FH University of Applied Sciences TECHNIKUM WIEN

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

Supervisor: Michael Schebek, MSc

Vienna, January 31, 2026

Declaration

Vienna, January 31, 2026 Signature

Kurzfassung

Im Kontext der digitalen Fabrik an der UAS Technikum Wien, wo Menschen und Roboter sich die Aufgaben und den Arbeitsbereich teilen, ist die sichere und effiziente Handhabung von Nutzlasten von entscheidender Bedeutung. In der digitalen Fabrik der UAS werden Nutzlasten derzeit noch ohne Kenntnis ihrer internen Parameter gehandhabt, was zu potenziellen Manipulationsfehlern führen kann, die Menschen Schaden zufügen. Diese Studie beschreibt die Entwicklung einer fortschrittlichen Methode zur Kraft-/Drehmomentabschätzung, um die Fähigkeit eines UR5-Roboters zu verbessern, verschiedene Nutzlastbedingungen zu erkennen und zu handhaben. Diese Fähigkeit gewährleistet die Wahrnehmung des auf einer mobilen Industrieroboterplattform montierten UR5-Roboters, um den sicheren und effizienten Transfer von Nutzlasten zwischen verschiedenen Arbeitsbereichen innerhalb der Fabrik zu erleichtern. Die modernsten Methoden zur Kraft-/Drehmomentabschätzung für Industrieroboter nutzen neuronale Netze und Gauß-Prozesse als führende Methoden für genaue Nutzlastabschätzungen. Es wurde ein Gauß-Prozess-Modell entwickelt, um die Kräfte und Drehmomente abzuschätzen, die vom Roboter bei der Ausführung von Trajektorien erzeugt werden. In einem zukünftigen Projekt kann das Bewusstsein für Nutzlasten auf dem UR5-Roboter hinzugefügt werden. Auf diese Weise zielt die Studie darauf ab, die Intelligenz von Robotersystemen in industriellen Umgebungen zu verbessern und den Weg für eine höhere Produktivität und Sicherheit in digitalen Fertigungsumgebungen zu ebnen. Dieses Projekt führte auch zu einer Simulation, die eine Grundlage für die Aufzeichnung der Sensordaten aus dem UR5-Interieur.

Schlagworte: Gaussian Process, Force Estimation, Newton/Euler, UR5 Robot, Rigid Body

Abstract

In the context of the digital factory, at UAS Technikum Vienna, where humans and robots share the tasks and the workspace, the safe and efficient handling of payloads is essential. At the UAS digital factory payload is still handled without recognising anything about the payloads internal parameters, leading to potential manipulation failures causing human harm. This study describes the development of an advanced force/torque estimation method to improve a UR5 robots ability to recognize and handle different payload conditions. This capability ensures the perception of the UR5 robot mounted on a mobile industrial robot platform to facilitate the safe and efficient transfer of payloads between different workspaces within the factory. The state of the art methods of force/torque estimation for industrial robots serve neuronal networks and gaussian processes as the leading methods for accurate payload estimations. A gaussian process model has been developed to estimate the forces and torques generated by the robot when executing trajectories. In a future project face, an awareness of payloads can be added on the UR5 robot. In this way, the study aims to improve the intelligence of robotic systems in industrial environments and pave the way for higher productivity and safety in digital manufacturing environments. This project face also yeelted in a simulation that provides a basis to record the sensor data from the UR5's internal sensors and a force/torque sensor and a pipeline to train and evaluate gaussian process models.

Keywords: Gaussian Process, Force Estimation, Newton/Euler, UR5 Robot, Rigid Body

Acknowledgements

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Contents

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].

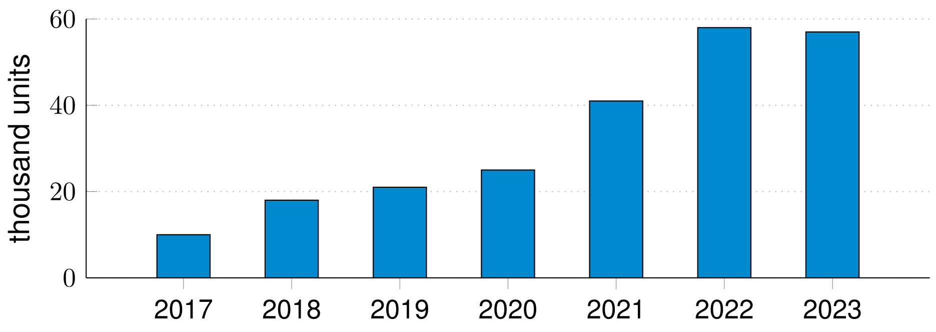

Figure 1: Global annual installations of collaborative robots from 2017 to 2023 (in thousand units). Figure modified from [3].

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

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

which enters the Newton-Euler equations

$$\begin{bmatrix} \mathbf{f} \\ \boldsymbol{\tau} \end{bmatrix} = m \begin{bmatrix} \mathbf{I}_{3\times3} & -[\mathbf{c}]^\times \\ [\mathbf{c}]^\times & \mathbf{J}_s \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \boldsymbol{\alpha} \end{bmatrix} + \begin{bmatrix} m[\boldsymbol{\omega}]^\times[\boldsymbol{\omega}]^\times \mathbf{c} \\ [\boldsymbol{\omega}]^\times \mathbf{J}_s \boldsymbol{\omega} \end{bmatrix}, \tag{2}$$

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

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

$$\boldsymbol{\phi}_{\text{eff}} = \begin{cases} \boldsymbol{\phi}_{\text{tool}}, & \text{no load,} \\ \boldsymbol{\phi}_{\text{tool}} + \boldsymbol{\phi}_{\text{load}}, & \text{with load,} \end{cases} \tag{3}$$

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 $\boldsymbol{\tau}_{\text{robot}}$ (clean flange),

$$\boldsymbol{\tau}_{\text{robot}} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) + \boldsymbol{\tau}_f(\dot{\mathbf{q}}), \tag{4}$$

where $\boldsymbol{\tau}_f(\dot{\mathbf{q}})$ 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

$$\boldsymbol{\tau}_{\text{ext}} = \mathbf{J}^T(\mathbf{q})\, \vec{F}_{\text{ext}}(\boldsymbol{\phi}_{\text{eff}}), \tag{5}$$

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

The motor torques are therefore

$$\boldsymbol{\tau}_{\text{motor}} = \boldsymbol{\tau}_{\text{robot}} + \boldsymbol{\tau}_{\text{ext}}(\boldsymbol{\phi}_{\text{eff}}), \tag{6}$$

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

$$\boldsymbol{\tau}_{\text{motor}} = k_t\, \boldsymbol{I} \quad \Rightarrow \quad \boldsymbol{I} = \frac{\boldsymbol{\tau}_{\text{robot}} + \boldsymbol{\tau}_{\text{ext}}(\boldsymbol{\phi}_{\text{eff}})}{k_t}. \tag{7}$$

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

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

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 $\vec{F}_{\text{bias}}$ and $\vec{n}$ denote sensor bias and noise, respectively.[1] 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

[1] All experiments in this work are simulation-based; in the subsequent method formulation, sensor bias and noise are therefore neglected and (8) is used without $\vec{F}_{\text{bias}}$ and $\vec{n}$.

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 indicates that the strongest results for robot inverse dynamics and friction modelling are achieved when physics-structured networks are combined with data-driven residual learners. Extended DeLaN models can capture motor couplings and current-torque relations from encoder and motor data alone [16, 17], while recent PINN-based approaches augment a structured dynamics model with temporal convolutions to obtain joint-torque prediction and friction compensation on industrial robots [18, 19]. At the same time, several works show that recurrent residual learners-in particular LSTMs-are highly effective for compensating modelling errors and estimating joint torques and end-effector forces in closely related settings [20, 21].

The aim of this thesis is to build on these insights and develop a physics-informed, sequence-model-based inverse-dynamics architecture that can be trained using only encoder and motor-current data, yields an accurate nominal model of the robot-gripper joint torques and can be consistently interpreted as a flange-wrench model in the force/torque sensor frame. Thereby providing a basis for tool/gripper compensation and subsequent payload dynamic parameter identification (PDPI) as formulated in Section **??**. Concretely, the work pursues the following objectives:

- **Stage 1 (structured inverse dynamics)** Stage 1 follows the improved DeLaN parameterisation of [22], using a Cholesky-factor inertia network and a learned potential, complemented by an explicit Coulomb-viscous joint-friction model [22, 23]. In contrast to the extended DeLaN-Motor architecture of [16], which embeds detailed motor couplings and is trained directly in current space, we adopt a simplified joint-side formulation and supervise the model using motor torques 6, obtained from encoder and motor-current data, where k_t is give [24]. The resulting DeLaN network is trained offline on carefully designed excitation trajectories to learn the nominal robot-gripper inverse dynamics.
- **Stage 2 (residual sequence model)** Stage 2 adopts the DeLaN + TCN paradigm of [19]: the structured inverse-dynamics model from Stage 1 provides a nominal torque prediction, and a separate sequence model (LSTM) learns residual joint torques from motion history.
- **Evaluation in joint space and measurement frame.** Evaluate the resulting DeLaN+LSTM architecture both in joint space and in the force/torque sensor frame, quantifying its ability to reproduce the nominal robot-gripper wrench across diverse trajectories.

In this work, a 6D force/torque sensor is used primarily as a research instrument to validate the joint-space modelling in the end-effector frame: flange-wrench measurements $\vec{F}_{\mathrm{meas}}$ are compared against the wrenches obtained by mapping the joint-space DeLaN+LSTM predictions through the Jacobian. Once consistency between predicted and measured flange wrenches has been demonstrated, the joint-space trained DeLaN+LSTM architecture provides a robust basis for PDPI by delivering a reliable tool/gripper wrench compensation in the measurement frame.

By combining a physics-informed inverse-dynamics backbone with a sequence model trained offline but executed online, this thesis aims to move from the calibration-heavy, fragmented SoA towards a unified and practically deployable notion of dynamic awareness: a cobot that can reliably predict its own joint torques, reproduce its nominal flange wrench in the measurement frame, and separate intrinsic robot-gripper dynamics from payload/collision-induced effects.

RQ1 How accurately can a DeLaN-based inverse-dynamics model, trained purely in joint space, predict the flange wrench in the measurement frame once mapped through the Jacobian? What extent does augmenting DeLaN with a residual sequence model improve this wrench prediction?

RQ2 How does augmenting a DeLaN with a residual sequence model (LSTM) improve this wrench prediction?

RQ3 Does augmenting DeLaN with an LSTM-based residual model achieve joint-torque prediction and friction compensation performance comparable to a TCN-based residual model?

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.

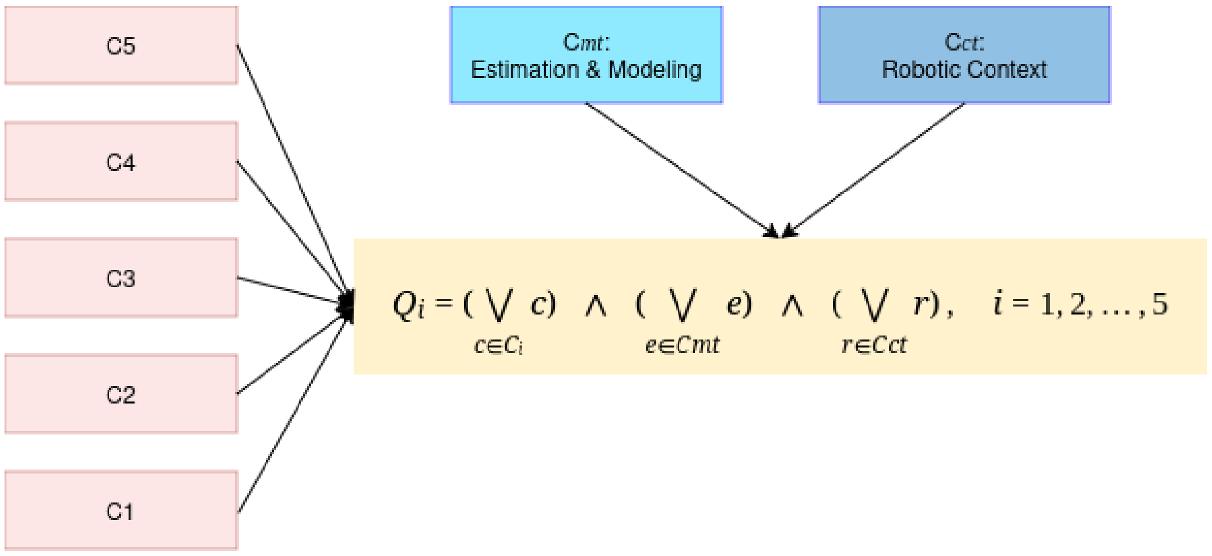
C5
C4
C3
C2
C1
Cmt:
Estimation & Modeling
Cct:
Robotic Context
$Q_i = (\bigvee_{c \in C_i} c) \wedge (\bigvee_{e \in Cmt} e) \wedge (\bigvee_{r \in Cct} r), \quad i = 1, 2, \ldots, 5$

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
$\mathbf{Q_1}$ = Classical / Observers	17	10	8	3
$\mathbf{Q_2}$ = Gaussian Process (GP)	4	0	1	3
$\mathbf{Q_3}$ = Deep Sequence Models	8	4	3	1
$\mathbf{Q_4}$ = Physics-Informed / Diff.	5	0	5	0
$\mathbf{Q_5}$ = 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. [17] and illustrated in Figure 3.

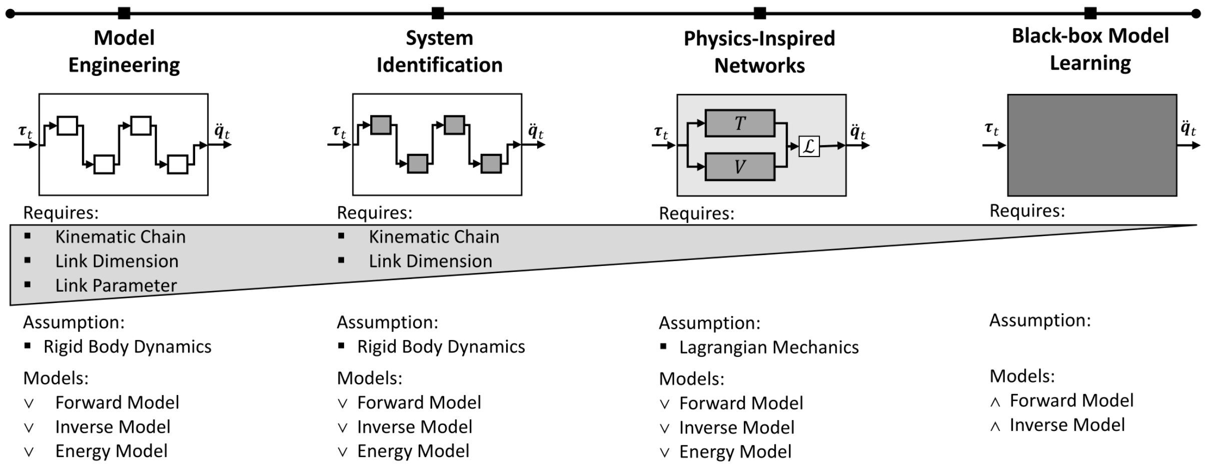

Figure 3: Overview of modelling paradigms for robot dynamics (reproduced by Lutter et al. [17]). The four panels correspond to the approaches discussed in this SoA: classical rigid-body model engineering and system identification (Category Q1), physics-inspired networks such as DeLaN/PINNs (Category 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.

Category Q1 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, 25–30]. 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, 26, 28, 29], and observer-based sensorless force/torque estimation and online payload identification using proprioceptive data [5, 7, 8, 25, 27, 30]. Across Q1, mass is usually identified accurately, CoM moderately well, and inertia emerges as the hardest quantity to estimate robustly [4, 6, 11, 14, 29].

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, 26, 28, 29]. 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, 26]. 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—as systematised, for example, by Swevers et al. [15]—remain a strong baseline for RDPI and PDPI [12, 13, 15, 26, 28, 29]. 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, 25, 27, 30]. 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, 30]. 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, 27, 30]. 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, 29]. 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, 29]. 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, 28]); and the resulting estimators not providing a unified, continuously updated representation of robot and load.

Category Q2 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 [31, 32]

Two studies use GPs to improve joint-space contact-force estimation. In [31], 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 [33] 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 [32]. 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, [34] 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.

Category Q3 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, 20, 21, 35–39].

A first group focuses on deep residual inverse dynamics on top of a nominal RBD model. In [20], 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, [21] 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 [35] 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 [36], 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,[38] proposes a batch ensemble of weak learners (NNs or decision trees) that directly map $(q, \dot{q}, \tau)$ 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 [37, 39], which introduce incremental ensemble learning for PDPI along arbitrary task paths. The initial incremental ensemble [37] adapts weak learners online based on Euclidean distance in feature space but suffers from catastrophic forgetting on previously seen trajectories. The follow-up work [39] 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.

Category Q4 groups physics-informed and differentiable dynamics models, 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 [17] 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, [16] 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 [18], 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, [19] 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, [40] 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, the **Q1** literature shows that classical model-based techniques 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, 26–30]. 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. [REFERENCES HERE!] (These limitations directly motivate the methodological choices and objectives formulated in the problem statement and aim of work.)

Overall, **Q2** shows that GPs are effective residual models for unmodelled robot dynamics and can enhance sensorless contact estimation [31–34]. 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.

Overall, the **Q3** 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 **Q4**, 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 [16–19, 40]. 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, Q4 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 (Q1), through GP residual models (Q2), to black-box deep learning (Q3) and physics-inspired networks (Q4). Classical RBD+LS/observer pipelines in Q1 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. GP and deep black-box methods in Q2-Q3 model unmodelled dynamics and contact well, 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 Q4 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 subsection Limitations of the Current State of the Art then identifies the main shortcomings of existing identification and estimation methods in the literature, thereby motivating the contribution of this work.

The SoA analysis in reveals several recurring gaps that are directly relevant for this work:

- **Weak and fragmentary treatment of payload inertia and effective rigid body.** In Q1, payload mass (and often CoM) can be estimated accurately, but inertia tensors are frequently weakly excited, ill-conditioned or only partially validated, especially in cobot-safe regimes [4, 6, 11, 14, 29] Q2-Q4 largely assume fixed tools/payloads and do not identify the full effective rigid body at all [16, 18, 19, 21, 31, 34, 35] As a result, there is no robust, general mechanism to obtain and maintain an accurate ϕ_{eff} that could be used systematically for tool/gripper compensation and subsequent PDPI.

- **Entangled or black-box treatment of friction, transmission and contact effects.** Many observer-based and LS/NE methods in Q1 are highly sensitive to imperfect friction and transmission models; errors grow around velocity reversals and at higher speeds, even with advanced observers or NN friction terms [7, 25–28, 30] Q2 explicitly models all unmodelled dynamics, friction and contacts as a single GP disturbance [31–33], which improves prediction but makes it hard to separate payload, friction and contact contributions in a physics-consistent way. Q3 and Q4 introduce powerful black-box or PINN components, but again focus on reducing torque/current residuals rather than producing explicit interaction wrenches [16, 19, 21, 35].

- **Dependence on carefully designed excitation and repeated with/without-payload experiments.** Strong RDPI/PDPI results in Q1 and Q3 typically rely on long, highly exciting trajectories (S-curves, Fourier/RRT paths) and repeated runs with and without payloads, often for each new object [4, 10–13, 15, 26, 29, 37–39] This calibration-style use of offline data is practical for commissioning, but gives limited evidence that a single offline-trained model, once obtained, can provide robust dynamic awareness across diverse everyday motions and payload changes without further re-identification.

- **Limited use of temporal models for physics-consistent PDPI and force estimation.** Deep sequence models (LSTM/GRU/TCN) are used either as black-box torque/force predictors [20, 21, 35, 36] or for payload classification/estimation in largely static mappings [10, 37–39]. Physics-informed DeLaN/PINN models, with or without temporal residuals [16, 18, 19, 22], focuse on joint-torque prediction in joint space. No existing work couples a temporal model with an explicit effective rigid-body representation to jointly obtain accurate joint-torque prediction, tool/gripper compensation and interaction-force estimation in the measurement frame.

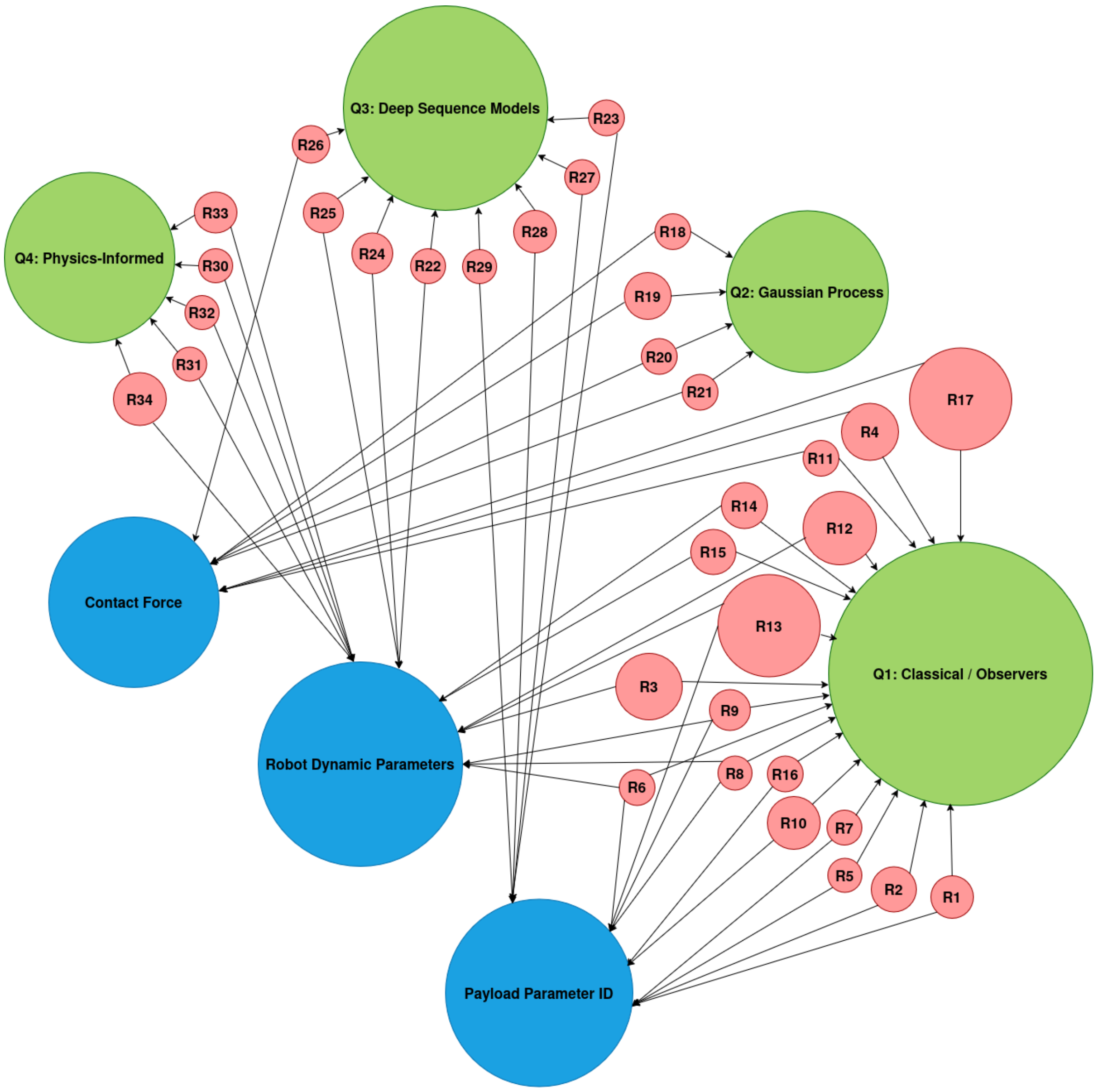

Figure 4: The Concept graph summarising the reviewed literature. Green nodes represent method categories Q1-Q4 with node the size proportional to the number of papers in each category. Blue nodes denote the main estimation goals scaled by how many papers address each goal. Red nodes correspond to individual 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 Deep Lagrangian Networks

Do i have to explain how DeLaN and LSTM work? -> [19, 22] show DeLaN (also with figures)

2.5 Long-Short-Term-Memory

and [20, 21] show LSTM (also with figures). I could also do one figure each, very short explain these two and cite with the given above and DeLaN / LSTM base paper.

3 Methods

Quick overview of what is found in methods (DeLaN + LSTM)-> May start with general DeLaN / LSTM explanation -> moves on with my pipeline mathematics -> then implementation of my pipeline, especially Dataset and input-/output vector dimentions here, also dataset recording offline training online execution.

3.1 Related Work

3.2 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

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

where $\mathbf{q}_k, \dot{\mathbf{q}}_k, \ddot{\mathbf{q}}_k \in \mathbb{R}^n$ are joint position, velocity and acceleration, $\boldsymbol{I}_k \in \mathbb{R}^n$ 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 [24], the measured motor torques are

$$\boldsymbol{\tau}_{\text{motor},k} = k_t\, \boldsymbol{I}_k. \tag{9}$$

The nominal robot-gripper dynamics are parameterised by a Deep Lagrangian Network (DeLaN) [22] with parameters $\boldsymbol{\theta}$ for the conservative dynamics and $\boldsymbol{\psi}$ for friction and other non-conservative terms. The network implements a Lagrangian

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

with positive-definite inertia matrix $\mathbf{M}_{\boldsymbol{\theta}}(\mathbf{q})$ (e.g. represented via a Cholesky-factor network (D.1)) and potential $V_{\boldsymbol{\theta}}(\mathbf{q})$ (D.2) represented by a neural network.[16, 19, 22] Using the Euler-Lagrange equations yields the conservative joint torques

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

where $\mathbf{C}_{\boldsymbol{\theta}}$ and $\mathbf{G}_{\boldsymbol{\theta}}$ are implicitly defined by $\mathcal{L}_{\boldsymbol{\theta}}$ [19, 22].

Joint friction and other non-conservative effects are captured by a Coulomb-viscous friction model, following the improved DeLaN design of [22, 23]. For each joint i we adopt

$$\tau_{\mathrm{fric},i}(\dot{q}_i;\boldsymbol{\psi}) = f_{c,i}\ \mathrm{sgn}(\dot{q}_i) + f_{v,i}\,\dot{q}_i, \tag{12}$$

with learned Coulomb and viscous coefficients $f_{c,i}$ and $f_{v,i}$, collected in $\boldsymbol{\psi}$. In vector form this can be written compactly as

$$\boldsymbol{\tau}_{\mathrm{fric}}(\dot{\mathbf{q}};\boldsymbol{\psi}) = f_{\mathrm{fric}}\big([\dot{\mathbf{q}},\mathrm{sgn}(\dot{\mathbf{q}})];\boldsymbol{\psi}\big), \tag{13}$$

where f_{fric} denotes the joint-wise affine map implementing the Coulomb-viscous law (D.3).

The DeLaN torque prediction is the sum of conservative and frictional parts,

$$\hat{\boldsymbol{\tau}}_{\mathrm{DeLaN}}(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}};\boldsymbol{\theta},\boldsymbol{\psi}) = \boldsymbol{\tau}_{\mathrm{cons}}(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}};\boldsymbol{\theta}) + \boldsymbol{\tau}_{\mathrm{fric}}(\dot{\mathbf{q}};\boldsymbol{\psi}). \tag{14}$$

For compactness we write this as a parametric model

$$\hat{\boldsymbol{\tau}}_{\mathrm{DeLaN}}(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}};\boldsymbol{\theta},\boldsymbol{\psi}) = f_{\mathrm{DeLaN}}\big(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}};\boldsymbol{\theta},\boldsymbol{\psi}\big), \tag{15}$$

so that for sample k

$$\hat{\boldsymbol{\tau}}_{\mathrm{DeLaN},k} = f_{\mathrm{DeLaN}}\big(\mathbf{q}_k,\dot{\mathbf{q}}_k,\ddot{\mathbf{q}}_k;\boldsymbol{\theta},\boldsymbol{\psi}\big). \tag{16}$$

The parameters $(\boldsymbol{\theta},\boldsymbol{\psi})$ are trained offline by minimising a joint-space regression loss [19, 22]

$$\mathcal{L}_{\mathrm{DeLaN}}(\boldsymbol{\theta},\boldsymbol{\psi}) = \frac{1}{N}\sum_{k=1}^{N} \left\|\hat{\boldsymbol{\tau}}_{\mathrm{DeLaN}}(\mathbf{q}_k,\dot{\mathbf{q}}_k,\ddot{\mathbf{q}}_k;\boldsymbol{\theta},\boldsymbol{\psi}) - \boldsymbol{\tau}_{\mathrm{motor},k}\right\|_2^2. \tag{17}$$

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.

3.3 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 fine-grained nonlinear friction. Using the same robot-gripper dataset (still without payload and without contact), we first compute the joint-space residual torques

$$\boldsymbol{r}_{\tau,k} = \boldsymbol{\tau}_{\mathrm{motor},k} - \hat{\boldsymbol{\tau}}_{\mathrm{DeLaN},k}. \tag{18}$$

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

$$\mathbf{x}_k = \Big[\mathbf{q}_{k-H+1:k},\ \dot{\mathbf{q}}_{k-H+1:k},\ \ddot{\mathbf{q}}_{k-H+1:k},\ \hat{\boldsymbol{\tau}}_{\mathrm{DeLaN},k-H+1:k}\Big], \tag{19}$$

where $\mathbf{q}_{a:b}$ denotes the stacked joint vectors $(\mathbf{q}_a,\ldots,\mathbf{q}_b)$, and analogously for $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$ and $\hat{\boldsymbol{\tau}}_{\mathrm{DeLaN}}$. 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

$$\hat{\boldsymbol{r}}_{\tau,k} = f_{\mathrm{LSTM}}(\mathbf{x}_k; \boldsymbol{\varphi}) \in \mathbb{R}^n. \tag{20}$$

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

$$\mathcal{L}_{\mathrm{LSTM}}(\boldsymbol{\varphi}) = \frac{1}{N_H} \sum_{k=H}^{N} \left\| \hat{\boldsymbol{r}}_{\tau,k} - \boldsymbol{r}_{\tau,k} \right\|_2^2 , \tag{21}$$

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

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

$$\hat{\boldsymbol{\tau}}_{\mathrm{RG},k} = \hat{\boldsymbol{\tau}}_{\mathrm{DeLaN},k} + \hat{\boldsymbol{r}}_{\tau,k}. \tag{22}$$

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

$$\hat{\vec{F}}_{\mathrm{RG},k} = {}^{S}J(\mathbf{q}_k)^{-\top}\, \hat{\boldsymbol{\tau}}_{\mathrm{RG},k}. \tag{23}$$

Since both stages are trained exclusively on data without payload and without environment contact, $\hat{\boldsymbol{\tau}}_{\mathrm{RG},k}$ and $\hat{\vec{F}}_{\mathrm{RG},k}$ 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.

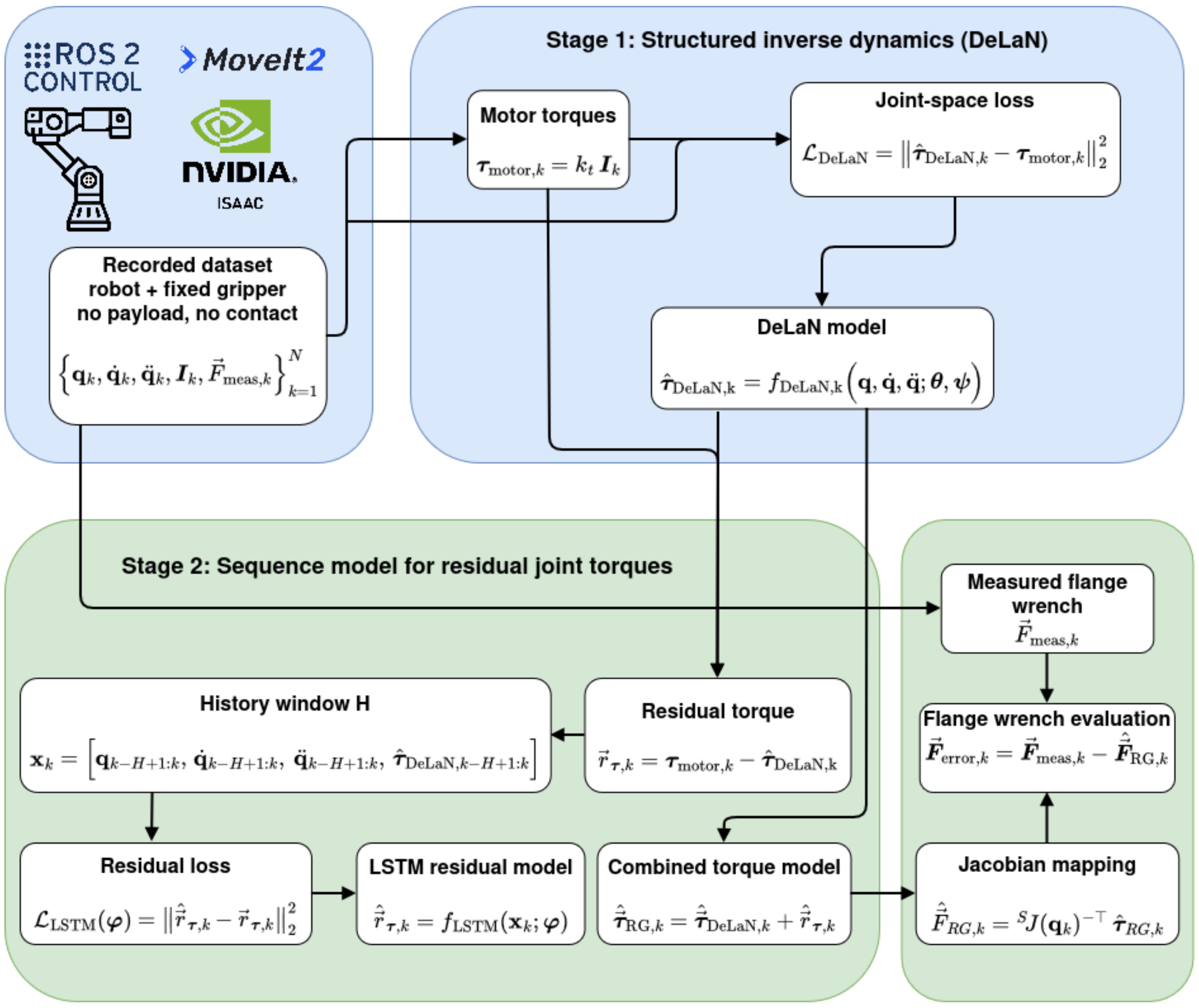
ROS 2 CONTROL
MoveIt2
NVIDIA ISAAC
Recorded dataset robot + fixed gripper no payload, no contact
$\left\{\mathbf{q}_k, \dot{\mathbf{q}}_k, \ddot{\mathbf{q}}_k, \boldsymbol{I}_k, \vec{F}_{\text{meas},k}\right\}_{k=1}^{N}$
Stage 1: Structured inverse dynamics (DeLaN)
Motor torques
$\boldsymbol{\tau}_{\text{motor},k} = k_t \boldsymbol{I}_k$
Joint-space loss
$\mathcal{L}_{\text{DeLaN}} = \left\|\hat{\boldsymbol{\tau}}_{\text{DeLaN},k} - \boldsymbol{\tau}_{\text{motor},k}\right\|_2^2$
DeLaN model
$\hat{\boldsymbol{\tau}}_{\text{DeLaN,k}} = f_{\text{DeLaN,k}}\left(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}; \boldsymbol{\theta}, \boldsymbol{\psi}\right)$
Stage 2: Sequence model for residual joint torques
Measured flange wrench
$\vec{F}_{\text{meas},k}$
History window H
$\mathbf{x}_k = \left[\mathbf{q}_{k-H+1:k}, \dot{\mathbf{q}}_{k-H+1:k}, \ddot{\mathbf{q}}_{k-H+1:k}, \hat{\boldsymbol{\tau}}_{\text{DeLaN},k-H+1:k}\right]$
Residual torque
$\vec{r}_{\tau,k} = \boldsymbol{\tau}_{\text{motor},k} - \hat{\boldsymbol{\tau}}_{\text{DeLaN},k}$
Flange wrench evaluation
$\vec{F}_{\text{error},k} = \vec{F}_{\text{meas},k} - \hat{\vec{F}}_{\text{RG},k}$
Residual loss
$\mathcal{L}_{\text{LSTM}}(\boldsymbol{\varphi}) = \|\hat{\vec{r}}_{\tau,k} - \vec{r}_{\tau,k}\|_2^2$
LSTM residual model
$\hat{\vec{r}}_{\tau,k} = f_{\text{LSTM}}(\mathbf{x}_k; \boldsymbol{\varphi})$
Combined torque model
$\hat{\vec{\tau}}_{\text{RG},k} = \hat{\boldsymbol{\tau}}_{\text{DeLaN},k} + \hat{\vec{r}}_{\tau,k}$
Jacobian mapping
$\hat{\vec{F}}_{RG,k} = {}^{S}J(\mathbf{q}_k)^{-\top}\, \hat{\boldsymbol{\tau}}_{RG,k}$

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.

3.4 Data preprocessing and dataset construction

The starting point is a long-format log of robot measurements, exported as a CSV with columns

`Time`, `Joint Name`, `Position`, `Velocity`, `Acceleration`, `Effort`.

Each row corresponds to a single joint at a given timestamp. In this first iteration, the joint velocities and accelerations are taken directly from the controller without additional filtering, and the joint effort is interpreted as a torque-like quantity. The precise mapping from motor currents to joint torques is introduced later via the torque constant k_t (cf. Section 3).

Joint selection (UR5 arm). From the full log we select only the six UR5 joints

$\mathcal{J} = \big\{\texttt{ur5\_shoulder\_pan\_joint}, \texttt{ur5\_shoulder\_lift\_joint}, \texttt{ur5\_elbow\_joint}, \texttt{ur5\_wrist\_1\_}$

which defines $n_{\text{dof}} = 6$ for all subsequent processing. For each timestamp t we thus obtain a 6-dimensional joint configuration, velocity, acceleration and effort vector.

Frame-wise representation and trajectory segmentation. Let $\{t_f\}_{f=1}^{F}$ denote the sorted set of unique timestamps in the log. For each frame index f and each joint $j \in \mathcal{J}$ we collect

$$q_{f,j}, \quad \dot{q}_{f,j}, \quad \ddot{q}_{f,j}, \quad \tau_{f,j}^{\text{eff}},$$

where $\tau_{f,j}^{\text{eff}}$ denotes the measured joint effort. Stacking over joints yields frame-wise vectors

$$\mathbf{q}_f,\ \dot{\mathbf{q}}_f,\ \ddot{\mathbf{q}}_f,\ \boldsymbol{\tau}_f^{\text{eff}} \in \mathbb{R}^6.$$

To obtain multiple trajectories from a single continuous log, we segment the frames into fixed-length windows. For a chosen number of frames per trajectory, T_{seg}, we define a trajectory index

$$\operatorname{traj}(f) = \left\lfloor \frac{f-1}{T_{\text{seg}}} \right\rfloor,$$

and group all frames with the same $\operatorname{traj}(f)$ into one trajectory.

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

$$\mathbf{t}^{(i)} = \big(t_1^{(i)}, \dots, t_{T_i}^{(i)}\big) \in \mathbb{R}^{T_i}, \tag{24}$$

$$\mathbf{Q}^{(i)} = \begin{bmatrix} \mathbf{q}_1^{(i)\top} \\ \vdots \\ \mathbf{q}_{T_i}^{(i)\top} \end{bmatrix} \in \mathbb{R}^{T_i \times 6}, \tag{25}$$

$$\dot{\mathbf{Q}}^{(i)} = \begin{bmatrix} \dot{\mathbf{q}}_1^{(i)\top} \\ \vdots \\ \dot{\mathbf{q}}_{T_i}^{(i)\top} \end{bmatrix} \in \mathbb{R}^{T_i \times 6}, \tag{26}$$

$$\ddot{\mathbf{Q}}^{(i)} = \begin{bmatrix} \ddot{\mathbf{q}}_1^{(i)\top} \\ \vdots \\ \ddot{\mathbf{q}}_{T_i}^{(i)\top} \end{bmatrix} \in \mathbb{R}^{T_i \times 6}, \tag{27}$$

$$\mathbf{T}^{(i)} = \begin{bmatrix} \boldsymbol{\tau}_1^{\text{eff},(i)\top} \\ \vdots \\ \boldsymbol{\tau}_{T_i}^{\text{eff},(i)\top} \end{bmatrix} \in \mathbb{R}^{T_i \times 6}. \tag{28}$$

In short, each trajectory i is represented by

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

with shapes

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

Train/test split by trajectory. Let $\mathcal{I} = \{1, \dots, N_{\text{traj}}\}$ denote the set of all trajectory indices. We randomly permute $\mathcal{I}$ and partition it into disjoint training and test index sets,

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

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.

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

$$\texttt{train\_labels}, \texttt{train\_t}, \texttt{train\_q}, \texttt{train\_qd}, \texttt{train\_qdd}, \texttt{train\_tau}, \tag{29}$$

$$\texttt{test\_labels}, \texttt{test\_t}, \texttt{test\_q}, \texttt{test\_qd}, \texttt{test\_qdd}, \texttt{test\_tau}. \tag{30}$$

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

$$\texttt{train\_q}[i] \mathrel{\widehat{=}} \mathbf{Q}^{(i)} \in \mathbb{R}^{T_i \times 6},$$

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

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

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 $\boldsymbol{\tau}_{\text{motor},k} = k_t \, \boldsymbol{I}_k$ is applied later in the training pipeline (cf. (6)).

Flattening trajectories for DeLaN training. 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 $\text{vstack}(\cdot)$, the training set becomes

$$\tilde{\mathbf{Q}}_{\text{train}} = \text{vstack}\big(\texttt{train\_q}[i]\big) \in \mathbb{R}^{N_{\text{train,tot}} \times 6}, \tag{31}$$

$$\dot{\tilde{\mathbf{Q}}}_{\text{train}} = \text{vstack}\big(\texttt{train\_qd}[i]\big) \in \mathbb{R}^{N_{\text{train,tot}} \times 6}, \tag{32}$$

$$\ddot{\tilde{\mathbf{Q}}}_{\text{train}} = \text{vstack}\big(\texttt{train\_qdd}[i]\big) \in \mathbb{R}^{N_{\text{train,tot}} \times 6}, \tag{33}$$

$$\tilde{\mathbf{T}}_{\text{train}} = \text{vstack}\big(\texttt{train\_tau}[i]\big) \in \mathbb{R}^{N_{\text{train,tot}} \times 6}, \tag{34}$$

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

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

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

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

$$\boldsymbol{\phi}(\mathbf{q}_k) = \begin{bmatrix} \mathbf{q}_k \\ \sin(\mathbf{q}_k) \\ \cos(\mathbf{q}_k) \end{bmatrix} \in \mathbb{R}^{3n_{\mathrm{dof}}}, \tag{35}$$

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. The feature vector $\boldsymbol{\phi}(\mathbf{q}_k)$ is used as input to the inertia MLPs and the potential network described in Appendix D.

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

$$\hat{\boldsymbol{\tau}}_{\mathrm{DeLaN},k} = f_{\mathrm{DeLaN}}(\mathbf{q}_k, \dot{\mathbf{q}}_k, \ddot{\mathbf{q}}_k; \boldsymbol{\theta}, \boldsymbol{\psi}) \tag{36}$$

denote the DeLaN torque prediction for sample k.

We compute empirical variances of the training torques

$$\boldsymbol{\sigma}_\tau^2 = \mathrm{Var}\big(\tilde{\mathbf{T}}_{\mathrm{train}}\big) \in \mathbb{R}^6, \tag{37}$$

and define a diagonal weighting matrix

$$\mathbf{W}_\tau = \mathrm{diag}\big(\boldsymbol{\sigma}_\tau^{-1}\big) \in \mathbb{R}^{6\times 6}, \tag{38}$$

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

$$\mathcal{L}_{\mathrm{DeLaN}}(\boldsymbol{\theta}, \boldsymbol{\psi}) = \frac{1}{N_{\mathrm{train,tot}}} \sum_{k=1}^{N_{\mathrm{train,tot}}} \big\|\mathbf{W}_\tau\big(\hat{\boldsymbol{\tau}}_{\mathrm{DeLaN},k} - \boldsymbol{\tau}_{\mathrm{motor},k}\big)\big\|_2^2, \tag{39}$$

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.

Replay buffer and random mini-batch sampling. 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 $S \subset \{1, \ldots, N_{\text{train,tot}}\}$ with $|S| = B$ is drawn (e.g. by taking a random permutation of indices and slicing), and the corresponding batch

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

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.

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

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

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 $\boldsymbol{\tau}_{\text{motor},k}$.

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

Export DeLaN Residuals

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

$$\hat{\boldsymbol{\tau}}_{\text{DeLaN},k} = f_{\text{DeLaN}}(\mathbf{q}_k, \dot{\mathbf{q}}_k, \ddot{\mathbf{q}}_k; \boldsymbol{\theta}, \boldsymbol{\psi}), \tag{40}$$

which we stack into a matrix

$$\hat{\mathbf{T}}_{\text{DeLaN,train}} = \begin{bmatrix} \hat{\boldsymbol{\tau}}_{\text{DeLaN},1}{}^{\top} \\ \vdots \\ \hat{\boldsymbol{\tau}}_{\text{DeLaN},N_{\text{train,tot}}}{}^{\top} \end{bmatrix} \in \mathbb{R}^{N_{\text{train,tot}} \times 6}. \tag{41}$$

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

$$\boldsymbol{r}_{\tau,k} = \boldsymbol{\tau}_{\text{motor},k} - \hat{\boldsymbol{\tau}}_{\text{DeLaN},k}, \qquad k = 1, \ldots, N_{\text{train,tot}}, \tag{42}$$

and collect them in

$$\mathbf{R}_{\tau,\text{train}} = \begin{bmatrix} \boldsymbol{r}_{\tau,1}{}^\top \\ \vdots \\ \boldsymbol{r}_{\tau,N_{\text{train,tot}}}{}^\top \end{bmatrix} \in \mathbb{R}^{N_{\text{train,tot}} \times 6}. \tag{43}$$

Build LSTM Windows

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

$$\mathbf{x}_k = \Big[\mathbf{q}_{k-H+1:k},\, \dot{\mathbf{q}}_{k-H+1:k},\, \ddot{\mathbf{q}}_{k-H+1:k},\, \hat{\boldsymbol{\tau}}_{\text{DeLaN},k-H+1:k}\Big], \tag{44}$$

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,

$$\mathbf{y}_k = \boldsymbol{r}_{\tau,k} \in \mathbb{R}^6. \tag{45}$$

Stacking all valid windows yields the Stage 2 training dataset

$$\mathbf{X}_{\text{train}}^{\text{LSTM}} = \{\mathbf{x}_k\}_{k=H}^{N_{\text{train,tot}}}, \tag{46}$$

$$\mathbf{Y}_{\text{train}}^{\text{LSTM}} = \{\mathbf{y}_k\}_{k=H}^{N_{\text{train,tot}}}, \tag{47}$$

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 $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \hat{\boldsymbol{\tau}}_{\text{DeLaN}})$ 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.

LSTM Training

We train an LSTM-based 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 <u>scaled</u> residuals.

Train-only standardisation. 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,

$$\mu_{x,j} = \frac{1}{N_H H} \sum_{k=1}^{N_H} \sum_{t=1}^{H} X_{\text{train}}^{\text{LSTM}}[k,t,j], \qquad \sigma_{x,j}^2 = \frac{1}{N_H H} \sum_{k=1}^{N_H} \sum_{t=1}^{H} \big(X_{\text{train}}^{\text{LSTM}}[k,t,j] - \mu_{x,j}\big)^2, \tag{48}$$

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,

$$\mu_{y,i} = \frac{1}{N_H} \sum_{k=1}^{N_H} Y_{\text{train}}^{\text{LSTM}}[k,i], \qquad \sigma_{y,i}^2 = \frac{1}{N_H} \sum_{k=1}^{N_H} \big(Y_{\text{train}}^{\text{LSTM}}[k,i] - \mu_{y,i}\big)^2, \tag{49}$$

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

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

the standardised inputs and targets are then given by

$$\mathbf{x}_{k,t}^{\mathrm{n}} = \mathbf{W}_x \big(\mathbf{x}_{k,t} - \boldsymbol{\mu}_x\big), \tag{50}$$

$$\boldsymbol{r}_{\tau,k}^{\mathrm{s}} = \mathbf{W}_y \big(\boldsymbol{r}_{\tau,k} - \boldsymbol{\mu}_y\big), \tag{51}$$

with $\boldsymbol{r}_{\tau,k} = \mathbf{y}_k$. 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.

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

$$\hat{\boldsymbol{r}}_{\tau,k}^{\mathrm{s}} = g_{\boldsymbol{\phi}}\big(\mathbf{x}_k^{\mathrm{n}}\big) \in \mathbb{R}^6, \tag{52}$$

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

$$\mathcal{L}_{\mathrm{LSTM}}(\boldsymbol{\phi}) = \frac{1}{N_H} \sum_{k=1}^{N_H} \big\| \hat{\boldsymbol{r}}_{\tau,k}^{\mathrm{s}} - \boldsymbol{r}_{\tau,k}^{\mathrm{s}} \big\|_2^2, \tag{53}$$

which corresponds to a balanced, per-joint normalisation via $\boldsymbol{\sigma}_y$. 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.

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

$$\hat{\boldsymbol{r}}_{\tau,k} = \mathbf{W}_y^{-1} \hat{\boldsymbol{r}}_{\tau,k}^{\mathrm{s}} + \boldsymbol{\mu}_y, \tag{54}$$

where $\mathbf{W}_y^{-1} = \operatorname{diag}(\boldsymbol{\sigma}_y)$.

4 Experimental Setup

Measurement convention (motor current domain). From this chapter onwards, we formulate training objectives and evaluation metrics in the measured actuation domain provided by the dataset, i.e., per-joint motor currents $\mathbf{i}_{\mathrm{motor}}$ 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, i.e., by multiplying with k_t. Consistent with this convention and the pipeline overview in Fig. 5, we omit the conversion $\boldsymbol{\tau}_{\mathrm{motor}} = k_t \, \mathbf{i}_{\mathrm{motor}}$ and the rightmost Jacobian-based flange-wrench mapping block, and evaluate the approach directly as a combined motor-current model $\hat{\mathbf{i}}_{\mathrm{comb}}$.

4.1 Dataset of Collaborative Robots

Dataset source and scope. All experiments are based on the publicly available “Dataset of Collaborative Robots for Energy Consumption Modeling” released via IEEE DataPort [41] and documented in [42]. The dataset contains measurements from two Universal Robots platforms (UR3e and UR10e) recorded both without load and with an external payload.

Recorded signals. 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.

Data collection protocol. 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 [42]:

$$q_i(t) = q_{i0} + A_i \cos\big(2\pi f_i t + \varphi_i\big), \tag{55}$$

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 $1.5\,\mathrm{kg}$ and RobotiQ 2F-85 gripper $1\,\mathrm{kg}$) [42].

Sampling and dataset size. The signals are recorded at $100\,\mathrm{Hz}$. For each robot (and load condition), the dataset provides $50{,}000$ samples for training and $5{,}000$ samples for testing [42]. 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 [42].

4.2 DeLaN + LSTM - Learning Curve Stroy

Key findings. The learning-curve study shows that trajectory quantity K is a primary driver of stability and generalisation for both stages: small subsets lead to higher dispersion and occasional failure modes, whereas sufficiently large K yields consistent convergence and low per-joint errors. In this regime, freezing the best DeLaN and learning residual dynamics with an LSTM provides an additional systematic reduction of the remaining modelling error.

4.2.1 K-Domination

Motivation and experimental protocol. 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.

Trajectory pool and signals. 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 $100\,\mathrm{Hz}$. Since trajectory durations vary substantially (roughly 5–$40\,\mathrm{s}$), all filtering and preprocessing steps are applied per trajectory.

Per-trajectory low-pass filtering. To attenuate sensor noise while avoiding temporal misalignment, we apply a 4th-order Butterworth low-pass filter with cutoff frequency $f_c = 10\,\mathrm{Hz}$. 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 derived from the filtered velocities via numerical differentiation and, if filtering is enabled, are filtered analogously.

Subsampling by K. 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 (i.e., trajectories are never split across subsets).

Stage 1 (DeLaN) configuration and model selection. For each (K, seed) subset, we train five DeLaN initialisations (seeds $s \in \{0, \ldots, 4\}$) 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 [17]): 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.

Stage 2 (LSTM) configuration. 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`, i.e., each LSTM input time step concatenates $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \hat{\mathbf{i}}_{\mathrm{DeLaN}})$. 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 $100\,\mathrm{Hz}$
Require: $K \in \{8, 16, 32, 48, 64, 86, 122\}$, dataset seeds $\mathcal{D} = \{0, 1, 2\}$
Require: DeLaN seeds $\mathcal{S}_{\mathrm{DeLaN}} = \{0, \ldots, 4\}$
Ensure: Aggregated learning curves and per-joint motor-current RMSE as a function of $K$
  Low-pass filter each trajectory ($4^{\mathrm{th}}$-order Butterworth, $f_c = 10\,\mathrm{Hz}$, 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 $s \in \mathcal{S}_{\mathrm{DeLaN}}$ do
        Train DeLaN with fixed hyperparameters  ▷ 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  ▷ 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
```

Aggregation and reporting. 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.

4.2.2 Best Model Approch

Motivation. 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 (i) which trajectories end up in the train/validation/test split (dataset seed) and (ii) 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). **main objective best DelaN -> best LSTM here**

Objective DeLaN 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 (i) reduce seed sensitivity (tight IQR with few or no catastrophic runs), (ii) reach low validation error quickly and consistently (stable early-stopping behaviour), and (iii) 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.

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

$$\mathrm{score}(h, d) = \mathrm{median}(\mathrm{RMSE}_{\mathrm{val}}) + \lambda \cdot \mathrm{IQR}(\mathrm{RMSE}_{\mathrm{val}}) + P \cdot \rho(d, h), \tag{56}$$

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 $\mathrm{score}(h, d)$ across dataset seeds and select the preset $h^\star$ with minimal median score. Lower values of $\mathrm{score}(h, d)$ (and thus of its median across d) therefore indicate both high accuracy and high reliability.

Aggregate scatter plots. In addition to the scalar score, we report two aggregated scatter plots for comparing presets h: (i) median validation RMSE versus IQR of validation RMSE, and (ii) 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.

DeLaN presets ($|\mathcal{H}_{\mathrm{DeLaN}}| = 5$). 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 $K = 84$:

1. **Baseline (Lutter-like):** SoftPlus, batch size 1024, learning rate 10^{-4}, weight decay 10^{-5}, width 256, depth 2.
2. **Smaller capacity:** baseline with width 128, depth 2.
3. **Deeper network:** baseline with width 256, depth 3.
4. **More regularisation:** baseline with weight decay 10^{-4}.
5. **Lower step size:** baseline with learning rate $5 \cdot 10^{-5}$.

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 $f_c = 10\,\mathrm{Hz}$
Require: Trajectory pool size $K = 86$ with fixed split $(N_{\text{train}}, N_{\text{val}}, N_{\text{test}}) = (60, 8, 17)$
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 $\mathrm{train\_loss}(e), \mathrm{val\_mse}(e)$, motor-current RMSE over progress, motor-current RMSE per joint, and $\mathrm{RMSE}_{\text{val}}, \mathrm{RMSE}_{\text{test}}, e^\star$
              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 $\rho(d,h) = \#\mathrm{diverged}/|\mathcal{S}_{\text{DeLaN}}|$
          Score $s(h,d) = \tilde{r}_{\text{val}}(d,h) + \lambda\, q_{\text{val}}(d,h) + P\,\rho(d,h)$
          Choose best seed $s^\star$ with minimal $\mathrm{RMSE}_{\text{val}}$ among non-diverged runs
          Save fold artifacts for $(h,d)$      ▷ median±IQR curves; best-run plots; checkpoint $(h,d,s^\star)$
          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 $h$:
      $S_h = \mathrm{median}_d(s(h,d))$, $Q_h = \mathrm{IQR}_d(s(h,d))$, $x_h = \mathrm{median}_d(\tilde{r}_{\text{val}}(d,h))$, $y_h^{\text{test}} = \mathrm{median}_d(\tilde{r}_{\text{test}}(d,h))$
      Save hyper-set summary for $h$ and add one point to scatter datasets
  Choose best hyper-set $h^\star = \arg\min_h S_h$
  Plot scatter A: $x$ vs stability (median or IQR of $q_{\text{val}}$ across $d$)
  Plot scatter B: $x$ vs $y_h^{\text{test}}$                     ▷ validation vs test alignment
  Choose final DeLaN checkpoint: $d^\star = \arg\min_d s(h^\star, d)$, use saved $(h^\star, d^\star, s^\star)$
  Freeze DeLaN and export residuals
  Freeze best DeLaN parameters and export residual motor currents per trajectory
```

4.2.3 Objective LSTM

Motivation. 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

$$\boldsymbol{r}_{i,k} = \mathbf{i}_{\text{motor},k} - \hat{\mathbf{i}}_{\text{DeLaN},k}. \tag{57}$$

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.

Objective and selection criterion. 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

$$\hat{\mathbf{i}}_{\text{RG},k} = \hat{\mathbf{i}}_{\text{DeLaN},k} + \hat{\boldsymbol{r}}_{i,k} \tag{58}$$

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

Ablation axes. Algorithm 3 sweeps three window lengths $H \in \{50, 100, 150\}$ and four feature modes f that probe whether residual dynamics are primarily explained by kinematics, by the DeLaN current baseline, or by their combination: `state` uses $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$, `i_hat` uses $\hat{\mathbf{i}}_{\text{DeLaN}}$, `full` uses $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \hat{\mathbf{i}}_{\text{DeLaN}})$, and `state_ihat` uses $(\dot{\mathbf{q}}, \ddot{\mathbf{q}}, \hat{\mathbf{i}}_{\text{DeLaN}})$ (dropping $\mathbf{q}$). 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.

Stability-aware ranking across seeds. 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 $(f^\star, H^\star) = \arg\min_{(f,H)} S_{f,H}$ as defined in Algorithm 3.

Algorithm 3 Best-model selection for the LSTM residual model

```
Require: Dataset seeds $\mathcal{D}$, feature modes $\mathcal{F} = \{\texttt{full}, \texttt{state}, \texttt{i\_hat}, \texttt{state\_ihat}\}$, window lengths $\mathcal{H} = \{50, 100, 150\}$
Require: LSTM seeds $\mathcal{S}_{\mathrm{LSTM}}$, 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 $\mathcal{C}_{\mathrm{train}}, \mathcal{C}_{\mathrm{val}} \leftarrow \emptyset$
              for all $\ell \in \mathcal{S}_{\mathrm{LSTM}}$ do
                  Train LSTM on NPZ$(d, f, H)$ with seed $\ell$          ▷ early stopping; save best checkpoint
                  if training crashed or NaN/Inf or missing metrics then
                      Mark diverged
                  else
                      Record curves $\mathrm{train\_loss}(e), \mathrm{val\_loss}(e)$ and $e^\star$
                      Record residual metrics $\mathrm{RMSE}_{\mathrm{res}}, \mathrm{MSE}_{\mathrm{res}}$
                      Run combined evaluation with DeLaN baseline for $d$ and this LSTM
                      Record $\mathrm{RMSE}_{\mathrm{rg}}$, gain, gain_ratio
                  Append curves and run metrics to $\mathcal{C}_{\mathrm{train}}, \mathcal{C}_{\mathrm{val}}, \mathcal{R}$
              Align curves to common length $E_{\max}$ and aggregate median $\pm$ IQR
              Compute medians and IQRs over non-diverged runs:
  $\tilde{r}_{\mathrm{res}}(d, f, H)$, $q_{\mathrm{res}}(d, f, H)$, $\tilde{r}_{\mathrm{rg}}(d, f, H)$, $q_{\mathrm{rg}}(d, f, H)$, $\tilde{g}(d, f, H)$, $q_g(d, f, H)$
              Compute divergence rate $\rho(d, f, H) = \#\mathrm{diverged}/|\mathcal{S}_{\mathrm{LSTM}}|$
              Score $s(d, f, H) = \tilde{r}_{\mathrm{rg}}(d, f, H) + \lambda\, q_{\mathrm{rg}}(d, f, H) + P\, \rho(d, f, H)$
              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 $d$:
  $S_{f,H} = \mathrm{median}_d(s(d, f, H))$, $Q_{f,H} = \mathrm{IQR}_d(s(d, f, H))$, $\tilde{r}_{\mathrm{rg}}(f, H) = \mathrm{median}_d(\tilde{r}_{\mathrm{rg}}(d, f, H))$, $\tilde{g}(f, H) = \mathrm{median}_d(\tilde{g}(d, f, H))$
          Save per-$(f, H)$ summary and add to boxplots / progress-curve aggregation
  Final selection
  Choose best $(f^\star, H^\star) = \arg\min_{(f,H)} S_{f,H}$
  Report boxplots and progress-normalized RMSE curves for best configs
```

5 Experimental Results

Chapter overview. This chapter reports experimental results for the proposed two-stage pipeline, covering (i) the effect of the number of available trajectories (K) on DeLaN and residual LSTM training dynamics and accuracy, (ii) a quantitative comparison to the dataset baseline, and (iii) a best-model evaluation on “with load” trajectories.

5.1 Learning Curve Results

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.

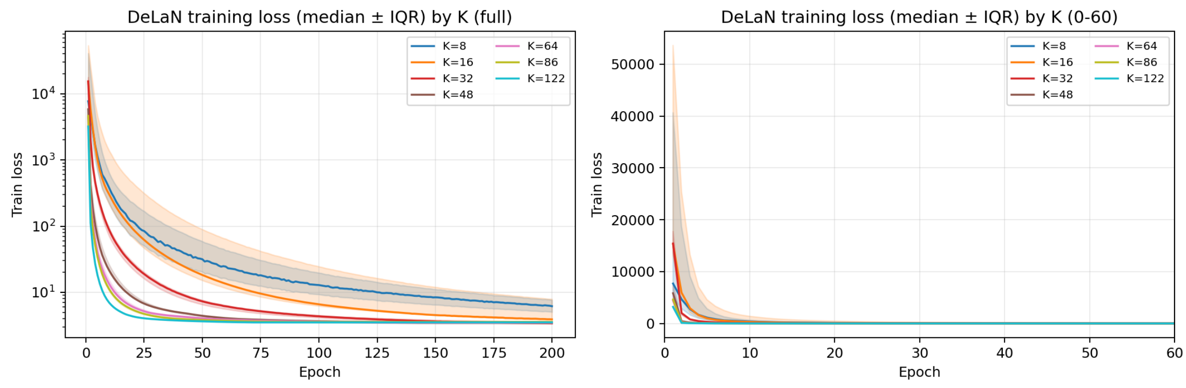

Figure 6: DeLaN training loss by K shown as median $\pm$ IQR across dataset seeds (with seed-wise aggregation across DeLaN initialisations).

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.

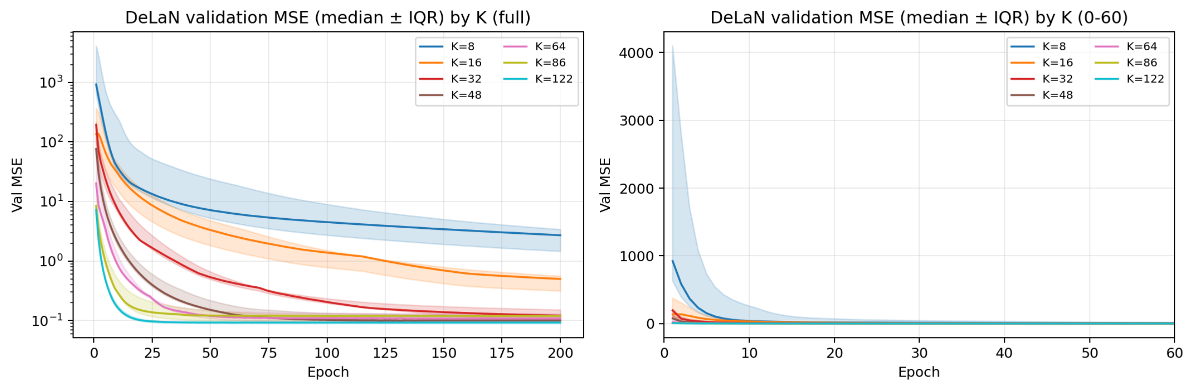

Figure 7: DeLaN validation MSE (motor current) by K shown as median $\pm$ IQR across dataset seeds (with seed-wise aggregation across DeLaN initialisations).

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.

Stage 1 (DeLaN): motor-current error as a function of motion progress. 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 $3\,\mathrm{A}$ 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.

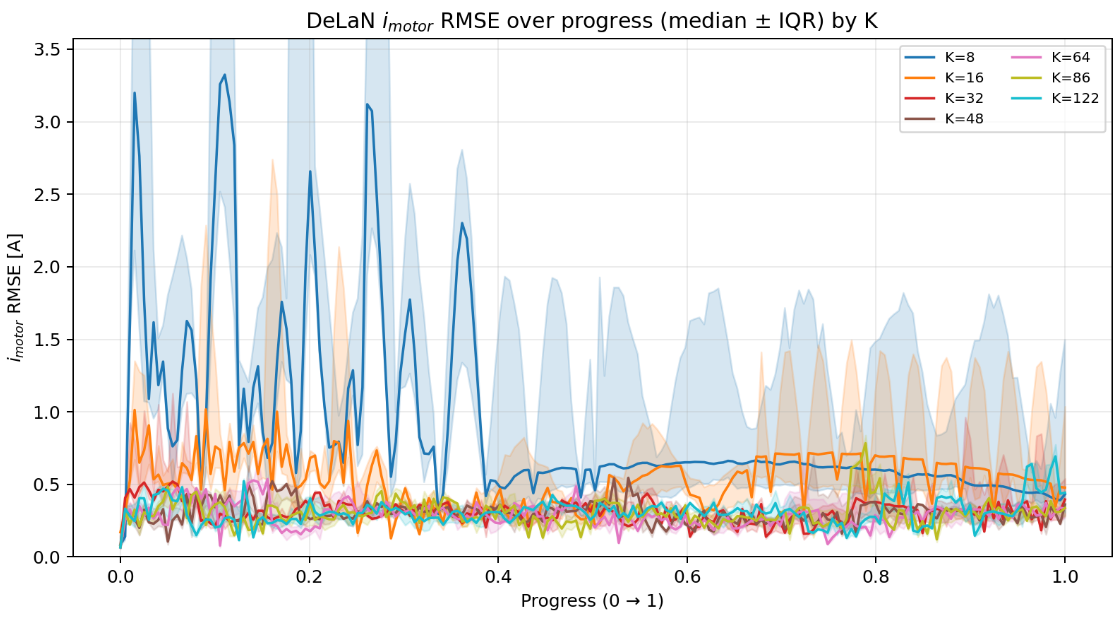

Figure 8: DeLaN motor-current RMSE over normalised progress ($0 \to 1$) by K shown as median $\pm$ IQR (shaded bands).

Stage 1 (DeLaN): per-joint error. 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 $3\,\mathrm{A}$. For intermediate K (e.g., $K = 16$), 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.

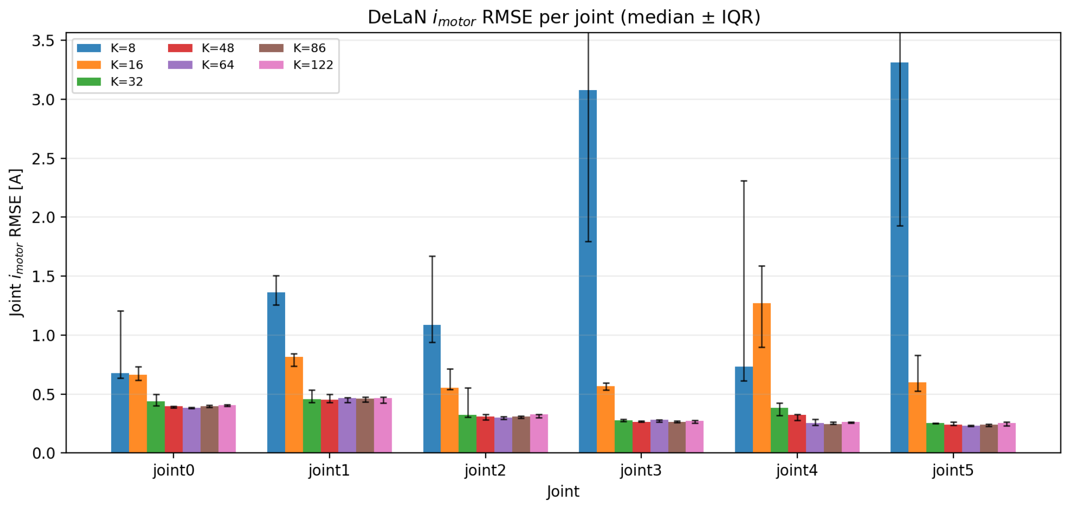

Figure 9: DeLaN motor-current RMSE per joint (median $\pm$ IQR) for all evaluated values of K (bars) with IQR error bars.

Stage 2 (LSTM): learning dynamics. 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.

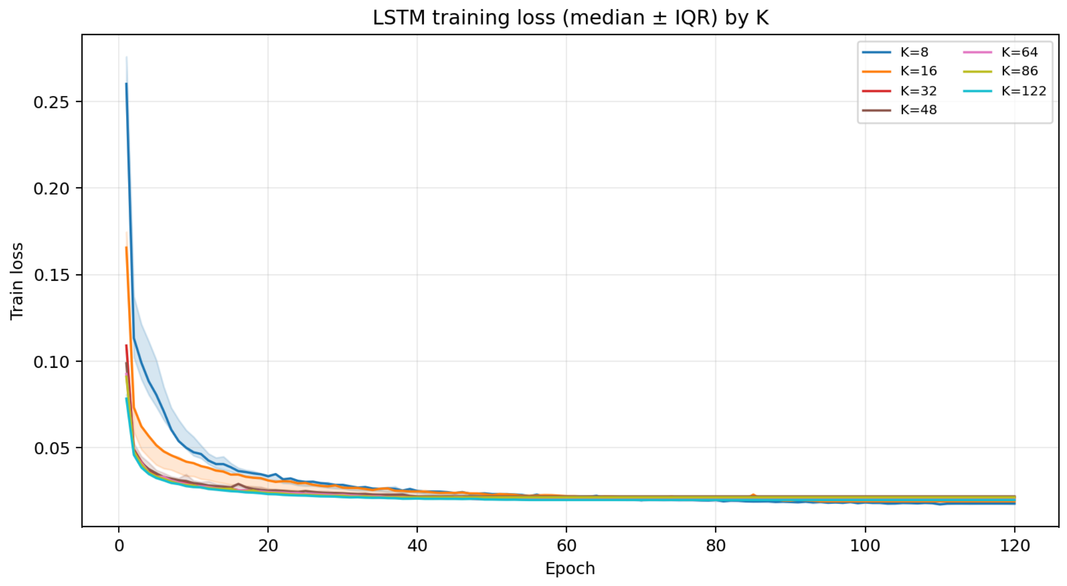

Figure 10: LSTM training loss by K shown as median $\pm$ IQR across dataset seeds ($H = 100$, feature mode `full`).

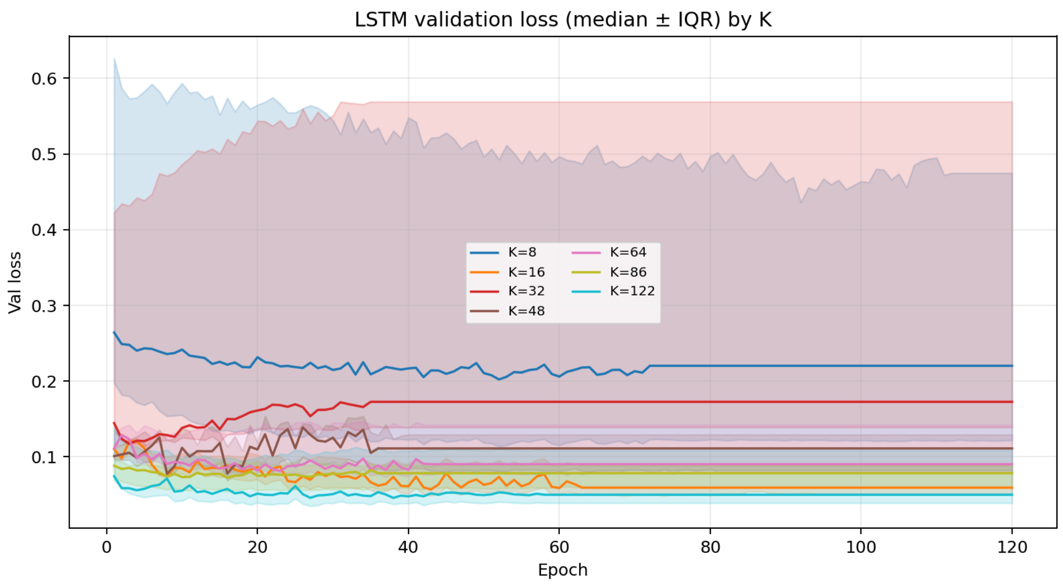

Figure 11: LSTM validation loss by K shown as median $\pm$ IQR across dataset seeds ($H = 100$, feature mode `full`).

Stage 2 (LSTM): residual error. 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.

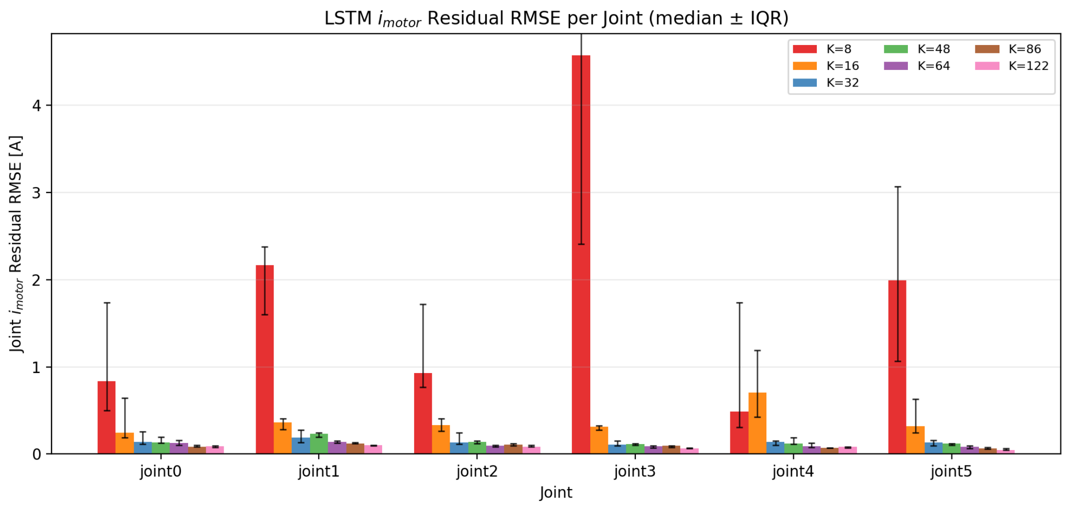

Figure 12: LSTM residual motor-current RMSE per joint (median $\pm$ IQR) for selected values of K ($H = 100$, feature mode `full`).

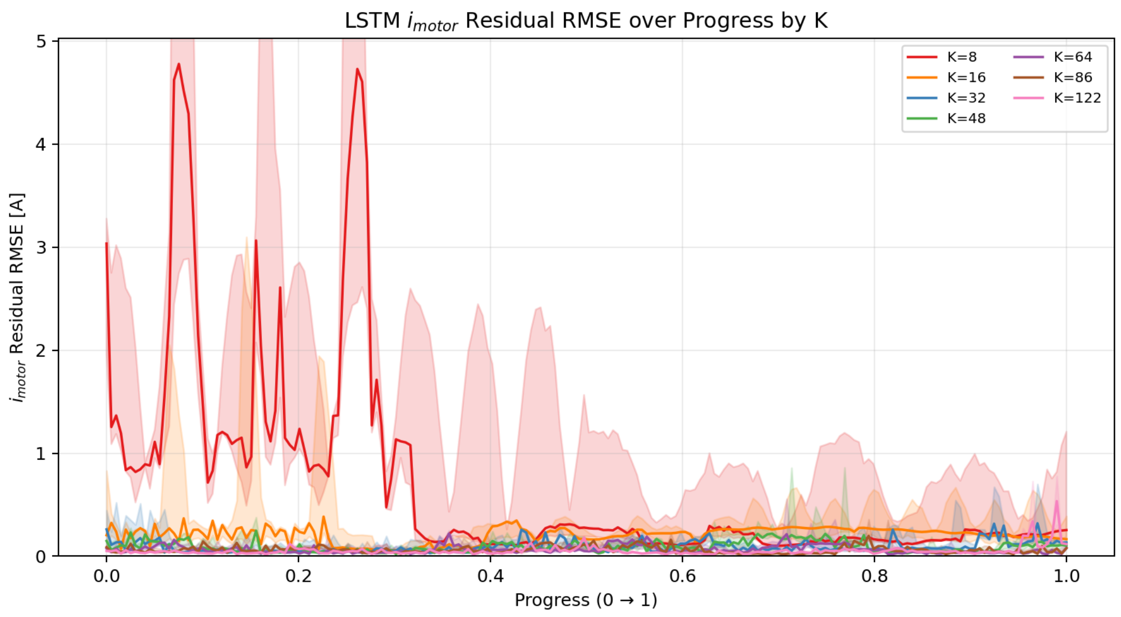

Figure 13: LSTM residual motor-current RMSE over normalised progress ($0 \to 1$) by K shown as median $\pm$ IQR (shaded bands) ($H = 100$, feature mode `full`).

End-to-end motor-current accuracy (per joint). 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.

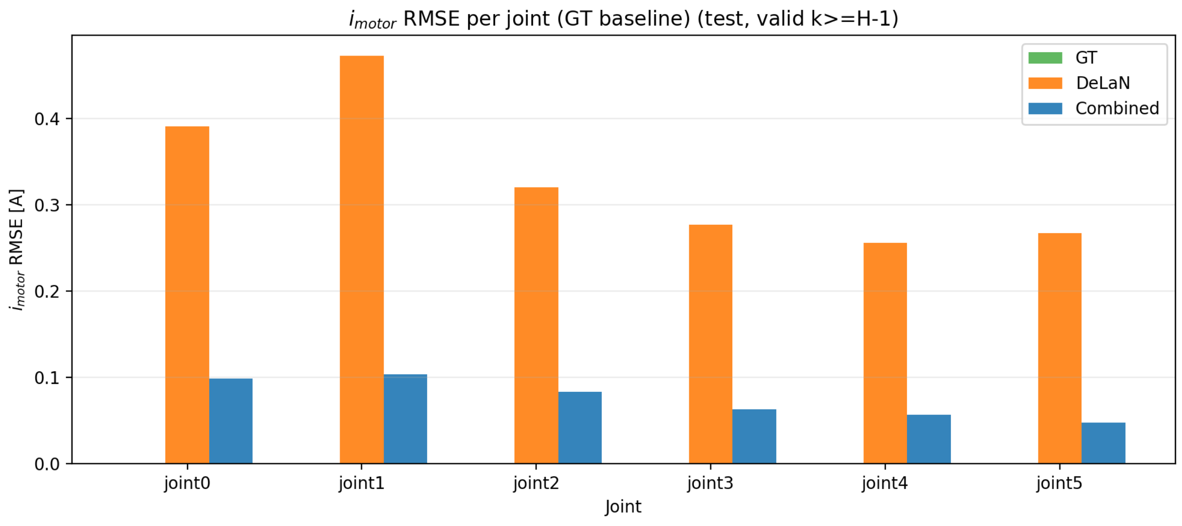

Figure 14: Representative per-joint motor-current RMSE on the test split comparing DeLaN and the combined DeLaN+LSTM predictor (valid indices $k \geq H - 1$, $H = 100$).

5.1.2 Best Model Approach Results

DeLaN Best Model Figures 15–18 summarise the DeLaN best-model sweep at fixed $K = 84$. The results are aggregated across dataset seeds with seed-wise aggregation across DeLaN initialisations.

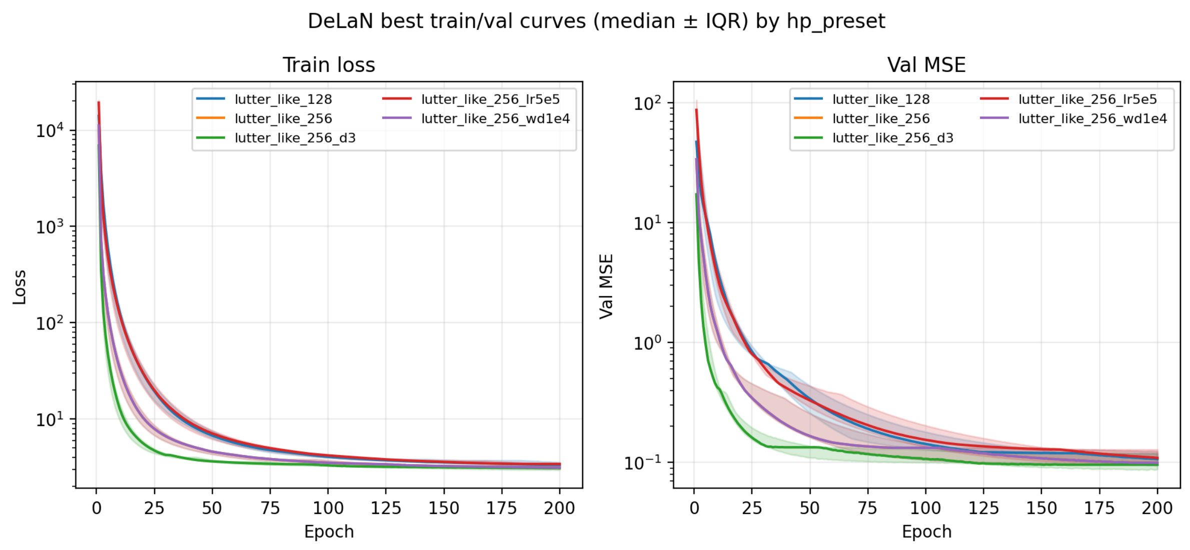

Figure 15: DeLaN training loss (left) and validation MSE in motor-current units (right) shown as median ± IQR across seeds for each hyperparameter preset.

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.

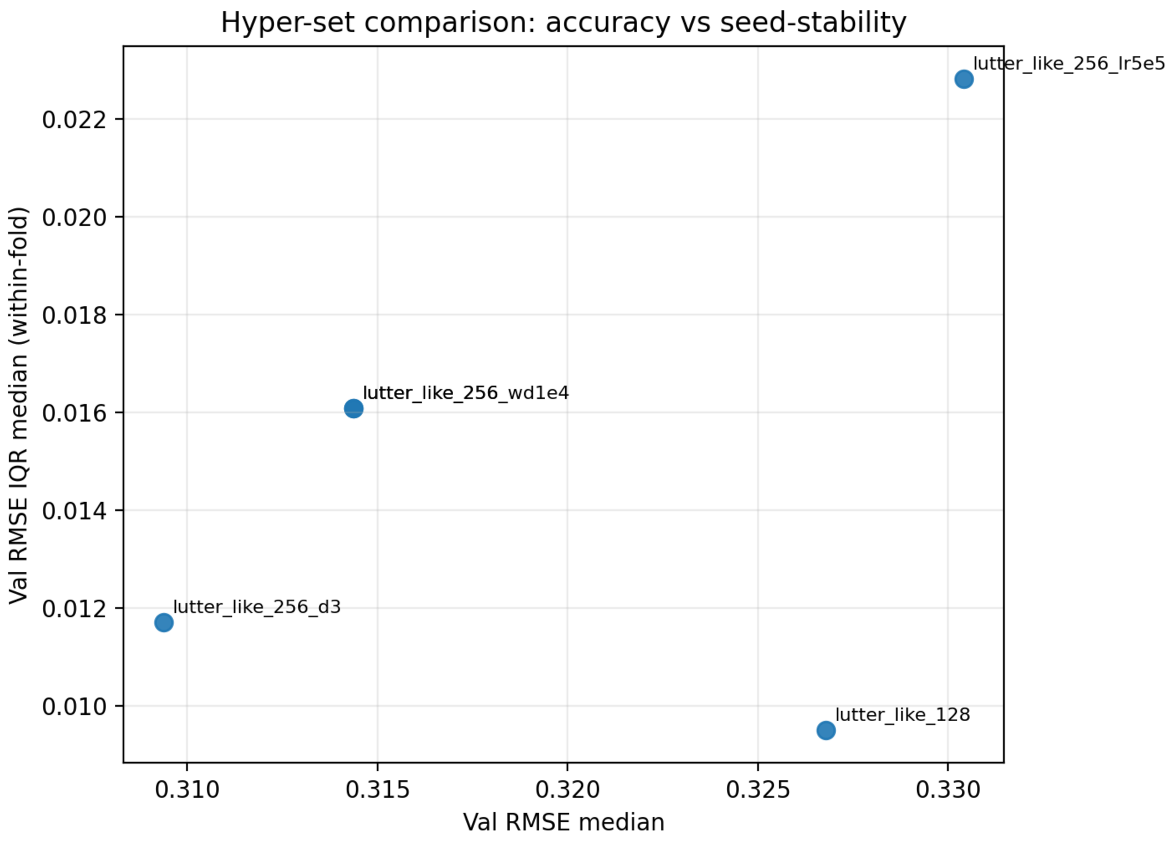

Figure 16: Hyperparameter comparison: median validation motor-current RMSE versus seed-stability measured by the median within-split IQR of validation RMSE (each point denotes one DeLaN hyperparameter preset; statistics aggregated across dataset seeds with seed-wise aggregation across DeLaN initialisations).

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–$0.330\,\mathrm{A}$ and a stability range of approximately 0.009–$0.023\,\mathrm{A}$.

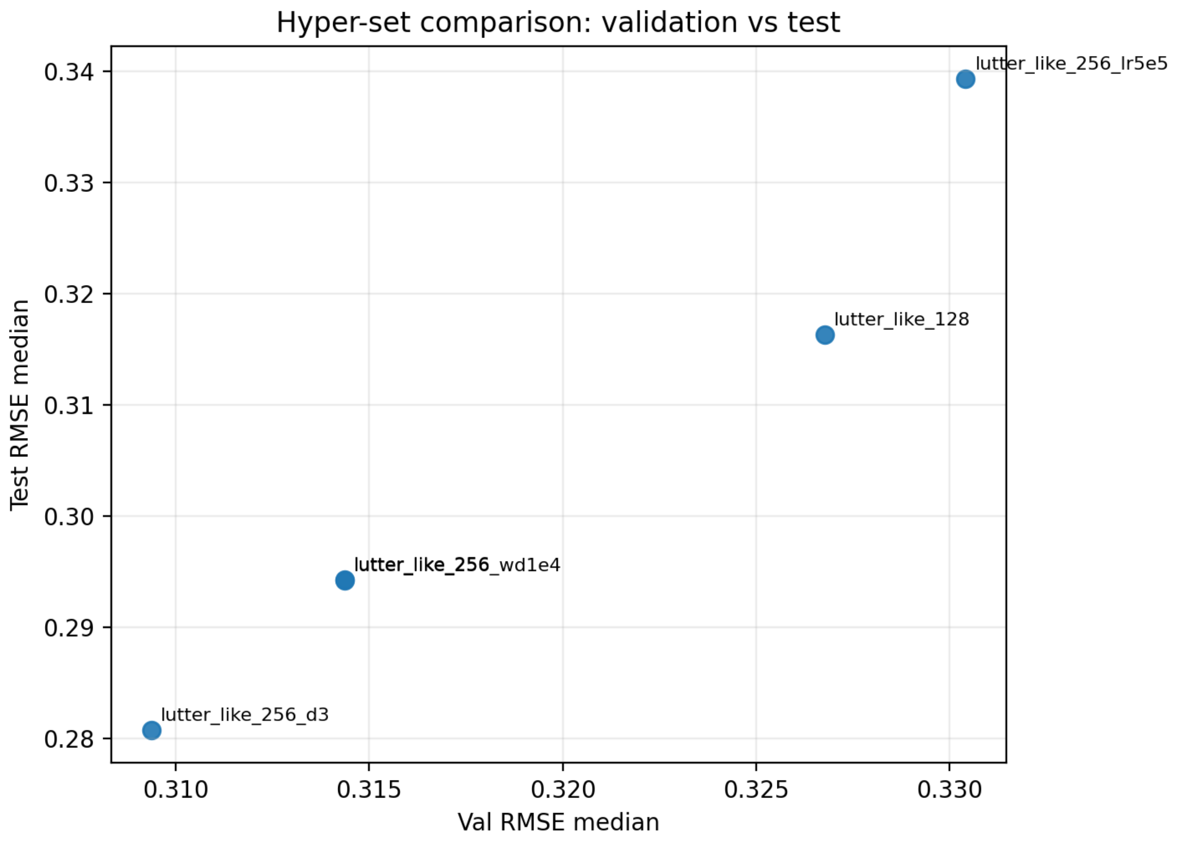

Figure 17: Hyperparameter comparison: median validation motor-current RMSE versus median test motor-current RMSE (each point denotes one DeLaN hyperparameter preset; statistics aggregated across dataset seeds with seed-wise aggregation across DeLaN initialisations).

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

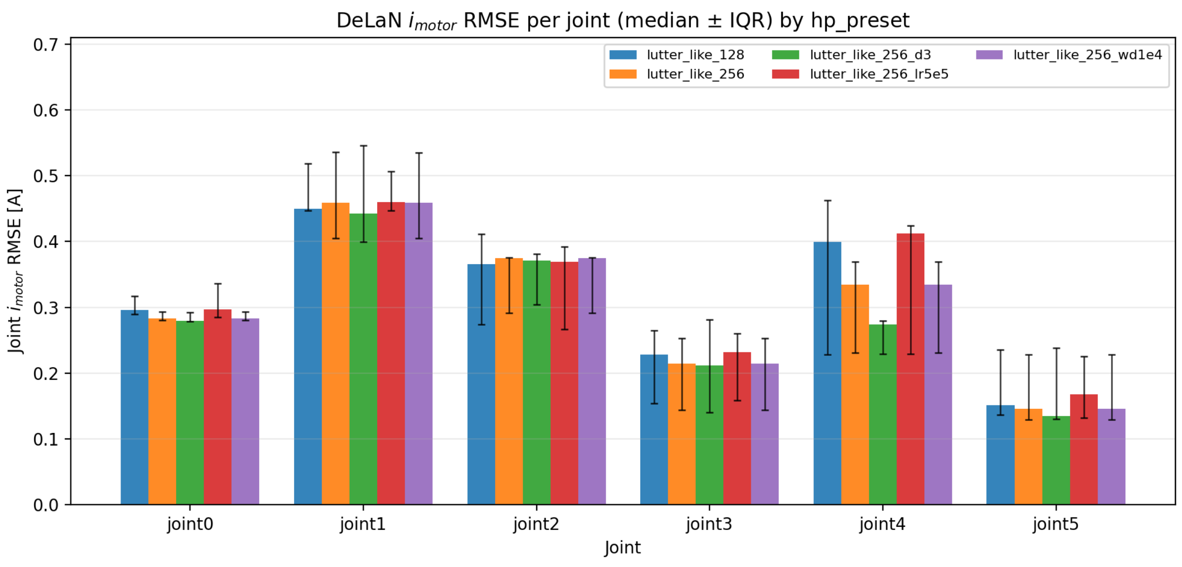

Figure 18: DeLaN motor-current RMSE per joint (median $\pm$ IQR) grouped by hyperparameter preset.

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.

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

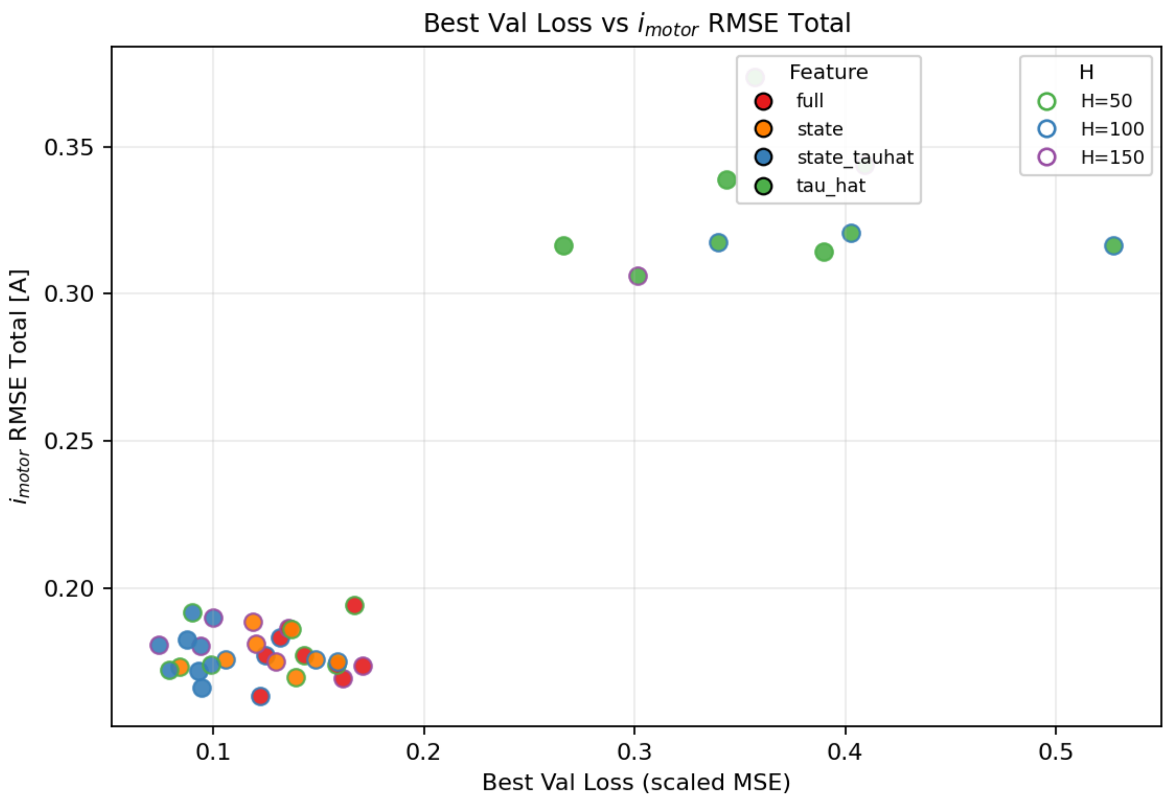

Figure 19: Best validation loss versus total motor-current RMSE on the test split for the LSTM residual model (marker colour denotes feature mode; marker outline denotes history length H).

Figure 19 shows that most configurations cluster at low test RMSE (approximately 0.165–$0.195\,\mathrm{A}$) 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–$0.34\,\mathrm{A}$.

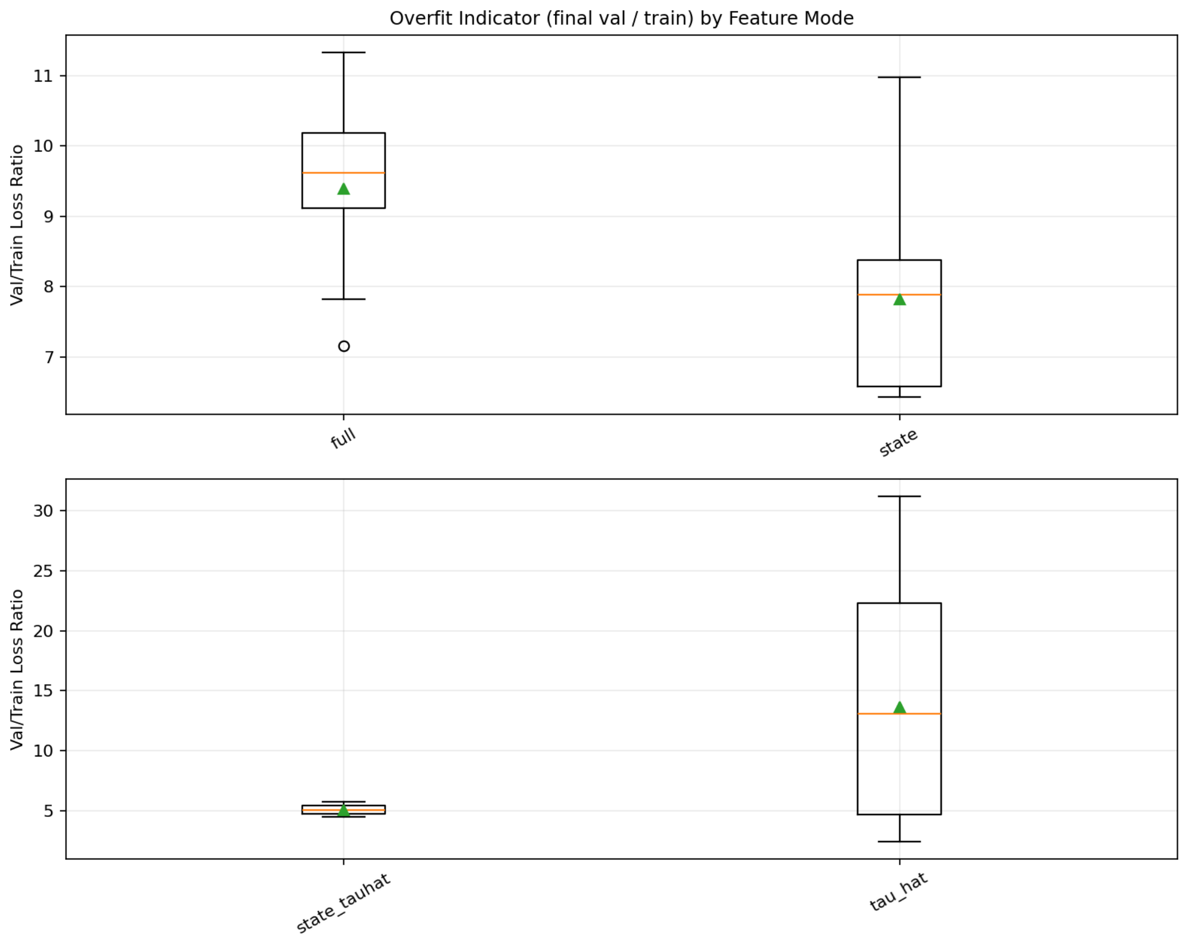

Figure 20: Overfit indicator (final validation loss / final training loss) by feature mode shown as boxplots across runs (green triangles indicate means; panels use different y-axis ranges for readability).

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.

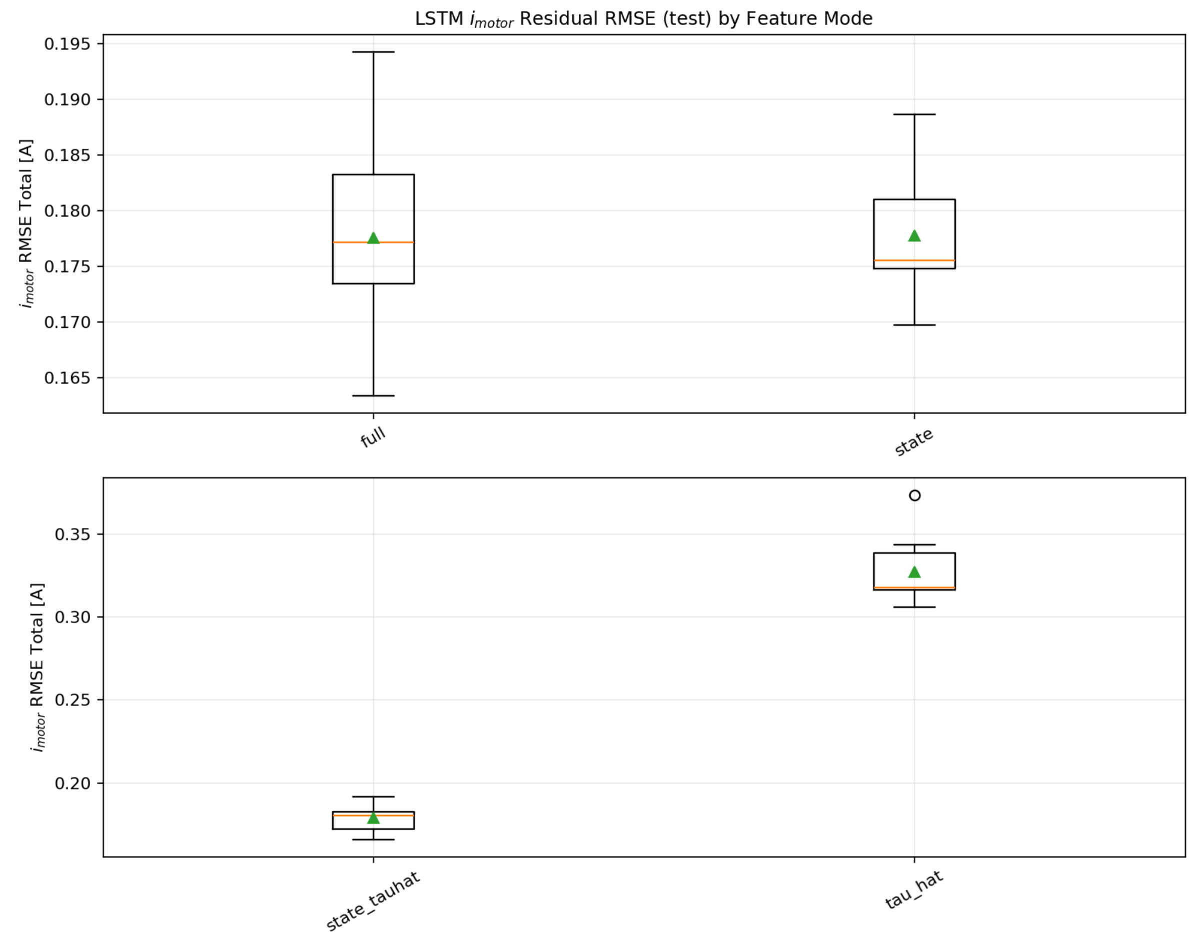

Figure 21: LSTM residual motor-current RMSE on the test split by feature mode shown as boxplots across runs (green triangles indicate means; panels use different y-axis ranges for readability).

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–$0.19\,\mathrm{A}$, whereas the high-range panel contains configurations with residual RMSE around 0.31–$0.37\,\mathrm{A}$.

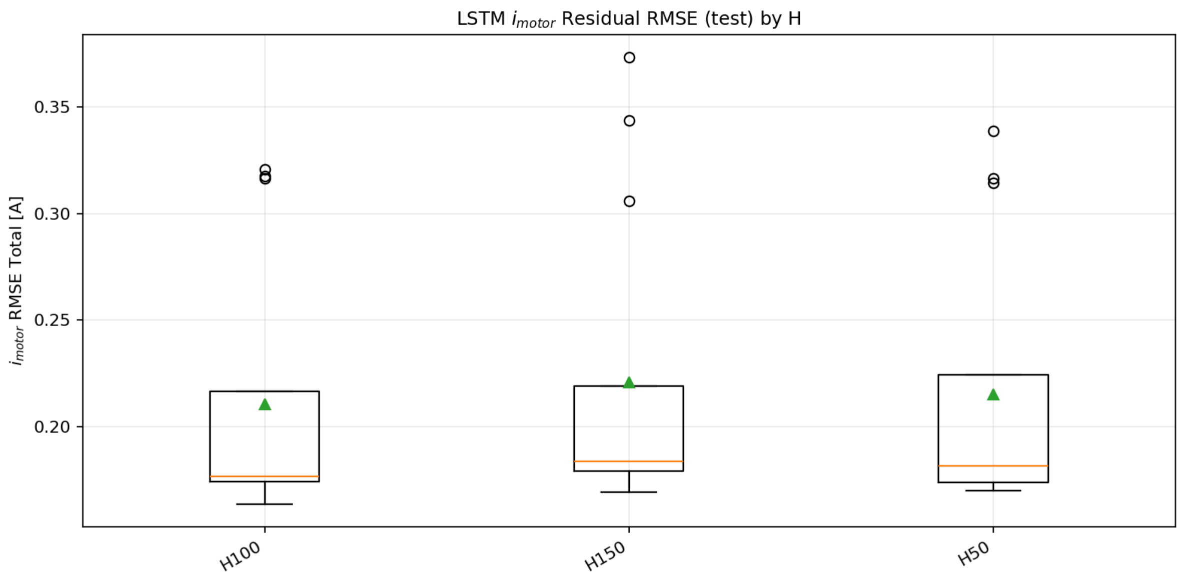

Figure 22: LSTM residual motor-current RMSE on the test split shown as boxplots across history lengths H (green triangles indicate means).

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.

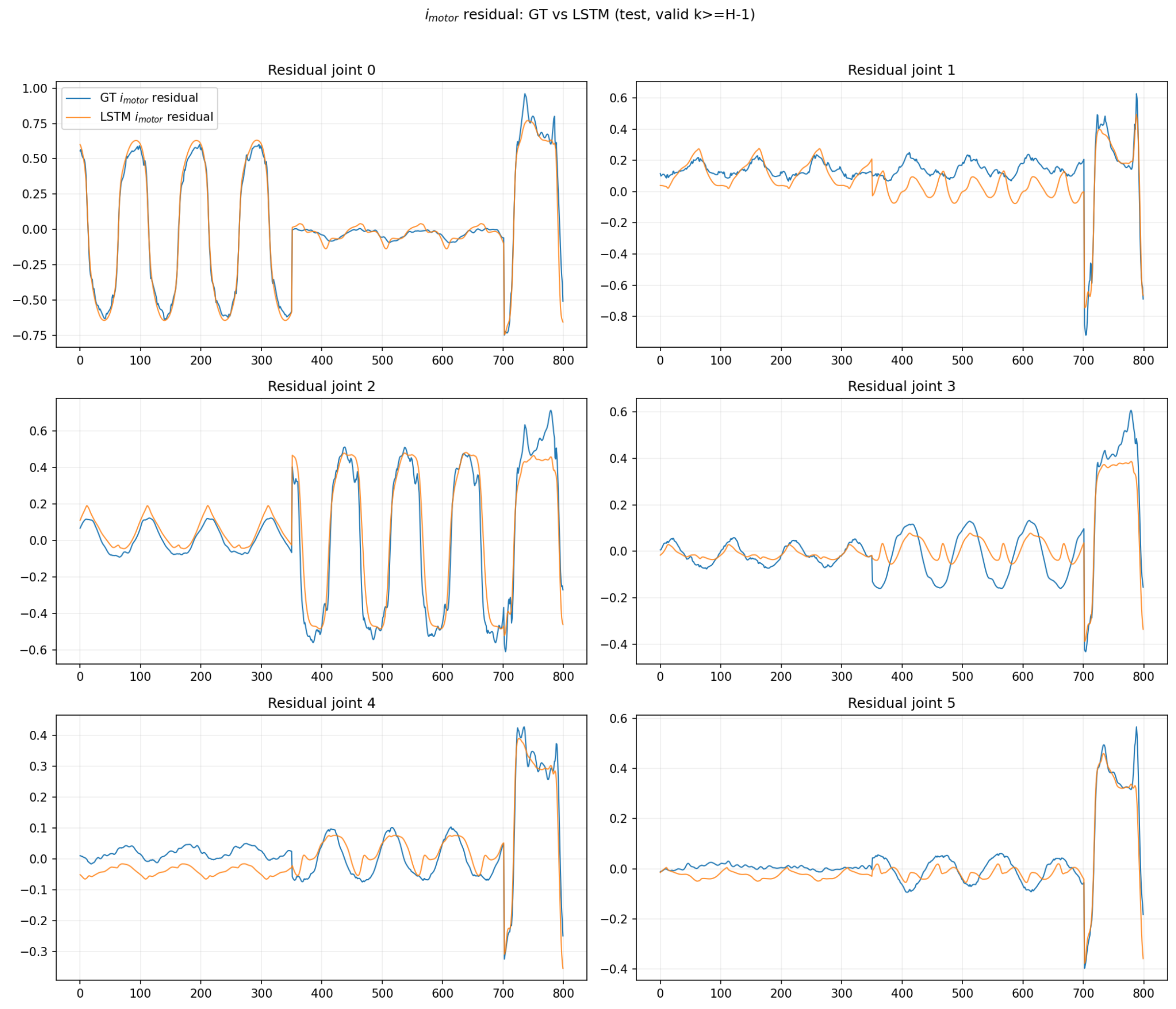

Figure 23: Representative residual motor-current traces: ground-truth residual versus LSTM residual prediction on the test split (valid indices $k \geq H - 1$, $H = 150$).

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

Best Pipeline

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 $0.46\,\mathrm{A}$, whereas the combined predictor ranges from approximately 0.07 to $0.25\,\mathrm{A}$. The largest RMSE is observed on joint 1 and the smallest on joint 5 for both models.

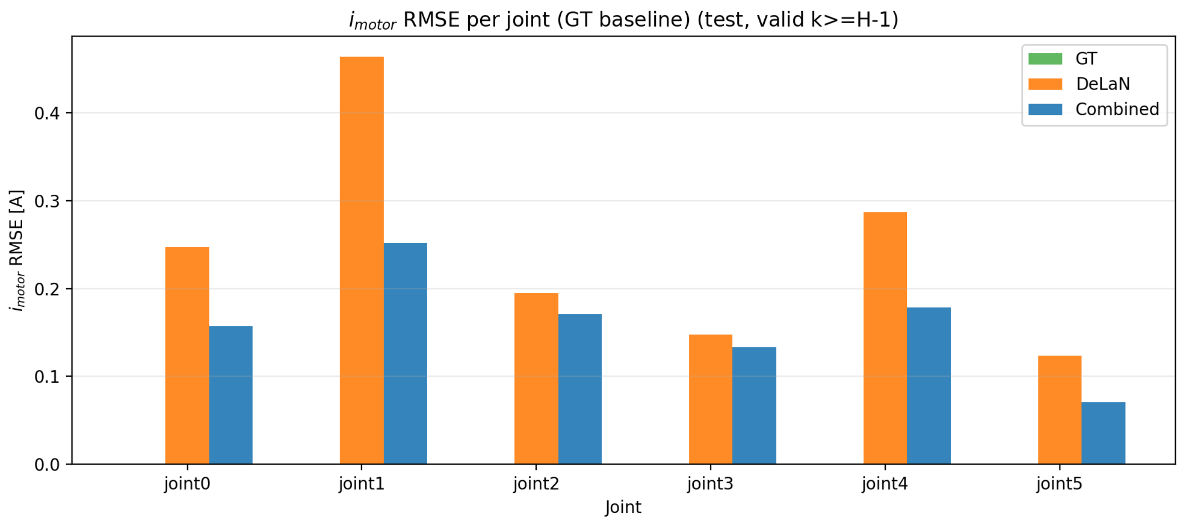

Figure 24: Best pipeline motor-current RMSE per joint on the test split comparing DeLaN and the combined DeLaN+LSTM predictor (valid indices $k \geq H - 1$, $H = 150$).

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.

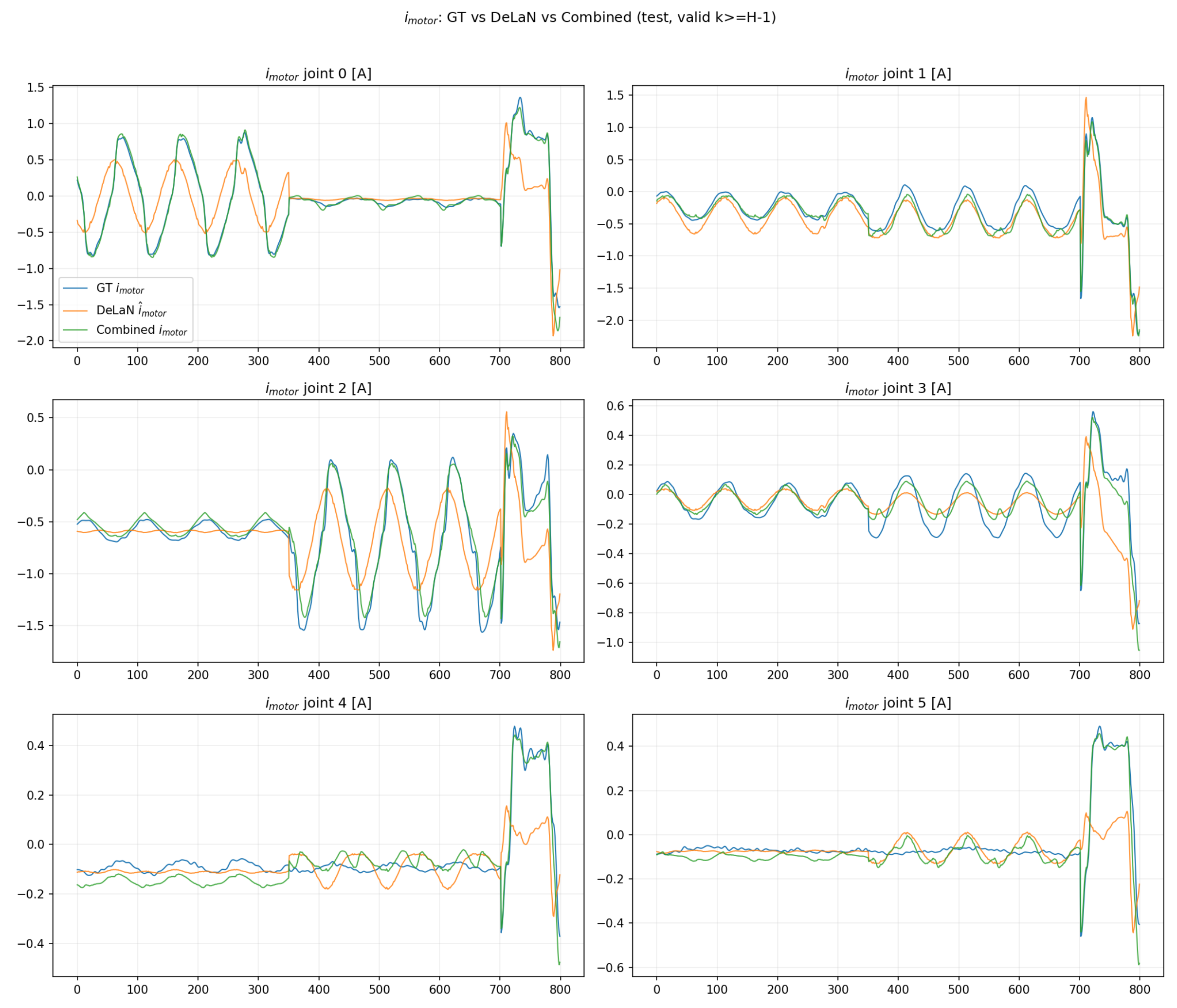

Figure 25: Representative motor-current overlay: ground truth, DeLaN prediction, and combined DeLaN+LSTM prediction on the test split (valid indices $k \geq H-1$, $H = 150$).

5.2 Performance Evaluation to Baseline

5.3 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 [42] and its accompanying IEEE DataPort release [41]. The baseline study likewise applies low-pass filtering to the recorded signals as part of its preprocessing [42]. 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:

$$\mathbf{K}_i = \left(\boldsymbol{\Theta}_{m,i}^{\top}\boldsymbol{\Theta}_{m,i}\right)^{-1}\boldsymbol{\Theta}_{m,i}^{\top}\mathbf{i}_{m,i}, \tag{59}$$

where $\boldsymbol{\Theta}_{m,i}$ is the measurement matrix of the regressor terms for joint i and $\mathbf{i}_{m,i}$ denotes the corresponding measured motor-current samples [42]. Evaluation is reported as per-joint

RMSE in current units,

$$\mathrm{RMSE}(i_i) = \sqrt{\frac{1}{n}\sum_{k=1}^{n}\left(i_{i,k}^{\mathrm{real}} - i_{i,k}^{\mathrm{pred}}\right)^2}. \tag{60}$$

Baseline results on IEEE DataPort dataset. Tables 2 and 3 reproduce the current-prediction RMSE reported by [42] for UR3e and UR10e, each without load and with load, for the provided training and testing datasets. Across conditions, UR3e baseline RMSE values are on the order of 0.06–$0.38\,\mathrm{A}$, while UR10e baseline RMSE values range up to $1.56\,\mathrm{A}$ on the test set.

Table 2: Baseline motor-current RMSE on the training dataset (Table I in [42]).

RMSE	i_1 [A]	%	i_2 [A]	%	i_3 [A]	%	i_4 [A]	%	i_5 [A]	%	i_6 [A]	%
UR3e without load	0.103	1.79	0.118	1.72	0.114	2.77	0.071	3.26	0.077	3.79	0.063	3.11
UR3e with load	0.101	1.76	0.182	2.49	0.154	3.43	0.136	5.32	0.092	4.50	0.071	3.47
UR10e without load	0.477	1.88	0.615	0.85	0.322	1.28	0.096	3.27	0.091	3.57	0.096	3.34
UR10e with load	0.444	1.38	0.692	0.96	0.338	1.34	0.104	3.35	0.089	3.47	0.087	3.02

Table 3: Baseline motor-current RMSE on the testing dataset (Table II in [42]).

RMSE	i_1 [A]	%	i_2 [A]	%	i_3 [A]	%	i_4 [A]	%	i_5 [A]	%	i_6 [A]	%
UR3e without load	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 with load	0.096	3.31	0.376	5.63	0.308	5.09	0.249	8.53	0.150	8.10	0.070	4.10
UR10e without load	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 with load	0.809	4.39	1.555	6.05	0.653	5.06	0.185	9.88	0.097	6.47	0.080	5.41

Our evaluation protocol and split. In addition to the dataset baseline, we report motor-current RMSE for the DeLaN and DeLaN+LSTM models produced by the thesis-internal model-selection procedure at fixed $K = 84$ trajectories and split sizes (train, val, test) = (59, 8, 17) trajectories. 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. Load-condition baselines are included as placeholders and will be filled once the best-model pipeline is re-run on the "with load" trajectories.

Table 4: DeLaN motor-current RMSE per joint on the held-out test split (17 trajectories, $K = 84$).

	i_1 [A]	i_2 [A]	i_3 [A]	i_4 [A]	i_5 [A]	i_6 [A]
UR3e without load	0.368	0.405	0.286	0.226	0.219	0.202
UR3e with load (*TBD*)	–	–	–	–	–	–

Table 5: DeLaN motor-current RMSE per joint on the validation split (8 trajectories, $K = 84$).

	i_1 [A]	i_2 [A]	i_3 [A]	i_4 [A]	i_5 [A]	i_6 [A]
UR3e without load	0.408	0.484	0.331	0.284	0.273	0.257
UR3e with load (*TBD*)	–	–	–	–	–	–

Table 6: Best-model DeLaN+LSTM pipeline motor-current RMSE per joint on the validation split (8 trajectories, valid indices $k \geq H - 1$).

	i_1 [A]	i_2 [A]	i_3 [A]	i_4 [A]	i_5 [A]	i_6 [A]
UR3e without load	0.069	0.083	0.124	0.063	0.086	0.052
UR3e with load (*TBD*)	–	–	–	–	–	–

Reporting notes. Direct numerical comparison to [42] is not one-to-one, since the baseline is identified on the dataset-provided training split (50k samples) and evaluated on the dataset-provided test split (5k samples), whereas our results are obtained from the thesis-specific trajectory-wise split (59/8/17 trajectories) and include the LSTM warm-up constraint $k \geq H - 1$. Accordingly, Tables 4–6 report results under the thesis-specific split and evaluation constraints.

5.3 DeLaN+LSTM Gripper Compensation

Best-model evaluation on "with load" trajectories. Following the best-model selection procedure, we evaluate the final DeLaN+LSTM pipeline on the "with load" trajectories of the IEEE DataPort dataset [41] and report the corresponding motor-current RMSE metrics. The corresponding best-model metrics are reported in Table 7 and Table 8 (placeholders shown until the best-model run on the loaded dataset is completed).

Table 7: Best-model DeLaN motor-current RMSE per joint for gripper compensation on dataset [41] (placeholders).

Condition	i_1 [A]	i_2 [A]	i_3 [A]	i_4 [A]	i_5 [A]	i_6 [A]
UR3e without load (*TBD*)	–	–	–	–	–	–
UR3e with load (*TBD*)	–	–	–	–	–	–

Table 8: Best-model DeLaN+LSTM pipeline motor-current RMSE per joint for gripper compensation on dataset [41] (placeholders).

Condition	i_1 [A]	i_2 [A]	i_3 [A]	i_4 [A]	i_5 [A]	i_6 [A]
UR3e without load (*TBD*)	–	–	–	–	–	–
UR3e with load (*TBD*)	–	–	–	–	–	–

6 Implementation

Chapter overview. 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`) to ensure reproducibility and to provide a consistent GPU-enabled runtime for both Stage 1 (DeLaN) and Stage 2 (residual LSTM). In the following, the actuation signal is the measured motor current $\mathbf{i}_{\mathrm{motor}}$ in A. For compatibility with the upstream DeLaN codebase [reference Lutter github], some internal variable names use the legacy identifier `tau`; throughout the implementation, `tau` should be interpreted as $\mathbf{i}_{\mathrm{motor}}$.

6.1 Execution Environment (Docker)

Containerised stack. 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`), whereas the learning stages run in GPU-enabled containers. Stage 1 is implemented in JAX (Haiku/Optax) 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`) such that sweeps can run unattended. Across services, `numpy` 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.

Artefact management. 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

Preprocessing. 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 (DeLaN) and residual export. 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 (residual LSTM). 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.

Combined evaluation. 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)

Sweep controller. 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.

Sweep variants. 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

K-Domination. The K-domination study (Section 5.1.1) quantifies how the number of available demonstration trajectories influences the two-stage pipeline. Across all reported metrics, increasing K reduces both (i) the typical error level and (ii) the variability across dataset seeds, indicating improved generalisation and robustness with more diverse motion coverage.

Stage 1 (DeLaN): learning dynamics and split sensitivity. 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.

Stage 1 (DeLaN): error over motion progress and per-joint balance. 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.

Stage 2 (LSTM): dependence on K and coupling to Stage 1. 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.

End-to-end implication. 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.

8 Summary and Outlook

The results in this thesis can be interpreted as a first step towards a more general notion of dynamic awareness for collaborative manipulators. Several conceptual implications and natural extensions are worth highlighting.

Online motor-current awareness for safety–critical control. Once the DeLaN+LSTM architecture has been trained and deployed, the robot possesses an accurate estimate of its joint motor currents $\hat{\mathbf{i}}_{\mathrm{RG},k}$ at every time step and—via the known conversion factor k_t and the Jacobian—of the corresponding end–effector wrench $\hat{\vec{F}}_{\mathrm{RG},k}$ in the measurement frame. In principle, this enables model–predictive controllers and safety monitors to operate directly on end–effector wrench limits, for example those prescribed for collaborative operation, rather than relying on conservative joint–space bounds. By exploiting the available actuation margin more tightly, such a controller could realise higher velocity and acceleration profiles while remaining within certified interaction–force constraints.

Beyond joint space: evaluation of end–effector wrench prediction. The present thesis restricts training and quantitative evaluation to joint space: the DeLaN+LSTM model is trained on joint states and motor currents, and its quality is assessed primarily via motor-current prediction errors. A natural next step is a systematic evaluation of end–effector wrench prediction against ground–truth FT data at the flange. One promising route is to combine IsaacSim–based data generation (with and without domain randomisation and sensor noise) with real–robot experiments using the same excitation families as in this work. This would allow a comprehensive comparison between (i) models trained purely in simulation and tested on the real robot, (ii) models trained on real data and tested on held–out real trajectories, and (iii) hybrid training schemes that mix simulated and measured data. Such an evaluation pipeline would provide strong evidence for the robustness of the joint–space training strategy when viewed in the measurement frame. Once this consistency between Jacobian–mapped model predictions and ground–truth flange wrenches has been demonstrated, the same DeLaN+LSTM architecture can, in principle, be trained and deployed using proprioceptive data alone, without requiring an FT sensor in the loop.

Tool/gripper calibration as a third learning stage. In the current experiments, the LSTM residual model in Stage 2 reduces the motor-current error on the nominal robot–only configuration to near zero. When a fixed gripper is mounted, one could either re–record the entire dataset with the gripper attached or, more interestingly, treat the gripper as an additional structured disturbance to be learned. Conceptually, this suggests a third learning stage in which a small network (e.g. another LSTM) takes the residual error of Stage 2 as input and learns a tool–specific compensation term. From a closed–loop viewpoint, the robot would then iteratively refine its internal model when a new tool is attached, realising a form of transfer learning for tool calibration without rebuilding the entire inverse–dynamics model from scratch.

Towards payload identification from residual wrenches. Once tool/gripper effects are compensated, remaining discrepancies between the nominal robot–gripper wrench model and the measured flange wrench can be attributed primarily to external interaction and payload dynamics. Mapping the Stage 2/Stage 3 residuals into the end–effector frame via the Jacobian (after applying k_t) yields an estimate of the interaction wrench along and around the $x/y/z$ axes. In combination with the corresponding end–effector linear and angular velocities and accelerations, this opens the door to a further stage in which a dedicated estimator recovers the payload dynamic parameters $\boldsymbol{\phi}_{\text{payload}} = (m, \mathbf{c}, \mathbf{I}) \in \mathbb{R}^{10}$ from these residual wrenches in a Newton–Euler framework. In other words, the same DeLaN+LSTM backbone that currently provides accurate joint–space prediction could serve, after appropriate extensions, as the core of an online PDPI pipeline that incrementally refines the effective rigid–body model (robot + tool + payload) during everyday manipulation tasks.

Interpretable sparse models via SINDy. The DeLaN+LSTM architecture used in this thesis is deliberately expressive but remains a largely black–box regression model: the Lagrangian network and the recurrent residual model jointly fit the data, yet they do not expose which physical effects are actually dominant in the identified dynamics. A complementary avenue for future work is to apply sparse identification of nonlinear dynamics (SINDy)to the same joint–space data. In such a setting, the candidate function library could explicitly include standard rigid–body inverse-dynamics terms (e.g. Newton–Euler and Lagrangian terms) expressed in current units, nonlinear Coriolis and gravity terms, and parametric friction models, together with a small number of additional generic nonlinear features. SINDy would then select a parsimonious subset of these terms that best explains the observed motor currents, potentially yielding an interpretable model in which a few physically meaningful contributions account for most of the variance in the data. This would provide insight into which dynamic effects the robot actually relies on in practice, and would offer a transparent counterpart to the high–capacity DeLaN+LSTM model used in the present work.

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

[5] A. Kurdas, M. Hamad, J. Vorndamme, N. Mansfeld, S. Abdolshah, and S. Haddadin, “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

[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

[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

[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 [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

[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

[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. 13 871–13 883, 2025. DOI: 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

[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 [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 [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

[16] 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

[17] M. Lutter and J. Peters, Combining physics and deep learning to learn continuous-time dynamics models, 2023. arXiv: 2110.01894 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2110.01894

[18] 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

[19] 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

[20] 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

[21] 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

[22] Y. Hu, W. Li, Y. Zhou, and D. T. Pham, "Improved deep lagragian network-enabled momentum observer for collision detection during human-robot collaboration," Robotics and Computer-Integrated Manufacturing, vol. 97, p. 103 093, 2026, ISSN: 0736-5845. DOI: https://doi.org/10.1016/j.rcim.2025.103093 [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0736584525001474

[23] M. Lahoud, G. Marchello, M. D'Imperio, A. Müller, and F. Cannella, "A deep learning framework for non-symmetrical coulomb friction identification of robotic manipulators," in 2024 IEEE International Conference on Robotics and Automation (ICRA), 2024, pp. 10 510–10 516. DOI: 10.1109/ICRA57147.2024.10610737

[24] 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

[25] 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. 102 210, 2021, ISSN: 0736-5845. DOI: https://doi.org/10.1016/j.rcim.2021.102210 [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0736584521000934

[26] 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

[27] 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

[28] 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 [Online]. Available: https://www.mdpi.com/2075-1702/11/2/316

[29] 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 [Online]. Available: https://doi.org/10.1017/S026357472400105X

[30] 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. 102 168, 2021, ISSN: 0736-5845. DOI: 10.1016/j.rcim.2021.102168 [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0736584521000521

[31] 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

[32] 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

[33] 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

[34] 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

[35] 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. 105 689, 2023, ISSN: 0952-1976. DOI: 10.1016/j.engappai.2022.105689 [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197622006790

[36] 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

[37] 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

[38] 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

[39] 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. 1 470 163, 2024, ISSN: 2296-9144. DOI: 10.3389/frobt.2024.1470163 [Online]. Available: https://www.frontiersin.org/articles/10.3389/frobt.2024.1470163/full

[40] 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

[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] 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

List of Figures

List of Tables

List of source codes

A Kinematic and Dynamic Background of Robot Manipulation and Environment Interaction

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

$$\vec{F}_{\text{ext}} = \begin{bmatrix} \mathbf{f} \\ \boldsymbol{\tau} \end{bmatrix} \in \mathbb{R}^6, \tag{61}$$

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 $\boldsymbol{\phi}_{\text{eff}}$ (mass, CoM and inertia) moving with linear and angular motion $(\mathbf{a}, \boldsymbol{\alpha}, \boldsymbol{\omega})$, the Newton-Euler equations (cf. (2)) give

$$\begin{bmatrix} \mathbf{f} \\ \boldsymbol{\tau} \end{bmatrix} = m \begin{bmatrix} \mathbf{I} & -[\mathbf{c}]^{\times} \\ [\mathbf{c}]^{\times} & \mathbf{J}_s \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \boldsymbol{\alpha} \end{bmatrix} + \begin{bmatrix} m[\boldsymbol{\omega}]^{\times}[\boldsymbol{\omega}]^{\times}\mathbf{c} \\ [\boldsymbol{\omega}]^{\times}\mathbf{J}_s\boldsymbol{\omega} \end{bmatrix}. \tag{62}$$

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

$$\mathbf{f} = m\mathbf{a} - m[\mathbf{c}]^{\times}\boldsymbol{\alpha} + m[\boldsymbol{\omega}]^{\times}[\boldsymbol{\omega}]^{\times}\mathbf{c}, \tag{63}$$

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

$$\boldsymbol{\tau} = m[\mathbf{c}]^{\times}\mathbf{a} + \mathbf{J}_s\boldsymbol{\alpha} + [\boldsymbol{\omega}]^{\times}\mathbf{J}_s\boldsymbol{\omega}, \tag{64}$$

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

$$\vec{F}_{\text{ext}} = \vec{F}_{\text{dyn}}(\mathbf{a}, \boldsymbol{\alpha}, \boldsymbol{\omega}; \boldsymbol{\phi}_{\text{eff}}) = Y(\mathbf{a}, \boldsymbol{\alpha}, \boldsymbol{\omega})\,\boldsymbol{\phi}_{\text{eff}}, \tag{65}$$

where $Y(\cdot)$ is a 6×10 regressor matrix that is linear in $\boldsymbol{\phi}_{\text{eff}}$ but nonlinear in the motion variables. Hence, $\vec{F}_{\text{ext}}$ is not simply $m\mathbf{a}$, nor can it be written as $\boldsymbol{\phi}_{\text{eff}}\ddot{\mathbf{q}}$; the mapping from joint accelerations $\ddot{\mathbf{q}}$ to $\vec{F}_{\text{ext}}$ 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

$$\boldsymbol{\tau}_{\text{ext}} = {}^{S}J(\mathbf{q})^{\top}\,\vec{F}_{\text{ext}}, \tag{66}$$

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

$$\boldsymbol{\tau}_{\text{ext}} = \mathbf{J}^{T}(\mathbf{q})\,Y(\mathbf{a}, \boldsymbol{\alpha}, \boldsymbol{\omega})\,\boldsymbol{\phi}_{\text{eff}}, \tag{67}$$

which makes explicit that $\boldsymbol{\tau}_{\text{ext}}$ is linear in $\boldsymbol{\phi}_{\text{eff}}$, 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 $C_1, \ldots, C_5$ 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

C_2: **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)

C_3: **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

C_4: **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 9: 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	[25]	2021	ScienceDirect	34
R5	[6]	2025	IEEE	0
R6	[26]	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	[27]	2022	IEEE	57
R13	[15]	2007	IEEE	303
R14	[8]	2022	MPDI	17
R15	[28]	2023	MPDI	16
R16	[29]	2024	Cambridge	2
R17	[30]	2021	ScienceDirect	100
R18	[31]	2022	IEEE	3
R19	[33]	2024	IEEE	19
R20	[32]	2022	IEEE	2
R21	[34]	2023	ScienceDirect	2
R22	[20]	2025	Cambridge	1
R23	[10]	2023	IEEE	3
R24	[21]	2021	MPDI	10
R25	[35]	2023	ScienceDirect	9
R26	[36]	2021	IEEE	5
R27	[37]	2024	IEEE	1
R28	[38]	2024	IEEE	13
R29	[39]	2024	Frontiersin	0
R30	[16]	2025	IEEE	0
R31	[17]	2023	ArXiv	-
R32	[18]	2025	IEEE	0

D DeLaN parameterisation and friction model

This appendix details the specific neural parameterisation of the Deep Lagrangian Network (DeLaN) used in Stage 1 of the proposed architecture, i.e. the construction of the inertia, potential/gravity and friction subnetworks that together implement the inverse-dynamics mapping $f_{\text{DeLaN}}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}; \boldsymbol{\theta}, \boldsymbol{\psi})$ described in Section 3.

D.1 Inertia subnetwork

Following the improved DeLaN formulation of [22] and its PINN-based extension to industrial robots in [19], the symmetric positive-definite inertia matrix $\mathbf{M}_{\boldsymbol{\theta}}(\mathbf{q}) \in \mathbb{R}^{n\times n}$ is obtained from a learned Cholesky factor. We parameterise a lower-triangular matrix $\hat{\mathbf{L}}(\mathbf{q}; \boldsymbol{\theta}_M)$ and set

$$\mathbf{M}_{\boldsymbol{\theta}}(\mathbf{q}) = \hat{\mathbf{L}}(\mathbf{q}; \boldsymbol{\theta}_M)\, \hat{\mathbf{L}}(\mathbf{q}; \boldsymbol{\theta}_M)^{\top}. \tag{68}$$

The factor $\hat{\mathbf{L}}$ is decomposed into a strictly lower-triangular part and a diagonal part,

$$\hat{\mathbf{L}}(\mathbf{q}; \boldsymbol{\theta}_M) = \mathbf{L}_o(\mathbf{q}; \boldsymbol{\theta}_o) + \mathbf{L}_d(\mathbf{q}; \boldsymbol{\theta}_d), \tag{69}$$

where both $\mathbf{L}_o$ and $\mathbf{L}_d$ are represented by multilayer perceptrons (MLPs) taking $\mathbf{q}$ as input:

- $\mathbf{L}_o(\mathbf{q}; \boldsymbol{\theta}_o)$ is lower-triangular with zero diagonal entries and uses a linear output layer.
- $\mathbf{L}_d(\mathbf{q}; \boldsymbol{\theta}_d)$ is diagonal. Its diagonal elements are obtained as

  $$[\mathbf{L}_d]_{ii} = \mathrm{ReLU}\big(h_i(\mathbf{q})\big) + \varepsilon,$$

  where h_i is the i-th output of an MLP and $\varepsilon > 0$ is a small constant. The ReLU+offset guarantees strictly positive diagonal entries and hence $\mathbf{M}_{\boldsymbol{\theta}}(\mathbf{q}) \succ 0$ for all $\mathbf{q}$.

In practice, all MLPs in the DeLaN core (the two inertia subnetworks and the potential network below) use sinusoidal activation functions as suggested in [19, 22], since for revolute manipulators the elements of $\mathbf{M}(\mathbf{q})$ can be written as linear combinations of $\sin(\cdot)$ and $\cos(\cdot)$.

D.2 Potential and gravity subnetworks

The conservative part of the dynamics is encoded by a learned Lagrangian

$$\mathcal{L}_{\boldsymbol{\theta}}(\mathbf{q}, \dot{\mathbf{q}}) = \tfrac{1}{2}\, \dot{\mathbf{q}}^{\top} \mathbf{M}_{\boldsymbol{\theta}}(\mathbf{q})\, \dot{\mathbf{q}} - V_{\boldsymbol{\theta}_V}(\mathbf{q}), \tag{70}$$

where $V_{\boldsymbol{\theta}_V}(\mathbf{q})$ is represented by another MLP (the "potential" subnetwork) with parameters $\boldsymbol{\theta}_V$. Using automatic differentiation we obtain the gravity term as the gradient of the potential,

$$\mathbf{G}_{\boldsymbol{\theta}}(\mathbf{q}) = \frac{\partial V_{\boldsymbol{\theta}_V}(\mathbf{q})}{\partial \mathbf{q}} \approx \mathbf{g}(\mathbf{q}). \tag{71}$$

This DeLaN-style representation of the potential and its gradient follows the improved formulations proposed in [19, 22], where the potential is modelled by a neural network and the gravity vector is obtained as its configuration-space gradient.

Together with the inertia matrix (68), the conservative torques are given by

$$\boldsymbol{\tau}_{\text{cons}}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}; \boldsymbol{\theta}) = \mathbf{M}_{\boldsymbol{\theta}}(\mathbf{q})\,\ddot{\mathbf{q}} + \mathbf{C}_{\boldsymbol{\theta}}(\mathbf{q}, \dot{\mathbf{q}})\,\dot{\mathbf{q}} + \mathbf{G}_{\boldsymbol{\theta}}(\mathbf{q}), \tag{72}$$

with the Coriolis/centrifugal term $\mathbf{C}_{\boldsymbol{\theta}}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$ obtained from $\mathcal{L}_{\boldsymbol{\theta}}$ via the Euler–Lagrange equations.

D.3 Friction subnetwork

Joint friction and other non-conservative effects are modelled in a physically interpretable way by a Coulomb-viscous law, following the improved DeLaN formulation in [22] and empirical studies on UR robots with Coulomb-viscous friction [23]. For each joint i we use

$$\tau_{\text{fric},i}(\dot{q}_i; \boldsymbol{\psi}) = f_{c,i}\,\text{sgn}(\dot{q}_i) + f_{v,i}\,\dot{q}_i, \tag{73}$$

where $f_{c,i}$ and $f_{v,i}$ are the Coulomb and viscous friction coefficients for joint i. Collecting these in vectors $\mathbf{f}_c, \mathbf{f}_v \in \mathbb{R}^n$ yields the compact vector form

$$\boldsymbol{\tau}_{\text{fric}}(\dot{\mathbf{q}}; \boldsymbol{\psi}) = \mathbf{f}_c\,\text{sgn}(\dot{\mathbf{q}}) + \mathbf{f}_v\,\dot{\mathbf{q}}, \tag{74}$$

with $\text{sgn}(\cdot)$ applied element-wise. This can be interpreted as a joint-wise affine map

$$\boldsymbol{\tau}_{\text{fric}} = f_{\text{fric}}\big([\dot{\mathbf{q}}, \text{sgn}(\dot{\mathbf{q}})]; \boldsymbol{\psi}\big),$$

consistent with the main Methods section.

Equation (74) corresponds directly to the friction term in [22], where $\tau_f = f_c\,\text{sgn}(\dot{q}) + f_v\dot{q}$ is added to the DeLaN prediction.

D.4 Resulting DeLaN inverse-dynamics map

Combining the inertia, potential/gravity and friction subnetworks, the DeLaN inverse-dynamics model used in this thesis is

$$\hat{\boldsymbol{\tau}}_{\text{DeLaN}}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}; \boldsymbol{\theta}, \boldsymbol{\psi}) = \boldsymbol{\tau}_{\text{cons}}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}; \boldsymbol{\theta}) + \boldsymbol{\tau}_{\text{fric}}(\dot{\mathbf{q}}; \boldsymbol{\psi}) = f_{\text{DeLaN}}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}; \boldsymbol{\theta}, \boldsymbol{\psi}), \tag{75}$$

which is trained in Stage 1 by minimising the joint-space loss

$$\mathcal{L}_{\mathrm{DeLaN}}(\boldsymbol{\theta}, \boldsymbol{\psi}) = \frac{1}{N} \sum_{k=1}^{N} \| f_{\mathrm{DeLaN}}(\mathbf{q}_k, \dot{\mathbf{q}}_k, \ddot{\mathbf{q}}_k; \boldsymbol{\theta}, \boldsymbol{\psi}) - \boldsymbol{\tau}_{\mathrm{motor},k} \|_2^2 \,, \tag{76}$$

with $\boldsymbol{\tau}_{\mathrm{motor},k} = k_t \boldsymbol{I}_k$ as defined in (6). The trained DeLaN then serves as the nominal inverse-dynamics backbone that is subsequently refined by the residual LSTM in Stage 2, as described in Section 3.