately 0.12 to 0.46 A, whereas the combined predictor ranges from approximately 0.07 to 0.25 A. The ground-truth baseline is zero by construction and is therefore not visible as a bar. The combined model attains lower RMSE than the DeLaN baseline on all joints, with the largest absolute reduction on joint 1 (the UR3 shoulder-lift joint), which is a major load-bearing joint of the arm. The largest RMSE is observed on joint 1 and the smallest on joint 5 for both models.

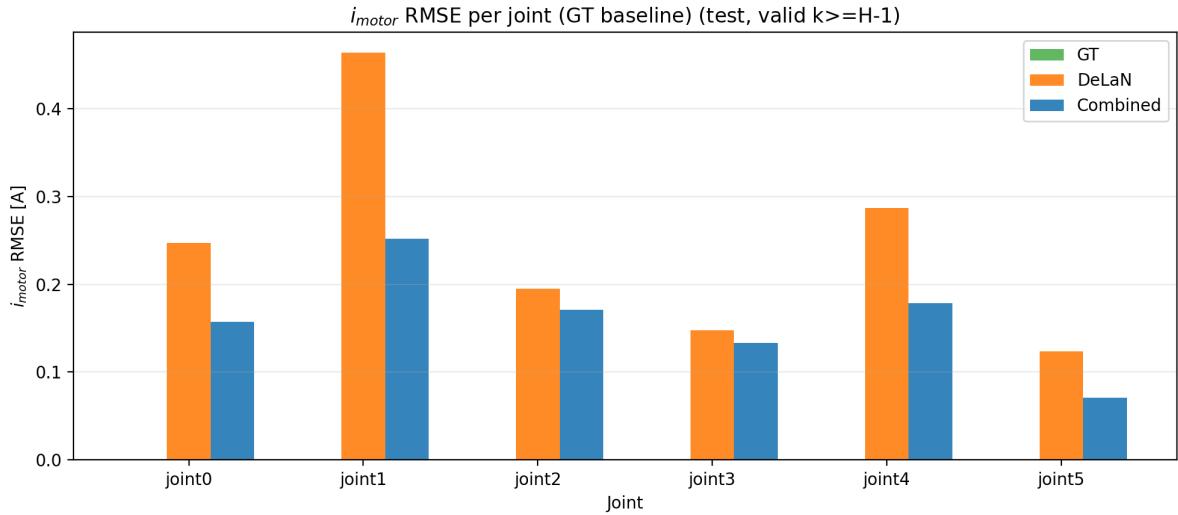


Figure 24: Best pipeline motor-current RMSE per joint on the test split comparing DeLaN and the combined DeLaN+LSTM predictor (valid indices $k \geq H - 1$, $H = 150$).

Figure 25 shows the corresponding motor-current time-series overlay for the same evaluation, plotting the ground truth together with the DeLaN prediction and the combined DeLaN+LSTM prediction for each joint.

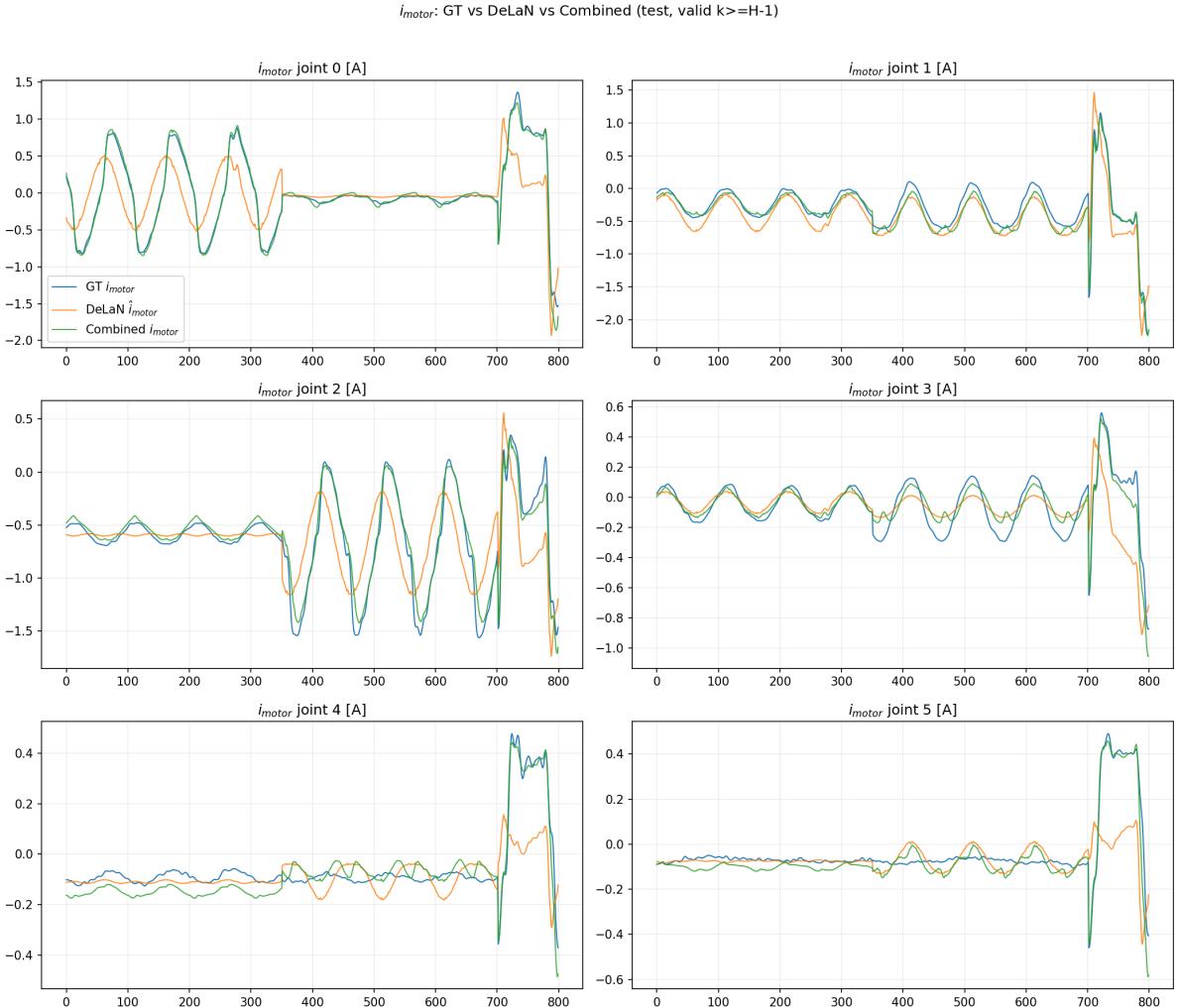


Figure 25: Representative motor-current overlay: ground truth, DeLaN prediction, and combined DeLaN+LSTM prediction on the test split (valid indices $k \geq H - 1$, $H = 150$).

5.2 Performance Evaluation to Baseline

To benchmark the proposed DeLaN+LSTM pipeline against prior work on the same dataset, we compare against the data-driven dynamic model presented in [21] and its accompanying IEEE DataPort release [41]. The baseline study likewise applies low-pass filtering to the recorded signals as part of its preprocessing [21]. The baseline identifies a joint-wise, linear-in-parameters model of the motor current i_i from measured joint positions, velocities, and accelerations, and estimates the unknown parameter vector via least squares:

$$\mathbf{K}_i = \left(\boldsymbol{\Theta}_{m,i}^\top \boldsymbol{\Theta}_{m,i} \right)^{-1} \boldsymbol{\Theta}_{m,i}^\top \mathbf{i}_{m,i}, \quad (55)$$

where $\Theta_{m,i}$ is the measurement matrix of the regressor terms for joint i and $\mathbf{i}_{m,i}$ denotes the corresponding measured motor-current samples [21]. Evaluation is reported as per-joint RMSE in current units,

$$\text{RMSE}(i_i) = \sqrt{\frac{1}{n} \sum_{k=1}^n (i_{i,k}^{\text{real}} - i_{i,k}^{\text{pred}})^2}. \quad (56)$$

Table 2 reproduces the current-prediction RMSE reported by [21] for UR3e and UR10e, each without load and with load, for the provided testing datasets. In the baseline study, the model is fitted on 50,000 training samples and evaluated on 5,000 unseen test samples [21]. Across conditions, UR3e baseline RMSE values are on the order of 0.06–0.38 A, while UR10e baseline RMSE values range up to 1.56 A on the test set. The UR3e without-load DeLaN+LSTM model corresponds to the resulting best-model configuration selected in the Best Model Approach (Subsection 5.1.2). The UR3e with-load and UR10e without-load DeLaN+LSTM models are trained using the same hyperparameter configuration identified in the Best Model Approach, and the corresponding benchmark test metrics are reported in the same table.

In addition to the dataset baseline, we report motor-current RMSE for the DeLaN and DeLaN+LSTM pipeline under the same benchmark scale as the IEEE DataPort split, means training on 50,000 samples and testing on 5,000 unseen samples. The split is constructed at the trajectory level to prevent leakage of samples from training trajectories into the test set. Benchmark-level evaluation is reported for `UR3_Load0Dataset`, `UR3_Load2Dataset`, and `UR10_Load0Dataset`, which provide the full 50,000/5,000 sample split. To align the benchmark conditions with the IEEE DataPort baseline nomenclature, we treat `UR3_Load0Dataset` as the counterpart to “UR3e without load” and `UR10_Load0Dataset` as the counterpart to “UR10e without load” in Tables 2. Analogously, we benchmark `UR3_Load2Dataset` against the baseline “UR3e with load” condition. The remaining datasets (`UR10Load1Dataset`, `UR10Load2Dataset`, and `UR3Load1Dataset`) contain substantially fewer than 50,000 total samples and are therefore not included in the benchmark-level comparison. Since the dataset provides motor current measurements, and motor current is treated as the primary actuation signal throughout this thesis, we report RMSE in current units [A] per joint. For the residual LSTM and the combined DeLaN+LSTM model, metrics are evaluated only on valid indices $k \geq H - 1$ to account for the history window warm-up.

Table 2: Baseline [21] vs. DeLaN vs. DeLaN+LSTM motor-current RMSE per joint on the benchmark test split (trained on 50,000 samples, evaluated on 5,000 unseen samples. DeLaN+LSTM evaluated on valid indices $k \geq H - 1$).

Dataset	Model	i_1 [A]	%	i_2 [A]	%	i_3 [A]	%	i_4 [A]	%	i_5 [A]	%	i_6 [A]	%
UR3e without load	Baseline	0.089	3.14	0.130	2.87	0.122	4.55	0.066	4.39	0.106	8.29	0.083	5.89
UR3e without load	DeLaN	0.247	0.764	0.464	0.555	0.195	0.386	0.147	0.525	0.287	0.969	0.124	0.982
UR3e without load	DeLaN+LSTM	0.153	0.473	0.307	0.368	0.162	0.320	0.146	0.520	0.200	0.677	0.089	0.710
UR3e with load	Baseline	0.096	3.31	0.376	5.63	0.308	5.09	0.249	8.53	0.150	8.10	0.070	4.10
UR3e with load	DeLaN	0.295	0.696	0.833	0.523	0.442	0.466	0.244	0.356	0.231	0.601	0.180	0.600
UR3e with load	DeLaN+LSTM	0.052	0.122	0.142	0.089	0.098	0.103	0.080	0.116	0.051	0.133	0.043	0.144
UR10e without load	Baseline	0.490	3.24	0.853	2.54	0.351	3.14	0.110	4.40	0.098	6.48	0.078	5.12
UR10e without load	DeLaN	1.473	0.753	2.158	0.346	0.954	0.318	0.275	0.543	0.216	1.011	0.261	0.947
UR10e without load	DeLaN+LSTM	0.316	0.161	0.705	0.113	0.350	0.117	0.075	0.148	0.076	0.355	0.073	0.264

Table 2 reports per-joint motor-current RMSE on the benchmark test split for the dataset baseline [21], the learned DeLaN predictor, and the combined DeLaN+LSTM model. For UR3e without load, the baseline yields RMSE values between 0.066 and 0.130 A across joints, while DeLaN and DeLaN+LSTM range between 0.124–0.464 A and 0.089–0.307 A, respectively. For UR3e with load, DeLaN+LSTM reports lower RMSE than both the baseline and DeLaN on all joints, with values in the range 0.043–0.142 A. For UR10e without load, DeLaN exhibits substantially larger RMSE than the other models, whereas DeLaN+LSTM reports 0.073–0.705 A across joints.

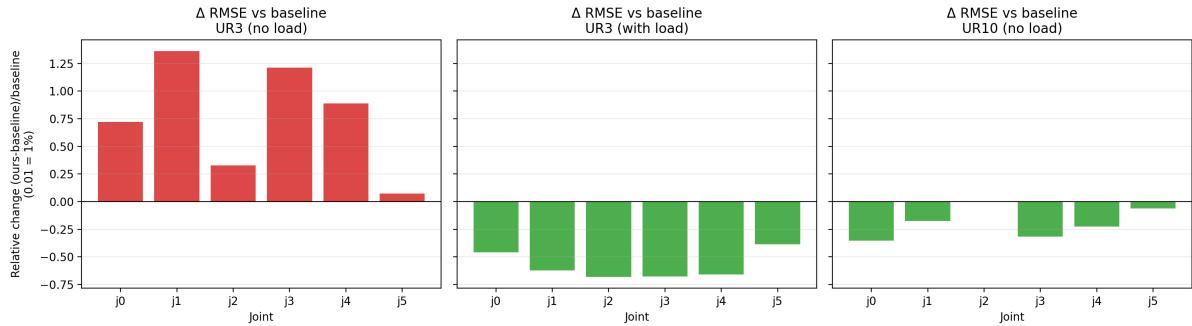


Figure 26: Relative change in per-joint motor-current RMSE of DeLaN+LSTM versus the baseline model [21], reported as $(\text{RMSE}_{\text{ours}} - \text{RMSE}_{\text{baseline}})/\text{RMSE}_{\text{baseline}}$ for UR3 (no load), UR3 (with load), and UR10 (no load). Negative values indicate a reduction in RMSE, positive values an increase.

Figure 26 summarises the joint-wise relative change of DeLaN+LSTM with respect to the baseline. For UR3 (no load), the relative change is positive for all joints and spans approximately +0.1 to +1.4. For UR3 (with load), the relative change is negative for all joints and lies approximately between -0.4 and -0.7. For UR10 (no load), the relative change is negative across joints and ranges from near 0 down to approximately -0.35.

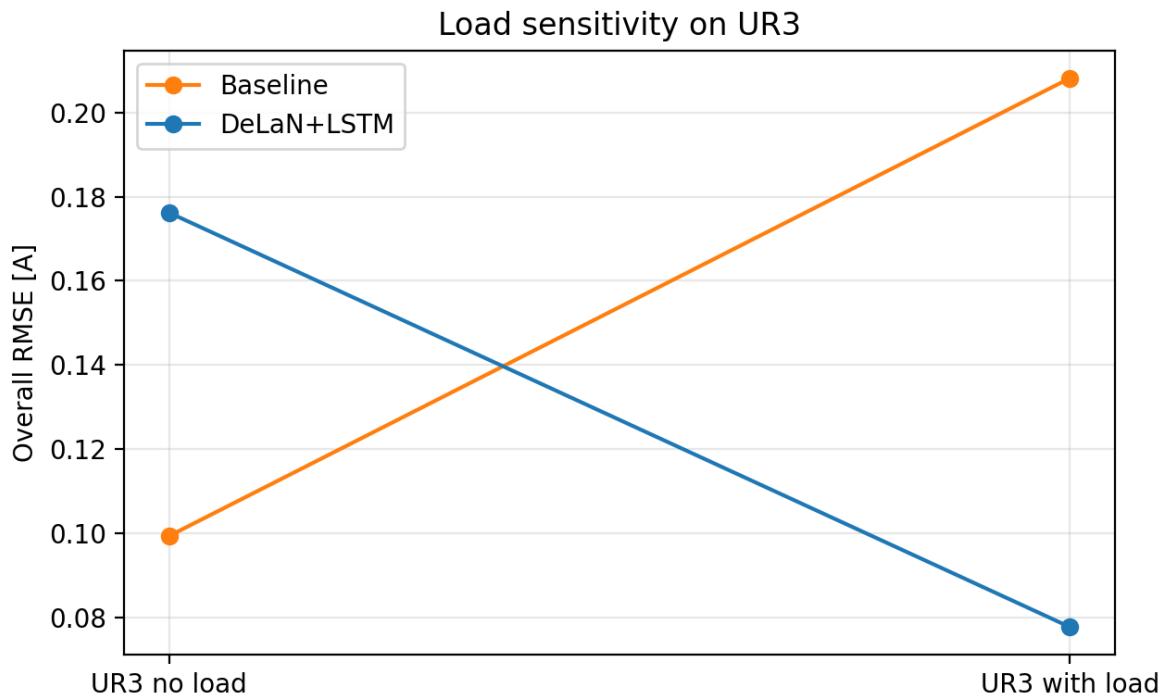


Figure 27: Load sensitivity on UR3: overall motor-current RMSE for the baseline model [21] and DeLaN+LSTM, comparing the no-load and with-load conditions.

Figure 27 reports overall motor-current RMSE on UR3 under the no-load and with-load conditions. The baseline increases from approximately 0.10 A (no load) to 0.21 A (with load), whereas DeLaN+LSTM decreases from approximately 0.18 A to 0.08 A.

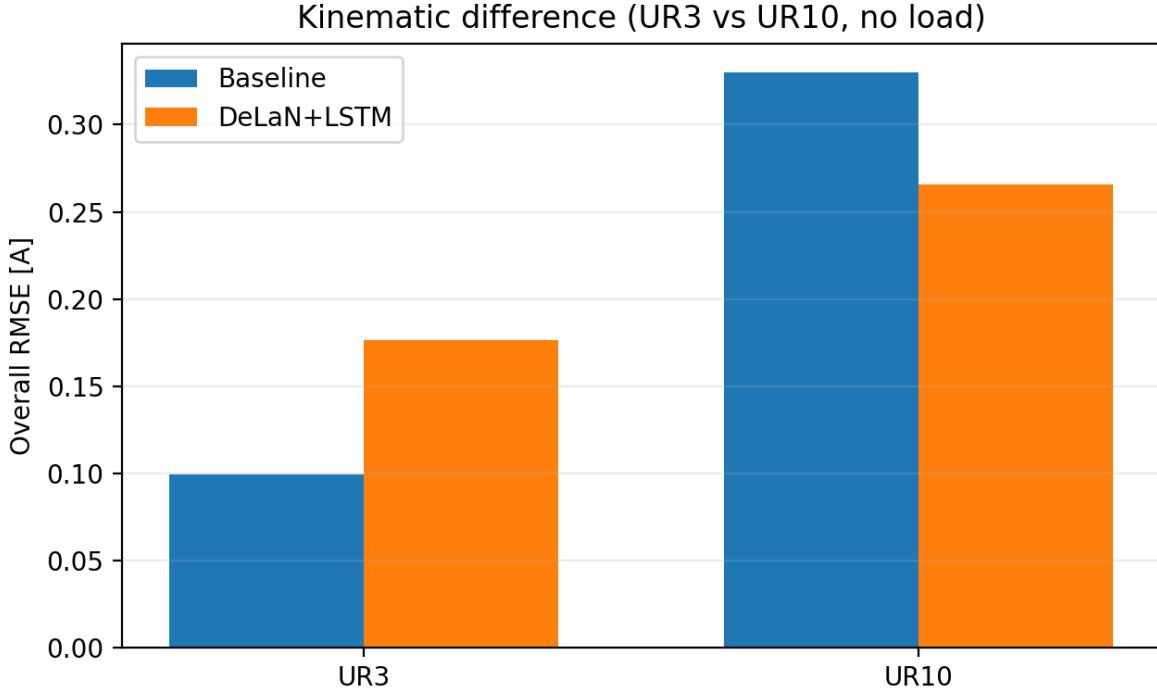


Figure 28: Kinematic difference (UR3 versus UR10, no load): overall motor-current RMSE for the baseline model [21] and DeLaN+LSTM on the no-load test split for both manipulators.

Figure 28 contrasts UR3 and UR10 on the no-load test split. For the baseline model, the overall RMSE increases from approximately 0.10 A (UR3) to 0.33 A (UR10). For the DeLaN+LSTM model, the overall RMSE increases from approximately 0.18 A (UR3) to 0.27 A (UR10).

6 Implementation

This chapter briefly describes the implementation of the proposed two-stage identification pipeline and the experimental automation used to generate the results in Chapter 5. All experiments are executed in a containerised environment (`docker compose`) [43] to ensure reproducibility and to provide a consistent GPU-enabled runtime for both Stage 1 (DeLaN) and Stage 2 (residual LSTM) [42]. In the following, the actuation signal is the measured motor current i_{motor} in A. For compatibility with the upstream DeLaN codebase [44], some internal variable names use the legacy identifier τ_{au} ; throughout the implementation, τ_{au} should be interpreted as i_{motor} .

6.1 Execution Environment (Docker)

The implementation resides in the submodule `payload_estimation/` and defines one compose stack that separates preprocessing, Stage 1 training, Stage 2 training, and evaluation into dedicated services. Preprocessing is executed in a lightweight CPU container (`python:3.11-slim`) [45], whereas the learning stages run in GPU-enabled containers. Stage 1 is implemented in JAX (Haiku/Optax) [46, 47] on top of `nvidia/cuda:12.4.1-cudnn-runtime-ubuntu22.04`, and Stage 2 as well as the evaluation scripts use the GPU TensorFlow runtime (`tensorflow/tensorflow:2.16.1-gpu`). Plot generation is configured for non-interactive execution (`MPLBACKEND=Agg`) [48] such that sweeps can run unattended. Across services, `numpy` [49] is used as the common tensor/array backbone; preprocessing relies on `scipy` for the Butterworth filtering, the residual LSTM pipeline uses `scikit-learn` for scaling utilities, and the JAX-based DeLaN container additionally installs `torch` because the upstream DeLaN codebase imports a PyTorch-based replay memory component.

All services exchange datasets and trained models via a shared host directory (`payload_estimation/shared/`) mounted into the containers. This directory contains raw data, intermediate `.npz` artefacts, trained checkpoints, evaluation outputs, and run logs. The compose configuration also maps the host user and group ID into the containers, ensuring that generated files remain writable on the host.

6.2 Implementation of the Two-Stage Pipeline

The preprocessing step converts the raw robot logs into a trajectory-wise `.npz` dataset suitable for DeLaN training. It supports trajectory-level train/validation/test splits and, for the K -domination study, subsampling a fixed number of trajectories using a seeded selection. A Butterworth low-pass filter is applied prior to derivative computation in order to stabilise acceleration and actuation signals.

Stage 1 trains a DeLaN model to predict motor currents from the joint state and its derivative acceleration. Training uses preset hyperparameters (e.g., `lutter_like_256`) and produces a checkpoint together with run metadata and learning curves. Given a selected checkpoint, the residual dataset is exported by evaluating the model on each trajectory and storing the residual signal

$$\mathbf{r}_i = \mathbf{i}_{\text{motor}} - \hat{\mathbf{i}}_{\text{DeLaN}}$$

alongside the corresponding kinematics. The exported residual `.npz` file forms the interface between Stage 1 and Stage 2.

Stage 2 trains an LSTM to predict the residual motor current from a fixed-length history of features. A windowing step constructs sliding input windows of length H and assigns the residual at the window end as target. The feature construction is shared between training and evaluation and supports multiple feature modes (state-only, DeLaN-only, and combined state+DeLaN); $\hat{\mathbf{i}}_{\text{DeLaN}}$ denotes the DeLaN-predicted motor current. During training, inputs

and targets are standardised using statistics computed on the training split, and the resulting scalers are stored together with the best model checkpoint.

Evaluation loads a residual trajectory dataset, the trained LSTM checkpoint, and the stored scalers, and reconstructs the combined estimate by

$$\hat{\mathbf{i}}_{\text{comb}} = \hat{\mathbf{i}}_{\text{DeLaN}} + \hat{\mathbf{r}}_i.$$

The evaluation scripts report MSE and RMSE for DeLaN alone, residual prediction alone, and the combined model, both aggregated across joints and per joint, and the evaluation service generates the plots used in Chapter 5.

6.3 Experimental Orchestration (Sweeps)

The experimental loops are implemented in a sweep service and are executed on the host, while the heavy computations run inside the containers. The sweep controller builds `docker compose exec` commands for preprocessing, DeLaN training, residual export, LSTM training, and evaluation, and stores a complete log of each run under `shared/logs/`. Experiment parameters such as dataset identifiers, K values, random seeds, feature modes, and history lengths are defined in a single configuration file and are therefore version-controlled together with the code.

Three sweep entrypoints correspond to the experiments reported in Chapters 4–5: the K -domination loop, the Stage 1 hyperparameter sweep for selecting a best DeLaN model, and the Stage 2 sweep over $(H, \text{feature mode})$ for selecting the best residual LSTM.

7 Discussion

The experimental results reported the experimental results in a descriptive manner. This chapter synthesises these findings and relates them back to the research questions introduced in the Introduction. In particular, we discuss how trajectory coverage K affects the accuracy and stability of the physics-structured DeLaN baseline (RQ1), how the residual LSTM reduces the remaining motor-current error and how feature choice and history length influence accuracy and overfitting behaviour (RQ2) and how the resulting two-stage predictor compares to the IEEE DataPort baseline across benchmark conditions (RQ3).

RQ1 (DeLaN accuracy and trajectory coverage). Section 5.1.1 shows that DeLaN motor-current prediction accuracy and seed stability improve systematically as trajectory coverage increases: small subsets yield higher losses, wider interquartile ranges, and pronounced RMSE spikes along progress, whereas for $K \geq 32$ the learning curves and RMSE profiles cluster more tightly. Section 5.1.2 complements this with the best-model sweep at fixed $K = 84$, demonstrating that validation-driven checkpoint selection transfers to similar test RMSE levels across the evaluated presets.

RQ2 (Residual learning, feature choice, and history length). The best-pipeline evaluation in Section 5.1.2 shows that adding an LSTM residual model reduces the remaining DeLaN motor-current error on the held-out test split across all joints (valid indices $k \geq H - 1$). The feature-mode and history-length sweeps further indicate that residual accuracy depends strongly on the chosen feature mode and the tested history lengths $H \in \{50, 100, 150\}$ yield comparable residual RMSE distributions, favouring shorter windows when latency and warm-up constraints matter.

RQ3 (Benchmark comparison to the dataset baseline). Section 5.3 benchmarks the proposed pipeline under the same 50,000/5,000 train/test protocol as the IEEE DataPort baseline and reports per-joint motor-current RMSE on three benchmark-capable datasets. On UR3e without load, the linear identification baseline attains the lowest RMSE, whereas on UR3e with load and on UR10e without load the DeLaN+LSTM model achieves lower RMSE than the baseline on most or all joints, as summarised by the relative-gain and overall-RMSE comparisons.

The K-domination study quantifies how the number of available demonstration trajectories influences the two-stage pipeline. Across all reported metrics, increasing K reduces both the typical error level and the variability across dataset seeds, indicating improved generalisation and robustness with more diverse motion coverage.

The DeLaN learning curves (Figures 6 and 7) summarise the Stage 1 optimisation as a function of K . Small trajectory sets lead to substantially higher training loss and validation error, together with markedly larger IQR. This behaviour is consistent with the fact that the dataset seed determines which trajectories are selected and how they are split into train/val/test at the trajectory level, which is comparable to evaluating different “folds” of the trajectory pool. At low K , some subsets can miss relevant dynamic regimes, resulting in splits that are harder to learn and therefore higher validation error. For $K \geq 32$ the curves collapse quickly, indicating reduced sensitivity to the particular split.

The progress-aligned motor-current RMSE (Figure 8) highlights that the smallest setting ($K = 8$) exhibits pronounced error spikes and large variability along the trajectory, while increasing K stabilises the error over the full progress range. Moreover, the per-joint analysis (Figure 9) shows that for small K individual joints can dominate the overall error, whereas larger K yields consistently low and more uniform per-joint errors. This aligns with the per-joint normalisation used in the Stage 1 loss, but also emphasises that sufficient trajectory diversity is required for this normalisation to translate into uniform generalisation across all joints.

The residual LSTM results (Figures 10, 11, and 13) indicate that larger K yields lower validation loss and reduced variability. Since Stage 2 is trained on residuals generated by the frozen Stage 1 model, its achievable performance is inherently coupled to the quality and coverage of the DeLaN residuals. Consequently, low K can manifest as larger residual peaks and higher seed sensitivity, whereas larger K provides a more stable residual learning problem.

The representative overlays (Figures 25 and 24) illustrate that the residual learner compensates systematic deviations of the DeLaN prediction and thereby reduces the final motor-current error. Taken together, the K-domination results support the interpretation that a minimum trajectory count is required before the two-stage pipeline becomes robust to the particular trajectory selection and split, with diminishing returns as K approaches the full dataset.

The best-model DeLaN sweep (Figures 15–18) starts from a “Lutter-like” baseline configuration and varies one axis at a time (capacity, depth, weight decay, or learning rate) to probe whether the remaining performance variability is driven by under-capacity, overfitting, or optimisation stability. The learning curves (Figure 15) show that all presets converge rapidly and then plateau, indicating that differences between presets are expressed primarily in the final validation regime rather than in early transient behaviour. The accuracy–stability scatter (Figure 16) reveals a trade-off between typical validation error and seed stability: the smaller-capacity variant attains the tightest within-split dispersion but at a higher median validation RMSE, consistent with capacity-limited fitting, whereas the deeper variant shifts toward lower validation RMSE while maintaining a comparable IQR. In contrast, both stronger weight decay and a lower learning rate move the operating point toward larger validation RMSE and higher dispersion, suggesting that these changes do not act as stabilisers in this setting. The validation–test scatter (Figure 17) indicates that the relative ranking on validation RMSE is broadly preserved on the test subset, supporting the use of validation-driven checkpoint selection in Algorithm 2. Finally, the per-joint breakdown (Figure 18) shows that the spread across presets is not uniform across joints: joint 1 consistently exhibits the largest errors and variability, and the differences between presets are most pronounced on the higher-error joints, which aligns with the design goal of selecting a preset that is both accurate and robust under the dominant seed variability.

The residual-model sweep (Figures 19–23) evaluates whether residual motor-current dynamics are best explained from kinematics alone, from the DeLaN baseline prediction alone, or from their combination (Section 3). Across configurations, the best-validation-loss versus test-RMSE scatter (Figure 19) separates a dominant cluster of low-error settings from a smaller set of markedly worse runs, indicating that some feature choices lead to consistently inferior residual predictors. This separation is reflected in the feature-mode boxplots: residual RMSE (Figure 21) and the overfit indicator (Figure 20) both show that the feature mode relying only on the DeLaN baseline prediction (“`tau_hat`” in the plot labels, i.e., the DeLaN-predicted motor current) exhibits substantially larger errors and greater train–validation discrepancy than the other modes. By contrast, the hybrid feature mode combining velocity/acceleration with the DeLaN baseline (“`state_tauhat`”) remains in the low-error regime and shows comparatively tight dispersion, consistent with the intended role of the residual model as a correction conditioned on both state and baseline prediction. At the same time, Figure 20 shows that even for `state_tauhat` the final validation-to-training loss ratio remains clearly above one, indicating a non-negligible generalisation gap: the residual predictor fits the training windows more strongly than the held-out validation windows. Together with the low and tightly clustered residual RMSE of `state_tauhat` in Figure 21, this explains why `state_tauhat` is consistently favoured by the robustness-aware selection in Algorithm 3, but it also highlights that its apparent dominance is established under the in-distribution trajectory split used here. Assessing whether this feature mode retains its advantage when extrapolating to more strongly shifted test conditions (like unseen tasks, payloads, or robots) remains an important direction for future work. Within the same comparison, the `state` and `full` modes yield similar residual RMSE levels but higher loss ratios than `state_tauhat`, suggesting that including absolute joint position (and, in `full`, the DeLaN baseline) does not provide a clear robustness benefit

in this setting. Conversely, the `tau_hat`-only mode performs worst and exhibits the largest train-validation discrepancy, indicating that the DeLaN baseline prediction alone is insufficient to recover the history-dependent residual dynamics. Comparing history lengths (Figure 22), the residual RMSE distributions for $H \in \{50, 100, 150\}$ are close in median and spread, suggesting that most predictive information is contained in relatively short histories and that increasing H does not systematically improve the residual fit. Interpreting H in physical time, the dataset is sampled at 100 Hz, such that the smallest window length $H = 50$ corresponds to 0.5 s of history. Since the three boxplots largely overlap, using the shortest window is preferable in practice: it reduces the required warm-up horizon ($k \geq H - 1$) and the amount of past context needed for the residual predictor without sacrificing accuracy. At the same time, a 0.5 s history can still be substantial for fast, safety-critical interaction and tool/payload compensation, motivating future work on shorter-horizon residual models (for example faster sampling, more compact history representations, or training on shorter windows) while maintaining sufficient coverage of relevant motion regimes. The residual overlay (Figure 23) further illustrates that the LSTM captures the dominant temporal structure of the residual across joints, while remaining errors concentrate around sharper transitions where the residual changes rapidly.

Section 5.3 compares the proposed pipeline to the model-based identification baseline reported with the IEEE DataPort dataset [21]. The baseline fits a joint-wise, linear-in-parameters current model from kinematics, whereas the DeLaN and DeLaN+LSTM models implement physics-inspired representation learning (Stage 1) and a learned residual correction (Stage 2). Table 2 highlights that the relative performance depends strongly on the benchmark condition: for UR3e without load, the baseline attains the lowest per-joint RMSE, while the DeLaN+LSTM pipeline yields higher errors across all joints, indicating that the linear current regressor is already highly effective in this regime. For UR3e with load, the ranking reverses and the DeLaN+LSTM pipeline achieves substantially lower RMSE than both the baseline and the DeLaN-only model on all joints, suggesting that the hybrid physics-informed plus residual formulation better captures the load-conditioned current dynamics in this dataset. For UR10e without load, DeLaN-only produces markedly larger errors, while the DeLaN+LSTM pipeline reduces RMSE relative to DeLaN and improves upon the baseline on most joints, indicating that the residual model compensates systematic errors that remain after the physics-constrained Stage 1 fit.

Figure 26 summarises these trends as a relative change with respect to the baseline. The UR3 no-load panel shows positive relative changes on all joints, meaning that the two-stage pipeline does not outperform the baseline in that condition. By contrast, the UR3 with-load panel shows negative relative changes on all joints, indicating consistent improvement of DeLaN+LSTM over the baseline under load. For UR10 no load, the relative change remains negative across joints but with smaller magnitude, which is consistent with the more modest gains observed in Table 2.

The overall-RMSE summaries in Figures 27 and 28 provide a compact view across conditions. On UR3, the baseline exhibits a higher overall RMSE for the with-load condition than for the no-load condition, while the DeLaN+LSTM pipeline shows the opposite trend in this benchmark. Across manipulators, both models exhibit higher overall RMSE on UR10 than on UR3 in the no-load setting, reflecting the increased error level on the larger robot. However,

the gap between UR3 and UR10 is smaller for the DeLaN+LSTM pipeline than for the baseline. Taken together, these results suggest that the physics-informed neural pipeline is most competitive in regimes where the linear current identification baseline exhibits larger errors (under load and on UR10), while the baseline remains difficult to beat on UR3 without load, where its parametric structure appears sufficient for the observed current dynamics.

8 Summary and Outlook

This thesis addressed the problem of learning accurate inverse-dynamics models for collaborative manipulators in a way that remains compatible with the requirements of online payload and interaction-force estimation. Rather than treating inverse dynamics as a purely black-box regression task, the work adopted a physics-inspired learning strategy that embeds mechanical structure into the model class and learns the remaining discrepancy from data. Since the IEEE DataPort dataset used for evaluation provides motor-current measurements (but no joint-torque ground truth), Chapters 4–7 consistently formulated objectives and metrics in the motor-current domain, noting that torque-domain quantities can be recovered up to a constant motor torque constant in the operating regime considered.

The proposed approach is a two-stage pipeline. Stage 1 uses Deep Lagrangian Networks (DeLaN) to learn a structured dynamics predictor from joint kinematics, yielding a baseline motor-current model that respects fundamental properties of rigid-body dynamics. Stage 2 augments this baseline with a residual Long Short-Term Memory (LSTM) sequence model trained on the remaining modelling error, enabling history-dependent corrections that capture effects not explained by the instantaneous state alone. To make both stages reproducible and comparable across dataset splits, the thesis introduced a trajectory-level evaluation protocol, an explicit best-model selection procedure that accounts for accuracy and seed stability, and an implementation that runs end-to-end within a containerised stack.

Experimentally, the K-domination study quantified how the available number of trajectories affects optimisation stability and generalisation for both stages. The results showed that increasing trajectory coverage reduces both the typical error level and the variability across dataset seeds, and that the residual learning problem in Stage 2 is tightly coupled to the quality of the Stage 1 baseline. The best-model approach further demonstrated how targeted hyperparameter and feature-mode ablations translate into measurable accuracy–robustness trade-offs. Finally, the benchmark comparison to the dataset baseline highlighted that the relative performance depends on the operating condition: the proposed physics-informed pipeline is most competitive in regimes where the linear identification baseline exhibits larger errors (e.g., under load and on the larger robot), while the baseline remains difficult to beat in the unloaded UR3 condition.

The results in this thesis motivate several directions for future work. First, extending the current-domain evaluation to a torque-domain pipeline would allow direct integration with es-

tablished payload identification methods. This requires converting motor currents to motor torques and, where needed, mapping joint torques to task-space quantities via the Jacobian. Second, the residual model is currently purely data-driven. Adding explicit, interpretable non-conservative terms (e.g., friction) to the structured dynamics model and using the sequence model only for the remaining error could improve transfer across payloads, trajectories, and robots. Third, the benchmark results are condition-dependent. Training across multiple datasets and robots, together with domain adaptation and uncertainty-aware model selection, could improve robustness when moving between load conditions and platforms. Finally, for on-line use, future work should study incremental adaptation and real-time computational budgets, and evaluate how better inverse-dynamics prediction improves payload parameter identification during manipulation tasks.

Bibliography

- [1] Stanford Institute for Human-Centered Artificial Intelligence (HAI). “Artificial intelligence index report 2025,” Accessed: Nov. 22, 2025. [Online]. Available: <https://hai.stanford.edu/ai-index/2025-ai-index-report>
- [2] International Federation of Robotics. “World robotics 2025 report.” Press release, Sept. 25, 2025, Accessed: Nov. 22, 2025. [Online]. Available: <https://ifr.org/ifr-press-releases/news/global-robot-demand-in-factories-doubles-over-10-years>
- [3] International Federation of Robotics. “Collaborative robots – how robots work alongside humans.” Bar chart illustration, Accessed: Nov. 22, 2025. [Online]. Available: <https://ifr.org/ifr-press-releases/news/how-robots-work-alongside-humans>
- [4] P. Nadeau, M. Giamou, and J. Kelly, “Fast object inertial parameter identification for collaborative robots,” in 2022 International Conference on Robotics and Automation (ICRA), 2022, pp. 3560–3566. DOI: [10.1109/ICRA46639.2022.9916213](https://doi.org/10.1109/ICRA46639.2022.9916213)
- [5] A. Kerdas, M. Hamad, J. Vorndamme, N. Mansfeld, S. Abdolshah, and S. Hadadin, “Online payload identification for tactile robots using the momentum observer,” in 2022 International Conference on Robotics and Automation (ICRA), 2022, pp. 5953–5959. DOI: [10.1109/ICRA46639.2022.9811691](https://doi.org/10.1109/ICRA46639.2022.9811691)
- [6] S. Zhang, M. Yuan, Z. Huo, J. Huang, and X. Zhang, “Accurate payload dynamics estimation and compensation of a robotic manipulator without external motion measuring sensors,” in 2025 11th International Conference on Electrical Engineering, Control and Robotics (EECR), 2025, pp. 1–8. DOI: [10.1109/EECR64516.2025.11077346](https://doi.org/10.1109/EECR64516.2025.11077346)
- [7] S. K. Kommuri, S. Han, and S. Lee, “External torque estimation using higher order sliding-mode observer for robot manipulators,” IEEE/ASME Transactions on Mechatronics, vol. 27, no. 1, pp. 513–523, 2022. DOI: [10.1109/TMECH.2021.3067443](https://doi.org/10.1109/TMECH.2021.3067443)
- [8] S. Long, X. Dang, S. Sun, Y. Wang, and M. Gui, “A novel sliding mode momentum observer for collaborative robot collision detection,” Machines, vol. 10, no. 9, p. 818, 2022. DOI: [10.3390/machines10090818](https://doi.org/10.3390/machines10090818) [Online]. Available: <https://www.mdpi.com/2075-1702/10/9/818>
- [9] X. Wei et al., “Composite disturbance filtering for interaction force estimation with online environmental stiffness exploration,” IEEE/ASME Transactions on Mechatronics, vol. 30, no. 1, pp. xxx–xxx, 2025. DOI: [10.1109/TMECH.2024.3443310](https://doi.org/10.1109/TMECH.2024.3443310)

- [10] Z. Lao, Y. Han, Y. Ma, and G. S. Chirikjian, “A learning-based approach for estimating inertial properties of unknown objects from encoder discrepancies,” *IEEE Robotics and Automation Letters*, vol. 8, no. 9, pp. 5283–5290, 2023. DOI: [10.1109/LRA.2023.3293723](https://doi.org/10.1109/LRA.2023.3293723)
- [11] M. Liu et al., “A two-stage payload dynamic parameter identification method for interactive industrial robots with large components,” *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 13871–13883, 2025. DOI: [10.1109/TASE.2025.3557064](https://doi.org/10.1109/TASE.2025.3557064)
- [12] T. Xu et al., “Identifying current dynamics of robot payload based on iterative weighting estimation,” *IEEE Transactions on Instrumentation and Measurement*, vol. 74, pp. 1–14, 2025. DOI: [10.1109/TIM.2025.3554883](https://doi.org/10.1109/TIM.2025.3554883)
- [13] T. Xu, J. Fan, Q. Fang, Y. Zhu, and J. Zhao, “An accurate identification method based on double weighting for inertial parameters of robot payloads,” *Robotica*, vol. 40, pp. 1–17, 2022. DOI: [10.1017/S0263574722000960](https://doi.org/10.1017/S0263574722000960) [Online]. Available: <https://doi.org/10.1017/S0263574722000960>
- [14] J. Duan, Z. Liu, Y. Bin, K. Cui, and Z. Dai, “Payload identification and gravity/inertial compensation for six-dimensional force/torque sensor with a fast and robust trajectory design approach,” *Sensors*, vol. 22, no. 2, 2022, ISSN: 1424-8220. DOI: [10.3390/s22020439](https://doi.org/10.3390/s22020439) [Online]. Available: <https://www.mdpi.com/1424-8220/22/2/439>
- [15] J. Swevers, W. Verdonck, and J. De Schutter, “Dynamic model identification for industrial robots,” *IEEE Control Systems Magazine*, vol. 27, no. 5, pp. 58–71, 2007. DOI: [10.1109/MCS.2007.904659](https://doi.org/10.1109/MCS.2007.904659)
- [16] M. Lutter and J. Peters, *Combining physics and deep learning to learn continuous-time dynamics models*, 2023. arXiv: [2110.01894](https://arxiv.org/abs/2110.01894) [cs.LG]. [Online]. Available: <https://arxiv.org/abs/2110.01894>
- [17] Y. Hu, W. Li, Y. Zhou, and D. T. Pham, “Improved deep lagrangian network-enabled momentum observer for collision detection during human-robot collaboration,” *Robotics and Computer-Integrated Manufacturing*, vol. 97, p. 103093, 2026, ISSN: 0736-5845. DOI: <https://doi.org/10.1016/j.rcim.2025.103093> [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0736584525001474>
- [18] S. Wu, F. Sun, W. Chen, and Y. Li, “Extended deep lagrangian network for robotic arm dynamics considering motor couplings,” in *2025 40th Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, 2025, pp. 2050–2054. DOI: [10.1109/YAC66630.2025.11150193](https://doi.org/10.1109/YAC66630.2025.11150193)
- [19] Tao, Chen, Liu, Wan, Wei, and Wang, “Robot hybrid inverse dynamics model compensation method based on the bll residual prediction algorithm,” *Robotica*, vol. 43, no. 3, pp. 649–663, 2025. DOI: [10.1017/S0263574724002911](https://doi.org/10.1017/S0263574724002911)
- [20] S. Kružić, J. Musić, R. Kamnik, and V. Papić, “End-effector force and joint torque estimation of a 7-dof robotic manipulator using deep learning,” *Electronics*, vol. 10, no. 23, 2021, ISSN: 2079-9292. [Online]. Available: <https://www.mdpi.com/2079-9292/10/23/2963>

- [21] J. Heredia, C. Schlette, and M. B. Kjærgaard, “Data-driven energy estimation of individual instructions in user-defined robot programs for collaborative robots,” *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6836–6843, 2021. DOI: [10.1109/LRA.2021.3094781](https://doi.org/10.1109/LRA.2021.3094781)
- [22] F. Cao, P. D. Docherty, S. Ni, and X. Chen, “Contact force and torque sensing for serial manipulator based on an adaptive kalman filter with variable time period,” *Robotics and Computer-Integrated Manufacturing*, vol. 72, p. 102210, 2021, ISSN: 0736-5845. DOI: <https://doi.org/10.1016/j.rcim.2021.102210> [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0736584521000934>
- [23] J. Hu, Z. Chen, Y. Lin, Z. Chen, B. Yao, and X. Ma, “On the fully decoupled rigid-body dynamics identification of serial industrial robots,” *IEEE Transactions on Robotics*, vol. 41, pp. 4588–4605, 2025. DOI: [10.1109/TRO.2025.3578229](https://doi.org/10.1109/TRO.2025.3578229)
- [24] L. Han, J. Mao, P. Cao, Y. Gan, and S. Li, “Toward sensorless interaction force estimation for industrial robots using high-order finite-time observers,” *IEEE Transactions on Industrial Electronics*, vol. 69, no. 7, pp. 7275–7284, 2022. DOI: [10.1109/TIE.2021.3095820](https://doi.org/10.1109/TIE.2021.3095820)
- [25] M. Tang, Y. Yan, B. An, W. Wang, and Y. Zhang, “Dynamic parameter identification of collaborative robot based on wls-rwpso algorithm,” *Machines*, vol. 11, no. 2, p. 316, 2023. DOI: [10.3390/machines11020316](https://doi.org/10.3390/machines11020316) [Online]. Available: <https://www.mdpi.com/2075-1702/11/2/316>
- [26] T. Xu et al., “An online payload identification method based on parameter difference for industrial robots,” *Robotica*, vol. 42, pp. 1–23, 2024. DOI: [10.1017/S026357472400105X](https://doi.org/10.1017/S026357472400105X) [Online]. Available: <https://doi.org/10.1017/S026357472400105X>
- [27] S. Liu, L. Wang, and X. V. Wang, “Sensorless force estimation for industrial robots using disturbance observer and neural learning of friction approximation,” *Robotics and Computer-Integrated Manufacturing*, vol. 71, p. 102168, 2021, ISSN: 0736-5845. DOI: [10.1016/j.rcim.2021.102168](https://doi.org/10.1016/j.rcim.2021.102168) [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0736584521000521>
- [28] Y. Wei, W. Li, Y. Yang, X. Yu, and L. Guo, “Decoupling observer for contact force estimation of robot manipulators based on enhanced gaussian process model;” in *2022 IEEE 8th International Conference on Cloud Computing and Intelligent Systems (CCIS)*, 2022, pp. 1–7. DOI: [10.1109/CCIS57298.2022.10016359](https://doi.org/10.1109/CCIS57298.2022.10016359)
- [29] K. Fathi, M. Rezayati, and H. W. Van de Venn, “Human-robot contact detection in assembly tasks,” in *2022 7th International Conference on Mechanical Engineering and Robotics Research (ICMERR)*, 2022, pp. 224–230. DOI: [10.1109/ICMERR56497.2022.10097827](https://doi.org/10.1109/ICMERR56497.2022.10097827)
- [30] Y. Wei, S. Lyu, W. Li, X. Yu, Z. Wang, and L. Guo, “Contact force estimation of robot manipulators with imperfect dynamic model: On gaussian process adaptive disturbance kalman filter,” *IEEE Transactions on Automation Science and Engineering*, vol. 21, no. 3, pp. 3524–3537, 2024. DOI: [10.1109/TASE.2023.3280750](https://doi.org/10.1109/TASE.2023.3280750)

- [31] G. Giacomuzzo, N. Turcato, A. D. Libera, and R. Carli, "Embedding the physics in black-box inverse dynamics identification: A comparison between gaussian processes and neural networks," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 1584–1590, 2023, 22nd IFAC World Congress, ISSN: 2405-8963. DOI: <https://doi.org/10.1016/j.ifacol.2023.10.1858> [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S240589632302267X>
- [32] M. Pan et al., "An adaptive sparse general regression neural network-based force observer for teleoperation system," *Engineering Applications of Artificial Intelligence*, vol. 118, p. 105689, 2023, ISSN: 0952-1976. DOI: [10.1016/j.engappai.2022.105689](https://doi.org/10.1016/j.engappai.2022.105689) [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0952197622006790>
- [33] J. Liang and O. Kroemer, "Contact localization for robot arms in motion without torque sensing," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 6322–6328. DOI: [10.1109/ICRA48506.2021.9562058](https://doi.org/10.1109/ICRA48506.2021.9562058)
- [34] W. Taie, K. ElGeneidy, A. Al-Yacoub, and R. Sun, "Payload parameters identification using incremental ensemble learning," in *2024 4th International Conference on Computer, Control and Robotics (ICCCR)*, 2024, pp. 241–245. DOI: [10.1109/ICCCR61138.2024.10585532](https://doi.org/10.1109/ICCCR61138.2024.10585532)
- [35] W. Taie, K. ElGeneidy, A. Al-Yacoub, and R. Sun, "Online identification of payload inertial parameters using ensemble learning for collaborative robots," *IEEE Robotics and Automation Letters*, vol. 9, no. 2, pp. 1350–1356, 2024. DOI: [10.1109/LRA.2023.3346268](https://doi.org/10.1109/LRA.2023.3346268)
- [36] W. Taie, K. ElGeneidy, A. Al-Yacoub, and R. Sun, "Addressing catastrophic forgetting in payload parameter identification using incremental ensemble learning," *Frontiers in Robotics and AI*, vol. 11, p. 1470163, 2024, ISSN: 2296-9144. DOI: [10.3389/frobt.2024.1470163](https://doi.org/10.3389/frobt.2024.1470163) [Online]. Available: <https://www.frontiersin.org/articles/10.3389/frobt.2024.1470163/full>
- [37] S. Yang, J. Hu, S. Liu, W. Chen, and Y.-H. Liu, "A residual-driven decomposed pinns method for dynamics identification of robot manipulators," in *2025 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, 2025, pp. 660–665. DOI: [10.1109/RCAR65431.2025.11139811](https://doi.org/10.1109/RCAR65431.2025.11139811)
- [38] H. Hu, Z. Shen, and C. Zhuang, "A pinn-based friction-inclusive dynamics modeling method for industrial robots," *IEEE Transactions on Industrial Electronics*, vol. 72, no. 5, pp. 5136–5144, 2025. DOI: [10.1109/TIE.2024.3476977](https://doi.org/10.1109/TIE.2024.3476977)
- [39] X. Yang, Y. Du, L. Li, Z. Zhou, and X. Zhang, "Physics-informed neural network for model prediction and dynamics parameter identification of collaborative robot joints," *IEEE Robotics and Automation Letters*, vol. 8, no. 12, pp. 8462–8469, 2023. DOI: [10.1109/LRA.2023.3329620](https://doi.org/10.1109/LRA.2023.3329620)
- [40] A. Raviola, R. Guida, A. De Martin, S. Pastorelli, S. Mauro, and M. Sorli, "Effects of temperature and mounting configuration on the dynamic parameters identification of industrial robots," *Robotics*, vol. 10, p. 83, Jun. 2021. DOI: [10.3390/robotics10030083](https://doi.org/10.3390/robotics10030083)

- [41] J. Heredia, C. Schlette, and M. B. Kjærgaard. “Dataset of collaborative robots for energy consumption modeling,” Accessed: Jan. 27, 2026. [Online]. Available: <https://dx.doi.org/10.21227/9wnt-8v86>
- [42] GeeksforGeeks, [Long short-term memory \(lstm\) rnn in tensorflow](#), 2025. Accessed: Jan. 31, 2026. [Online]. Available: <https://www.geeksforgeeks.org/deep-learning/long-short-term-memory-lstm-rnn-in-tensorflow/>
- [43] Docker, Inc., [Docker: Accelerated container application development](#), 2026. Accessed: Jan. 31, 2026. [Online]. Available: <https://www.docker.com/>
- [44] Michael Lutter, [Deep_lagrangian_networks: Open-source implementation of deep lagrangian networks \(delan\)](#), 2025. Accessed: Feb. 11, 2025. [Online]. Available: https://github.com/milutter/deep_lagrangian_networks
- [45] Python Software Foundation, [Welcome to python.org](#), 2026. Accessed: Jan. 31, 2026. [Online]. Available: <https://www.python.org/>
- [46] DeepMind, [Optax documentation](#), 2025. Accessed: Feb. 11, 2025. [Online]. Available: <https://optax.readthedocs.io/en/latest/>
- [47] DeepMind, [Haiku documentation](#), 2025. Accessed: Feb. 11, 2025. [Online]. Available: <https://dm-haiku.readthedocs.io/en/latest/>
- [48] The Matplotlib Development Team, [Matplotlib: Visualization with python](#), 2026. Accessed: Jan. 31, 2026. [Online]. Available: <https://matplotlib.org/>
- [49] NumPy Developers, [Numpy](#), 2025. Accessed: Feb. 11, 2025. [Online]. Available: <https://numpy.org/>

List of Figures

Figure 1	Global annual installations of collaborative robots from 2017 to 2023 (in thousand units). Figure modified from [3].	1
Figure 2	Query logic	6
Figure 3	Overview of modelling paradigms	7
Figure 4	Concept graph of the SoA literature	14

Figure 5	Two-stage learning pipeline for the nominal robot-gripper dynamics. Stage 1 learns a structured inverse-dynamics model (DeLaN) in joint space from encoder and motor-current data and is trained by regressing motor torques. In Stage 2, a recurrent sequence model (LSTM) takes joint-state histories and DeLaN torque predictions as input and learns residual joint torques over a fixed history window. The combined joint-space model is then mapped through the Jacobian to obtain the nominal flange wrench in the force/torque sensor frame, which is compared against the measured wrench for evaluation.	20
Figure 6	DeLaN training loss by K shown as median \pm IQR across dataset seeds (with seed-wise aggregation across DeLaN initialisations).	34
Figure 7	DeLaN validation MSE (motor current) by K shown as median \pm IQR across dataset seeds (with seed-wise aggregation across DeLaN initialisations).	35
Figure 8	DeLaN motor-current RMSE over normalised progress ($0 \rightarrow 1$) by K shown as median \pm IQR (shaded bands).	35
Figure 9	DeLaN motor-current RMSE per joint (median \pm IQR) for all evaluated values of K (bars) with IQR error bars.	36
Figure 10	LSTM training loss by K shown as median \pm IQR across dataset seeds ($H = 100$, feature mode full).	36
Figure 11	LSTM validation loss by K shown as median \pm IQR across dataset seeds ($H = 100$, feature mode full).	37
Figure 12	LSTM residual motor-current RMSE per joint (median \pm IQR) for selected values of K ($H = 100$, feature mode full).	37
Figure 13	LSTM residual motor-current RMSE over normalised progress ($0 \rightarrow 1$) by K shown as median \pm IQR (shaded bands) ($H = 100$, feature mode full).	38
Figure 14	Representative per-joint motor-current RMSE on the test split comparing DeLaN and the combined DeLaN+LSTM predictor (valid indices $k \geq H - 1$, $H = 100$).	38
Figure 15	DeLaN training loss (left) and validation MSE in motor-current units (right) shown as median \pm IQR across seeds for each hyperparameter preset.	39
Figure 16	Hyperparameter comparison: median validation motor-current RMSE versus seed-stability measured by the median within-split IQR of validation RMSE (each point denotes one DeLaN hyperparameter preset. Statistics aggregated across dataset seeds with seed-wise aggregation across DeLaN initialisations).	40
Figure 17	Hyperparameter comparison: median validation motor-current RMSE versus median test motor-current RMSE (each point denotes one DeLaN hyperparameter preset. Statistics aggregated across dataset seeds with seed-wise aggregation across DeLaN initialisations).	41
Figure 18	DeLaN motor-current RMSE per joint (median \pm IQR) grouped by hyperparameter preset.	41
Figure 19	Best validation loss versus total motor-current RMSE on the test split for the LSTM residual model (marker colour denotes feature mode, marker outline denotes history length H).	42

Figure 20 Overfit indicator (final validation loss / final training loss) by feature mode shown as boxplots across runs (green triangles indicate means, panels use different y -axis ranges for readability).	43
Figure 21 LSTM residual motor-current RMSE on the test split by feature mode shown as boxplots across runs (green triangles indicate means, panels use different y -axis ranges for readability).	44
Figure 22 LSTM residual motor-current RMSE on the test split shown as boxplots across history lengths H (green triangles indicate means).	45
Figure 23 Representative residual motor-current traces: ground-truth residual versus LSTM residual prediction on the test split (valid indices $k \geq H - 1$, $H = 150$).	46
Figure 24 Best pipeline motor-current RMSE per joint on the test split comparing DeLaN and the combined DeLaN+LSTM predictor (valid indices $k \geq H - 1$, $H = 150$).	47
Figure 25 Representative motor-current overlay: ground truth, DeLaN prediction, and combined DeLaN+LSTM prediction on the test split (valid indices $k \geq H - 1$, $H = 150$).	48
Figure 26 Relative change in per-joint motor-current RMSE of DeLaN+LSTM versus the baseline model [21], reported as $(\text{RMSE}_{\text{ours}} - \text{RMSE}_{\text{baseline}})/\text{RMSE}_{\text{baseline}}$ for UR3 (no load), UR3 (with load), and UR10 (no load). Negative values indicate a reduction in RMSE, positive values an increase.	50
Figure 27 Load sensitivity on UR3: overall motor-current RMSE for the baseline model [21] and DeLaN+LSTM, comparing the no-load and with-load conditions.	51
Figure 28 Kinematic difference (UR3 versus UR10, no load): overall motor-current RMSE for the baseline model [21] and DeLaN+LSTM on the no-load test split for both manipulators.	52

List of Tables

Table 1 Overview of query results by category.	6
Table 2 Baseline [21] vs. DeLaN vs. DeLaN+LSTM motor-current RMSE per joint on the benchmark test split (trained on 50,000 samples, evaluated on 5,000 unseen samples. DeLaN+LSTM evaluated on valid indices $k \geq H - 1$).	50
Table 3 Overview of selected references by category, source and citation count. (accessed 2025-11-30T18:53:00 [YYYY-MM-DDTHH:mm:ss])	76

List of source codes

A Kinematic and Dynamic Background of Robot Manipulation and Environment Interaction

The external wrench \vec{F}_{ext} in (5) is a 6-dimensional vector expressed in the sensor/tool frame S ,

$$\vec{F}_{\text{ext}} = \begin{bmatrix} \mathbf{f} \\ \boldsymbol{\tau} \end{bmatrix} \in \mathbb{R}^6, \quad (57)$$

with $\mathbf{f} \in \mathbb{R}^3$ the linear force and $\boldsymbol{\tau} \in \mathbb{R}^3$ the moment about the frame origin. For a rigid body with parameters ϕ_{eff} (mass, CoM and inertia) moving with linear and angular motion $(\mathbf{a}, \boldsymbol{\alpha}, \boldsymbol{\omega})$, the Newton-Euler equations (cf. (2)) give

$$\begin{bmatrix} \mathbf{f} \\ \boldsymbol{\tau} \end{bmatrix} = m \begin{bmatrix} \mathbf{I} & -[\mathbf{c}]^\times \\ [\mathbf{c}]^\times & \mathbf{J}_s \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \boldsymbol{\alpha} \end{bmatrix} + \begin{bmatrix} m[\boldsymbol{\omega}]^\times [\boldsymbol{\omega}]^\times \mathbf{c} \\ [\boldsymbol{\omega}]^\times \mathbf{J}_s \boldsymbol{\omega} \end{bmatrix}. \quad (58)$$

The translational part \mathbf{f} can be written as

$$\mathbf{f} = m\mathbf{a} - m[\mathbf{c}]^\times \boldsymbol{\alpha} + m[\boldsymbol{\omega}]^\times [\boldsymbol{\omega}]^\times \mathbf{c}, \quad (59)$$

where the first term $m\mathbf{a}$ is the familiar inertial force, while $-m[\mathbf{c}]^\times \boldsymbol{\alpha}$ and $m[\boldsymbol{\omega}]^\times [\boldsymbol{\omega}]^\times \mathbf{c}$ collect the additional centripetal and Coriolis contributions induced by the angular motion $\boldsymbol{\omega}$ and the CoM offset \mathbf{c} . Similarly, the rotational part $\boldsymbol{\tau}$ can be written as

$$\boldsymbol{\tau} = m[\mathbf{c}]^\times \mathbf{a} + \mathbf{J}_s \boldsymbol{\alpha} + [\boldsymbol{\omega}]^\times \mathbf{J}_s \boldsymbol{\omega}, \quad (60)$$

where $\mathbf{J}_s \boldsymbol{\alpha}$ is the inertial moment due to angular acceleration, $m[\mathbf{c}]^\times \mathbf{a}$ is the torque induced by the translational acceleration of the offset CoM, and $[\boldsymbol{\omega}]^\times \mathbf{J}_s \boldsymbol{\omega}$ represents gyroscopic effects associated with the angular velocity $\boldsymbol{\omega}$.

In compact form, for a given motion $(\mathbf{a}, \boldsymbol{\alpha}, \boldsymbol{\omega})$ this can be written as

$$\vec{F}_{\text{ext}} = \vec{F}_{\text{dyn}}(\mathbf{a}, \boldsymbol{\alpha}, \boldsymbol{\omega}; \phi_{\text{eff}}) = Y(\mathbf{a}, \boldsymbol{\alpha}, \boldsymbol{\omega}) \phi_{\text{eff}}, \quad (61)$$

where $Y(\cdot)$ is a 6×10 regressor matrix that is linear in ϕ_{eff} but nonlinear in the motion variables. Hence, \vec{F}_{ext} is not simply $m\mathbf{a}$, nor can it be written as $\phi_{\text{eff}} \ddot{\mathbf{q}}$; the mapping from joint accelerations $\ddot{\mathbf{q}}$ to \vec{F}_{ext} passes through the robot kinematics and the Newton-Euler relations.

Once the wrench at the flange is known, the corresponding joint torques are obtained via

$$\boldsymbol{\tau}_{\text{ext}} = {}^S J(\mathbf{q})^\top \vec{F}_{\text{ext}}, \quad (62)$$

where ${}^S J(\mathbf{q})$ is the Jacobian of the sensor/tool frame S . Combining the relations above yields the identification-friendly form

$$\boldsymbol{\tau}_{\text{ext}} = \mathbf{J}^T(\mathbf{q}) Y(\mathbf{a}, \boldsymbol{\alpha}, \boldsymbol{\omega}) \phi_{\text{eff}}, \quad (63)$$

which makes explicit that $\boldsymbol{\tau}_{\text{ext}}$ is linear in ϕ_{eff} , but nonlinear in $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}$ through the dependence on $\mathbf{a}, \boldsymbol{\alpha}, \boldsymbol{\omega}$.

B Query Categories-Index Terms

This appendix lists the index terms used to construct the query categories illustrated in Fig. 2. The original nine term groups were consolidated into five content clusters C_1, \dots, C_5 and the goal/context term sets C_{mt} and C_{ct} .

Content Clusters C_i

C_1 : Classical / Observers

- momentum observer (MO)
- generalized momentum observer (GMO)
- disturbance observer (DOB)
- reaction force observer (RFOB)
- Kalman filter (KF)
- extended Kalman filter (EKF)
- unscented Kalman filter (UKF)
- state observer
- least squares (LS)
- weighted least squares (WLS)
- iterative reweighted least squares (IRLS)
- recursive least squares (RLS)
- momentum-based observer
- dynamic state observer
- observer
- force observer
- torque observer

C₂: Gaussian Process (GP)

- gaussian process regression (GPR)
- sparse gaussian process (SGP, SGPR)
- multi-output gaussian process (MOGP)
- multi-task gaussian process (MTGP)
- gaussian process state space model (GPSSM)
- hybrid gaussian process
- GP residual
- gaussian process dynamics
- GP inverse dynamics
- bayesian nonparametric regression (BNPR)

C₃: Deep Sequence Models (MLP / GRU / TCN / Transformer / LSTM)

- neural network inverse dynamics (NN-ID)
- deep learning
- multi layer perceptron (MLP)
- residual network (ResNet)
- long short-term memory (LSTM)
- gated recurrent unit (GRU)
- temporal convolutional network (TCN)
- causal convolution
- dilated convolution
- transformer model
- attention model
- sequence-to-sequence (seq2seq, S2S)
- sequence GAN (SeqGAN, TimeGAN)
- GAN
- Generative Adversarial Networks
- residual neural network (ResNN)

- residual GAN
- domain adaptation (DA)
- transfer learning (TL)
- meta learning (ML)
- context variable dynamics
- latent variable model (LVM)
- amortized inference (AI)
- test time adaptation (TTA)
- online adaptation (OA)
- feature invariance
- domain invariant features (DIF)
- few shot learning (FSL)
- zero shot transfer (ZSL)
- reinforcement
- reinforcement learning
- Isaac Gym differentiable
- Isaac Lab differentiable
- Isaac Gym
- Isaac Lab

C₄: Physics-Informed / Differentiable

- residual learning dynamics
- hybrid model dynamics
- analytical dynamics neural network (ADNN)
- physics residual
- rigid body dynamics residual (RBD residual)
- Newton Euler residual (NE residual)
- nominal dynamics model (NDM)
- neural correction

- learning inverse dynamics residual (ID residual)
- physics-informed neural network (PINN)
- differentiable physics
- differentiable simulation (DiffSim)
- differentiable robot model
- differentiable dynamics
- neural ODE (NODE)
- torchdiffeq
- ODE-net
- physics-guided machine learning robotics (PGML)

C_5 : Surveys

- survey
- benchmarking
- review
- overview
- systematic comparison

Goal & Domain Terms C_T

C_{mt} : Estimation & Modeling Terms

- external force
- force measurement
- force estimation
- force/torque estimation
- wrench estimation
- joint torque estimation
- end-effector force
- end-effector torque
- inertial parameters

- inertial parameter identification (IPI)
- online payload identification
- payload identification
- payload estimation
- object parameter estimation
- parameter identification
- inertia tensor
- inertia tensor estimation
- center of mass (CoM)
- rigid body dynamics
- friction approximation
- nonlinear friction model
- external perturbations
- force torque sensor (F/T sensor)
- external force estimation (EFE)
- external torque estimation (ETE)
- torque estimation
- parameter identification differentiable simulation
- payload identification (PI)
- payload estimation (PE)
- contact force
- nonlinear systems
- noise
- signal noise
- noise estimation

Note that the last four entries (nonlinear systems, noise, signal noise, noise estimation) are generic terms that occur across many physical systems beyond robotic manipulators. Including them in the queries significantly increased the number of retrieved results.

C_{ct} : Robotics Context Terms

- robotic manipulator
- robotic arm
- robotic manipulation
- robot payload

C Concept Graph

Table 3: Overview of selected references by category, source and citation count. (accessed 2025-11-30T18:53:00 [YYYY-MM-DDTHH:mm:ss])

Reference	Cite	Year	Database	Citations
R1	[4]	2022	IEEE	13
R2	[5]	2022	IEEE	16
R3	[7]	2022	IEEE	47
R4	[22]	2021	ScienceDirect	34
R5	[6]	2025	IEEE	0
R6	[23]	2025	IEEE	2
R7	[11]	2025	IEEE	1
R8	[12]	2025	IEEE	0
R9	[13]	2022	Cambridge	10
R10	[14]	2022	MPDI	27
R11	[9]	2025	IEEE	2
R12	[24]	2022	IEEE	57
R13	[15]	2007	IEEE	303
R14	[8]	2022	MPDI	17
R15	[25]	2023	MPDI	16
R16	[26]	2024	Cambridge	2
R17	[27]	2021	ScienceDirect	100
R18	[28]	2022	IEEE	3
R19	[30]	2024	IEEE	19
R20	[29]	2022	IEEE	2
R21	[31]	2023	ScienceDirect	2
R22	[19]	2025	Cambridge	1
R23	[10]	2023	IEEE	3
R24	[20]	2021	MPDI	10
R25	[32]	2023	ScienceDirect	9
R26	[33]	2021	IEEE	5
R27	[34]	2024	IEEE	1
R28	[35]	2024	IEEE	13
R29	[36]	2024	Frontiersin	0
R30	[18]	2025	IEEE	0
R31	[16]	2023	ArXiv	-
R32	[37]	2025	IEEE	0
R33	[38]	2024	IEEE	10