CHALMERS
UNIVERSITY OF TECHNOLOGY

Data-Driven Plant Models for Hydraulic Excavators

Development of Compact Machine Learning Architectures across Simulated and Physical Platforms

Master's thesis in Systems, Control and Mechatronics

OSKAR HAAPALO, ADAM OSKARSSON

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's thesis in Systems, Control and Mechatronics

Data-Driven Plant Models for Hydraulic Excavators

Development of Compact Machine Learning Architectures across Simulated and Physical Platforms

OSKAR HAAPALO
ADAM OSKARSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology
Gothenburg, Sweden 2026

Data-Driven Plant Models for Hydraulic Excavators
Development of Compact Machine Learning Architectures across Simulated and Physical Platforms
OSKAR HAAPALO, ADAM OSKARSSON

Supervisor: Marcus Carlsson, CPAC Systems AB
Examiner: Peter Forsberg, Department of Mechanics and Maritime Sciences

Master's Thesis 2026
Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A Volvo Construction Equipment excavator (model EC300E) with the hydraulic cylinders highlighted.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2026

Data-Driven Plant Models for Hydraulic Excavators
Development of Compact Machine Learning Architectures across Simulated and Physical Platforms
OSKAR HAAPALO, ADAM OSKARSSON
Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology

Abstract

Modern electro-hydraulic systems, such as those found in hydraulic excavators, exhibit highly nonlinear, coupled, and time-varying dynamics. Developing accurate mathematical plant models for these systems is a challenge in heavy machinery automation. Traditional system identification methods often fail to capture complex behaviors like valve dead-zones and hysteresis, while high-fidelity analytical simulation models are computationally heavy and difficult to calibrate. This thesis introduces a data-driven alternative developed in collaboration with CPAC Systems AB: an "Excavator Dynamics Predictor" (EDP) capable of providing fast, accurate real-time dynamic predictions.

Using an automated optimization suite powered by Optuna, this research systematically evaluated a variety of deep sequence learning architectures, including Long Short-Term Memory networks, Temporal Convolutional Networks, and hybrid configurations. Testing revealed that a hybrid TCN-LSTM model, augmented with targeted differential pressure features, achieves the highest prediction accuracy while compressing the model size. The framework utilizes small, specialized machine learning modules to reduce composite prediction errors by 41.3% while keeping the same parameter footprint.

To validate the practical utility of the EDP, a transfer learning framework was developed to bridge the gap between simulation and reality. While models trained purely on simulated data struggled with real-world noise and unpredictable soil-tool interactions, a two-phase fine-tuning schedule successfully adapted the model to a physical excavator using a limited amount of real operational logs. The transfer learning framework lowered the prediction mean absolute error from 8.0 mm/s to 2.9 mm/s, proving the model's sim-to-real ability and generalizability across excavator models. Finally, to meet the strict computational and memory constraints of an industrial microcontroller, model compression via knowledge distillation and quantization-aware training was applied. The resulting compressed architecture achieved a model size of 24 K parameters while only increasing prediction errors by $\approx 30\,\%$. This work demonstrates that hardware-aware deep learning models can serve as a fast, scalable foundation for digital twins and advanced model predictive control algorithms in autonomous construction applications.

Keywords: autonomous excavation, hydraulic modeling, sim-to-real transfer, LSTM, TCN, hyperparameter optimization, knowledge distillation, excavator plant models.

Preface

This thesis represents the culmination of our Master of Science degree in Systems, Control, and Mechatronics at Chalmers University of Technology. The project was carried out over twenty weeks during the spring semester of 2026.
The research was conducted in collaboration with CPAC Systems AB, focusing on the development and evaluation of AI-based plant models for hydraulic excavators. This project provided a unique opportunity to explore how AI and machine learning can be implemented in an engineering area currently dominated by traditional physical modeling.

Acknowledgements

We would like to express our sincere thanks to our supervisor and advisor, Marcus Carlsson, for his insights and support – he is truly a walking encyclopedia on excavators. We would also like to thank our examiner, Peter Forsberg, for his continuous guidance and interest in our work; no one has more AI experience than the man with his own data center at home. Additionally, a warm thanks goes to the entire Active Control team for spending their precious time discussing ideas and providing counsel throughout our project.
Furthermore, we want to thank CPAC Systems AB and Mathias Andreasson for their trust and for giving us this opportunity. The great workplace environment (and the chance to drive excavators) made us excited to get to the office each week. Finally, there would have been no AI plant models without the top-tier computers provided to us, or the occasional table tennis tournaments.

Oskar Haapalo & Adam Oskarsson, Gothenburg, June 2026

Thesis advisor: Marcus Carlsson, CPAC Systems AB
Thesis examiner: Peter Forsberg, Department of Mechanics and Maritime Sciences

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

APRBS	Amplitude-Modulated Pseudo-Random Binary Sequence
ARX	AutoRegressive with eXogenous inputs
CNN	Convolutional Neural Network
DNN	Deep Neural Network
EDP	Excavator Dynamics Predictor
ESN	Echo-State Networks
FDR	False Discovery Rate
GAP	Global Average Pooling
KD	Knowledge Distillation
LME	Linear Mixed-Effect
LSTM	Long Short-Term Memory
MAC	Multiply-Accumulate
MAE	Mean Absolute Error
MI	Mutual Information
MLP	Multi Layer Perceptron
MSE	Mean Squared Error
MPC	Model Predictive Control
NN	Neural Network
ONNX	Open Neural Network Exchange
PE	Persistent Excitation
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SE	Squeez-and-Excitation
SINDy	Sparse Identification of Nonlinear Dynamics
SWA	Stochastic Weight Averaging
TCN	Temporal Convolutional Network
TPE	Tree-structured Parzen Estimator
VAC	Volvo Active Control

Nomenclature

The following nomenclature classifies the indices, sets, operational parameters, and structural variables utilized throughout this thesis. For clarity, components are organized by their respective algorithmic modules, statistical frameworks, and physical subsystems.

LSTM & Recurrent modules

$\mathbf{x}_t$	Input vector at timestep t
$\mathbf{h}_t$	Hidden state vector at timestep t
$\mathbf{W}$	Weight matrices for hidden state transitions
$\mathbf{U}$	Weight matrices for input mappings
$\mathbf{b}$	Bias vector
$\sigma(\cdot)$	Sigmoid activation function
$\mathbf{f}_t$	Forget gate vector at timestep t
$\mathbf{i}_t$	Input gate vector at timestep t
$\mathbf{o}_t$	Output gate vector at timestep t
AttnPool$(\cdot)$	Temporal attention pooling function
$\tilde{\mathbf{h}}$	Attention-pooled summary vector

TCN

$\mathbf{z}_t^{(l)}$	Output feature vector at layer l and timestep t
$\mathbf{W}^{(l)}$	Convolutional weights at layer l
l	Layer index
L	Total number of layers
d	Dilation factor
R	Total effective receptive field size

$F(\cdot)$	Residual mapping function
$\theta^{(l)}$	Residual block parameters at layer l

Metrics & Loss formulations

r_{xy}	Pearson correlation coefficient between x and y
p_{xy}	Joint probability density function between x and y
$I(X;Y)$	Mutual information between X and Y
$R_{xy}(\tau)$	Cross-correlation at displacement temporal lag τ
I_j	Permutation importance of feature j
$\ell_\delta(\cdot)$	Huber loss evaluation function
e	Prediction error residual
δ	Huber loss quadratic-to-linear transition threshold
w_i	Loss scaling weight assigned to cylinder channel i
$\propto$	Proportionality operator

Statistical framework

$\mathcal{D}$	Complete dataset
$\mathcal{D}_{\text{dev}}$	Development dataset
$\mathcal{D}_{\text{test}}$	Frozen test dataset
$\hat{\sigma}_{\text{split}}$	Estimated between-split variance
$\hat{\sigma}_{\text{seed}}$	Estimated within-split seed variance
J	Number of allocated data partitions
S	Number of random seeds per partition
$\bar{\mu}_j$	Mean performance metric for partition j across seeds
M	Total simulated Monte Carlo experiments
δ	Targeted true effect size threshold
Δ_j	Expected difference variable across partitions
$\mathcal{N}$	Normal (Gaussian) distribution
H_1	Alternative statistical hypothesis
μ_Δ	Mean of the distribution of differences
α	Significance level (Type I error threshold)
β	Type II error probability

$z_{1-\alpha}$	Critical z-score for chosen significance level
$z_{1-\beta}$	Critical z-score for targeted statistical power
$\text{Power}(\cdot)$	Empirical statistical power function
$\mathbb{I}(\cdot)$	Indicator function
$p^{(k)}$	p-value from the k-th Monte Carlo simulation

Knowledge distillation

$\hat{y}$	Predicted probability distribution of the student network
y	Category ground-truth label
$\mathbf{z}$	Unnormalized log-probabilities of the student
$\mathbf{z}_{\text{teacher}}$	Unnormalized log-probabilities of the teacher
α	Loss weighting coefficient balancing hard and soft objectives
T	Temperature scaling factor
$\mathcal{L}$	Total joint objective loss function
$\mathcal{L}_{\text{hard}}$	Standard supervised hard-target loss function
$\mathcal{L}_{\text{soft}}$	Soft-target distillation loss function

Physics-based modules

$\mathbf{z}_{\text{phys}}$	Deterministic prediction vector from the physics base
i	Actuator or joint index variable ($i \in \{\text{boom, arm, bucket}\}$)
Δt	Discrete sampling period (0.01 s)
$\hat{v}_i(t + \Delta t)$	Predicted cylinder velocity for actuator i at step $t + \Delta t$
$v_{i,t}$	Measured cylinder velocity for actuator i at time t
$\mathbf{q}_t$	Joint position vector at time t
$\dot{q}_{i,t}$	Joint velocity for actuator i at time t
$J_{v,i}$	Kinematic leverage ratio for actuator i
$F_{\text{cyl},i,t}$	Net hydraulic cylinder force for actuator i at time t
$\bar{M}_{\text{eq},i}$	Constant, time-averaged equivalent mass for actuator i
$M_{\text{eq},i}(\mathbf{q}_t)$	Configuration-dependent equivalent mass matrix

Structural modules

$\mathbf{X}$	Raw input sequence matrix
$\mathbf{R}$	Output sequence matrix from the TCN layer
T	Total sequence length (window size)
C	Channel dimension width of the TCN output
$\mathbf{R}'$	Intermediate sequence after self-attention
$\mathrm{MHA}(\cdot)$	Multi-Head Self-Attention processing block
$\mathrm{LN}(\cdot)$	Layer normalization function
$\mathbf{R}''$	Enriched sequence after the feed-forward network
$\mathrm{FFN}(\cdot)$	Position-wise Feed-Forward Network processing block
h	Number of attention heads in the MHA sublayer
$\mathbf{r}_t$	Update gate vector of the GRU at time step t
$\mathbf{z}_t$	Reset gate vector of the GRU at time step t
$\tanh(\cdot)$	Hyperbolic tangent activation function
$\odot$	Hadamard (element-wise) product operator
$\mathbf{h}_{\text{aug}}$	Augmented hidden state vector with temporal differences
$\mathbf{z}$	Global channel descriptor vector via pooling
$\mathbf{u}_t$	Feature vector of the TCN backbone at time step t
$\mathbf{s}$	Channel modulation weight vector
$\delta(\cdot)$	Rectified Linear Unit (ReLU) activation function
$\widetilde{\mathbf{u}}_t$	Recalibrated feature vector after channel scaling
$\mathbf{g}$	Temporal summary vector from global average pooling
$\mathbf{h}_i^{\text{gated}}$	Gated backbone feature vector for cylinder head i
D	Dimensionality of the shared backbone feature space
$\mathbf{s}_i$	Per-feature scale vector for cylinder gating mechanism i
$\mathcal{O}(\cdot)$	Big-O computational complexity scaling operator
$\mathbf{C}$	Stacked feature matrix of the three cylinder channels
d_{head}	Dimensionality of an individual cylinder head channel
$\mathbf{C}'$	Refined feature matrix after cross-cylinder attention
$g(\cdot)$	Output transformation or activation function
a_i	Learnable amplitude parameter scaling output range for cylinder i
$a_{\min}$	Minimum allowable bound for activation amplitude
$a_{\max}$	Maximum allowable bound for activation amplitude

Optimization suite

η	Optimization learning rate
λ	Decoupled weight decay coefficient
δ	Dropout rate hyperparameter
$S_{\text{composite}}$	Composite evaluation objective score
P_{95}	95th percentile error bound
P_{99}	99th percentile error bound
β_1	Fixed effect coefficient in the linear mixed-effects model
p_{BH}	Adjusted p-value via Benjamini-Hochberg procedure
$\lvert d_z \rvert$	Absolute value of Cohen's effect size
B	Total count of binary module activation toggles
$\mathbf{m}^*$	Optimized joint binary module selection vector
$\boldsymbol{\theta}^*$	Optimized joint continuous hyperparameter vector
$\mathbf{m}$	Arbitrary binary module configuration vector
$\boldsymbol{\theta}$	Arbitrary continuous hyperparameter configuration vector
INT8	8-bit integer precision numerical format

Contents

List of Figures

List of Tables

1
Introduction

Earthmoving machinery, particularly hydraulic excavators, plays an essential role in the global construction and mining industries. As these sectors face increasing demands for higher productivity, reduced environmental impact, and improved workplace safety [1], heavy machinery automation has emerged as a technological frontier. Semi-autonomous functions, such as automatic grading, significantly reduce the cognitive load on human operators while ensuring consistent, high-quality results. Industrial frameworks, such as the Volvo Active Control system (VAC), exemplify the shift towards operator-assist technologies that bridge the gap between manual operation and full autonomy [2].
The realization of these advanced autonomous features requires highly accurate and responsive control architectures. Whether utilizing next-generation feedback controllers or exploring optimization-based frameworks like Model Predictive Control (MPC), the system's performance will be fundamentally bound by the accuracy and computational efficiency of its internal plant model. Given the complexity of hydraulic systems, this thesis explores machine learning as a means of developing accurate, data-driven plant models capable of providing the real-time dynamic predictions required for advanced control.

1.1 Background

In the field of systems, control, and mechatronics, the design of high-performance control algorithms is dependent on the existence of accurate mathematical models. The research area of system identification addresses this need by deriving the behavior of complex systems directly from observed data [3]. For complex machinery, such as hydraulic excavators, obtaining these models is particularly challenging. Traditional methods for system identification often rely on parametric models, which require significant prior knowledge of the system and extensive manual tuning. This process is not only time-consuming but often fails to accurately capture the complex, nonlinear dynamics inherent in hydraulic systems.
This project addresses the limitations of current modeling techniques used in industrial applications. Specifically, current engineering workflows for systems such as VAC rely on Simulink models to simulate the excavator plant for controller testing and validation. Although functional, these models are difficult to calibrate and lack the efficiency required for rapid development cycles. Recent advances in machine learning have introduced data-driven approaches, such as sparse identification of nonlinear dynamics (SINDy) [4] and Neural Networks (NN), which allow models to

be learned directly from data with high approximation capabilities.
The development of accurate and data-driven plant models unlocks significant potential for the implementation of advanced control strategies. One promising application in this domain is MPC. Unlike traditional feedback controllers, MPC relies on an internal dynamic model to predict future system states and proactively optimize control inputs over a receding horizon. However, the performance of an MPC formulation is limited by the quality and execution speed of this internal model [5]. A highly accurate, NN-based model that captures the non-linear hydraulic dynamics, input delays, and dead-zones would enable an MPC framework to compute optimal, energy-efficient trajectories in real-time while strictly adhering to hardware constraints.
In the context of industrial excavators, such advanced control schemes serve as the foundation for sophisticated semi-autonomous functions. For existing operator-assist frameworks, such as VAC, integrating a data-driven plant model could improve the precision and reliability of features such as automatic grading, precise level cutting, and automated trenching [6]. As an accurate model allows the controller to anticipate the complex interplay between simultaneous multi-joint movements and highly variable soil interaction forces, the machine can execute tasks with greater precision than what is achievable through manual operation alone. Ultimately, unlocking these semi-autonomous capabilities not only reduces the cognitive load and fatigue on human operators, but also improves overall site productivity and represents a stepping stone toward fully autonomous excavation.

1.2 Related work

The application of ML to model the complex dynamics of hydraulic excavators has gained significant traction, particularly to enable advanced control strategies such as MPC and inverse dynamics control. A major focus in recent literature is the transition from purely data-driven black-box models to grey-box or physics-informed architectures that ensure physical consistency, enhance safety, and improve generalization to unseen operational conditions.
Research by Weigand et al. [7] provides a comprehensive comparison of data-driven white-box, black-box, and grey-box (hybrid) feedforward controllers for hydraulic excavators. The authors developed a black-box model using a Multilayer Perceptron (MLP) trained on sensor data, and a hybrid grey-box model that combines basic physical equations from fluid mechanics (orifice equations) with MLPs to correct for dynamic deviations. Their experimental validation on a real excavator demonstrated that both the black-box and hybrid models significantly outperformed the purely analytical white-box model in trajectory tracking under nominal conditions. However, when subjected to previously unseen external payloads, the hybrid grey-box model exhibited superior extrapolation capabilities and robustness compared to the pure black-box approach. This study underscores the limitation of standard neural networks in covering the entire phase space of an excavator and highlights the need to embed expert physical knowledge to ensure safe and reliable control.
Building on the need for physical consistency, Feng et al. [8] proposed a highly structured physics-informed neural network to model the inverse dynamics of an ex-

cavator. Recognizing that standard neural networks can overfit and produce physically impossible predictions, they employed a Deep Lagrangian Network to explicitly encode the rigid body dynamics of the excavator, making sure that the learned parameters conserve energy and mass. Because DeLaN alone struggles to represent unmodeled nonlinearities such as hydraulic flexibility and stick-slip friction, the authors augmented the architecture with a Convolutional Neural Network (CNN) and a Long Short-Term Memory (LSTM) network to perform residual learning on the hydraulic disturbances. The resulting DeLaN-CNN-LSTM model achieved high predictive accuracy across simulations, a scaled-down platform, and a real 36-ton excavator. Furthermore, this accurate dynamics model was integrated into a Prescribed Performance Inverse Dynamics Controller, which successfully constrained trajectory tracking errors within a finite region even when subjected to sudden variations in external loads.

Expanding on the methodology of physics-informed learning, Zhang et al. [9] introduced a framework that embeds physical laws, specifically equations of motion and state dependency directly into the loss function of LSTM networks. Although standard LSTMs are adept at capturing temporal dependencies, they often struggle with data scarcity and can produce results that violate fundamental physical principles. By penalizing physical inconsistencies during the training phase, their Physics-Informed Multi-LSTM approach ensures that the model remains within a feasible solution space, even when training data is limited. This regularized learning approach provides a robust alternative for modeling the nonlinear hysteretic behavior inherent in mechanical systems, offering a middle ground between purely data-driven recurrent networks and rigid analytical models.

To address the challenge of time-varying dynamics, Park et al. [10] investigated an online learning control framework utilizing Echo-State Networks (ESN). ESNs are a class of recurrent neural networks (RNN) in which a large pool of hidden nodes provides complex dynamics, but only the final output weights are trained. This turns the training process into a simple regression problem, making the networks fast and well suited for real-time online implementation. In their framework, an indirect adaptive control approach was used: one ESN continuously learned the inverse model of the excavator using current and delayed input-output signals, while a second ESN utilized these updated weights to simultaneously generate the control inputs required for trajectory tracking. Validated on a 21-ton excavator, the controller was tested on both single-joint movements and complex combined digging motions. The ESN-based approach successfully compensated for changing plant dynamics over time without requiring a prior mathematical representation of the target plant. It also outperformed a standard PD controller by naturally correcting the tracking phase errors caused by the inherent delays of the hydraulic system.

While online learning offers excellent adaptability, it can present significant safety risks during the exploration phase on heavy machinery, and generic RNNs often require massive datasets to implicitly learn explicit time delays and dead-zones. To mitigate this, Lee et al. [11] proposed a highly structured, physics-inspired data-driven model that explicitly isolates these hydraulic characteristics. Rather than employing a single monolithic neural network, their architecture features a modular design: an infinite impulse response unit explicitly handles the input delays, a

piecewise linear map accommodates the state-dependent valve dead-zones, and multilayer perceptrons capture the remaining coupled nonlinearities. Because the model is modular, it could be inverted efficiently for control and trained safely offline using real operational data. When tested on a 38-ton commercial excavator, this modular inversion controller achieved high precision, maintaining a path-following error of less than 2 cm even when subjected to severe soil interactions during digging and grading tasks.
Egli and Hutter [12, 13] proposed a fully data-driven reinforcement learning framework for excavator automation, intentionally bypassing the need for machine-specific analytical modes or manual gain tuning. They utilized a two step process: firstly training an NN-based actuator model via supervised learning on operational data to capture the highly nonlinear dynamics, including hydraulic coupling, dead-zones and cylinder to joint space mapping. Secondly, the learned model was embedded into a simulation environment to serve as a digital twin for training an RL policy which directly outputs pilot stage valve commands.
While their initial work [12] demonstrated the feasibility of sim-to-real transfer for bucket position tracking in free space, their subsequent study [13] introduced a more robust velocity tracking paradigm. By incorporating bucket orientation control and accounting for joint velocities, the updated framework achieved significantly higher responsiveness and precision. Experimental validation on a 12-ton excavator during realistic grading tasks showed that the RL-based controller outperformed a commercial grading system, which had been hand tuned, even during intensive soil interaction. By requiring only basic kinematic link lengths as machine-specific input, this approach represents a highly generalizable and scalable solution for the automation of diverse heavy machinery without the overhead of classical system identification.
However, the existing literature lacks a fully automated data-driven approach for optimal model construction. Specifically, the use of advanced optimization frameworks, such as genetic algorithms or Optuna, to rapidly develop a forward dynamics model remains unexplored. To address this, the proposed method utilizes a model-agnostic approach, requiring only rigid body parameters to initialize the architecture search. Furthermore, the potential of transfer learning within this domain has not been adequately investigated. There is value in developing a foundational excavator backbone model - a base model able to capture shared dynamic similarities that can be rapidly fine-tuned for new excavator variants. Moreover, questions regarding model evaluation remain open: How do we formally define an 'optimal' plant model? How can one determine the most effective training objectives and establish the theoretical performance limits, given the inherent constraints of the available data.

1.3 Aims and objectives

In collaboration with CPAC Systems, this thesis aims to improve the efficiency and accuracy of current excavator plant models by developing a data-driven alternative. The goal is to create an "Excavator Dynamics Predictor" (EDP) capable of predicting the next state of an excavator, such as cylinder positions, velocities, and pressures,

based on current state variables and control inputs.
At the end of this project, the goal is to demonstrate a functioning EDP model trained on data collected from MATLAB Simulink simulations and potentially real excavator data. The thesis aims to show that this data-driven model can accurately learn the dynamics of new machines, thus improving simulation efficiency for digital twins and facilitating the future implementation of advanced control strategies such as MPC.
The following technical objectives were defined:

O1: Establish a robust data pipeline for processing operational data from Simulink and physical sensors.
O2: Implement and train multiple deep learning architectures to serve as the EDP backbone.
O3: Develop an automated optimization suite for model selection.
O4: Validate the EDP through closed-loop simulation and comparison against physical machine data.

1.3.1 Research questions

Guided by the objectives above, the thesis investigates the following questions:

RQ1: How does the inclusion of a physics base module in a neural network affect the model's ability to generalize to new excavator variants?
RQ2: How does the EDP model perform on real-world data containing stochastic soil-tool interactions when trained on simulated free-air data?
RQ3: What are the optimal neural network architectures and machine learning modules for training hydraulic excavator plant models?
RQ4: To what extent can an excavator backbone model utilize transfer learning to accelerate calibration and fine-tuning for new excavator variants?

1.4 Delimitations and limitations

To ensure the feasibility of this thesis within the allocated time frame and to define the operational boundaries of the developed data-driven plant model, specific delimitations and limitations are established.

1.4.1 Delimitations

The data-driven plant model is strictly delimited to predicting the primary kinematic chain dynamics of the excavator, specifically the boom, arm, and bucket cylinders. Consequently, the swing motion of the upper carriage is assumed to be locked in position, and the locomotion of the undercarriage tracks is excluded from the system identification process.
This exclusion is motivated by both computational efficiency and operational relevance. Modeling track locomotion and swing dynamics simultaneously would exponentially increase the system dimensionality and data requirements. In addition, isolating the boom, arm, and bucket is representative of common automated construction tasks. For example, grading is typically executed while the machine is

stationary and digging along a fixed vertical plane, making this scoped system identification applicable to real-world target applications.

1.4.2 Limitations

- **Absence of terramechanics in simulation:** The study relies on a Simulink environment that lacks modeling for terramechanics or complex soil-tool interactions. Since the baseline training data is generated from simulated "free-air" operations, the data set lacks the highly nonlinear, stochastic resistive forces generated when a physical bucket interacts with diverse material types (e.g., gravel, clay, or rock). This constraint restricts the model's capacity to learn external load-bearing dynamics solely from virtual environments, causing the sim-to-real transfer pipeline to be highly dependent on physical log data for fine-tuning.
- **Onboard hardware constraints:** Because the final plant model is designed for real-time onboard deployment, its architecture is constrained by the strict computational power and memory limits of the physical machine's target hardware. This embedded deployment environment restricts the maximum size, layer depth, and execution latency of the neural network architectures, imposing a trade-off between real-time deterministic execution and the theoretical maximum approximation accuracy achievable by unconstrained models.
- **Stochastic noise discrepancies:** Although artificial noise is injected into the Simulink signals to approximate physical telemetry, it cannot perfectly replicate the complex stochastic noise distributions, sensor drift, temperature-dependent variations, and asynchronous CAN-bus (Controller Area Network) communication latencies present in the physical excavator hardware. Consequently, a systematic gap exists in the model's inherent ability to reject physical hardware noise without fine-tuning on real logged data.

2
Theory

This chapter establishes the theoretical foundations required to understand the modeling, optimization, and evaluation frameworks used in this thesis. The material is divided into three primary domains. First, the physical characteristics of hydraulic excavators and their corresponding control challenges are detailed. Next, data-driven system identification techniques are explored, focusing on deep learning architectures for time-series forecasting. Finally, the chapter covers data quality considerations, hyperparameter optimization via the Optuna framework, and the statistical validation methodologies used to assess model performance and deployment feasibility.

2.1 Excavator dynamics

To build accurate data-driven models, it is first necessary to understand the underlying physical and mechanical behavior of the machine. This section provides an overview of hydraulic excavator dynamics, focusing on the fundamental principles of fluid power systems and the dynamic characteristics of hydraulic actuation. Additionally, it outlines the standard electro-hydraulic architectures currently used to regulate these machines, alongside the unique control challenges introduced by automation and semi-autonomous operational functions.

2.1.1 Hydraulics

An excavator relies on its hydraulic system to deliver the high power required to actuate its heavy structural components. While grounded in Pascal's law of uniform fluid pressure, the system's practical dynamics are highly complex and dictate the overall behavior of the machine.

2.1.1.1 Hydraulic system overview

The delivered hydraulic power is highly dependent on operating temperature and pressure. Variations in these states affect the fluid's viscosity, bulk modulus, and overall flow characteristics, thereby affecting system efficiency and response. Because hydraulic fluid naturally flows from regions of high pressure to regions of low pressure, precise flow control is essential to mitigate energy losses, manage heat generation, and ensure stable operation.
A typical hydraulic system of an excavator consists of the following main components:

- **Fluid reservoir:** Stores hydraulic fluid, allowing cooling and filtering before recirculation.
- **Pumps:** Converts mechanical energy from the engine into hydraulic energy by generating a pressurized fluid flow.
- **Control valves:** Regulates the direction, pressure, and flow rate of the hydraulic fluid supplied to the actuators. Valves enable controlled motion of excavator joints in response to the operator's commands.
- **Actuators:** Convert hydraulic energy back into mechanical force and motion. In an excavator, these include hydraulic motors for driving and cabin rotation, as well as hydraulic cylinders. The latter act as linear actuators, providing the forces required to move the boom, arm, and bucket linkages which are the primary focus of this work.

During operation, pressurized fluid is routed from the pumps through the control valves to the respective cylinders, with the valve spool displacements determining exactly which cylinders are supplied and to what extent.

2.1.1.2 Dynamic characteristics of hydraulic actuation

From a dynamics perspective, the resulting actuator motion depends not only on the applied control input but also on the internal dynamics of the hydraulic circuit. Hydraulic actuation exhibits nonlinear and time-dependent behavior due to several physical effects inherent to hydraulic systems [14], including:

- **Hysteresis**: Stochastic I/O relationship caused by friction and valve dynamics, leading to different system responses for increasing and decreasing inputs.
- **Deadband**: A range of control input for which no actuator motion occurs, commonly introduced by valve overlap and mechanical tolerances.
- **Saturation**: Physical limitations on pressure, flow rate, or actuator stroke, which restrict the maximum achievable force or velocity.
- **Stick-slip**: Friction-induced oscillatory motion that occurs at low velocities, resulting in irregular actuator behavior.
- **Bulk modulus**: Finite compressibility of the hydraulic fluid, which introduces compliance and affects pressure dynamics and system stiffness.
- **Cross-coupling**: Dynamic interaction between multiple actuators sharing common hydraulic supply lines or pumps, leading to coupled responses between different joints.

2.1.2 Control of excavators

Regulating heavy construction machinery requires control architectures capable of handling highly coupled nonlinear physical systems. This subsection examines modern electro-hydraulic control setups and their evolution toward semi-autonomous and fully automated functions. It also outlines the core technical challenges associated with autonomous excavator control, highlighting why accurate, real-time predictions of machine kinematics are necessary to improve closed-loop performance.

2.1.2.1 Modern electro-hydraulic control architecture

Modern excavators use electro-hydraulic control architectures in which operator commands are processed electronically and translated into valve actuation signals. Compared to purely mechanical or hydro-mechanical systems, electro-hydraulic architectures enable signal conditioning, feedback control, and software-based control laws [14].
The integration of electronic control units, sensors, and communication networks has transformed excavator control from simple energy-saving systems into intelligent, automated platforms. This evolution enables advanced functions, including trajectory control, coordinated multi-actuator control, and semi-autonomous operation.

2.1.2.2 Automation and semi-autonomous functions

Traditionally, heavy construction equipment is operated manually by trained operators. However, increasing demand for productivity, energy efficiency, and safety has driven the development of semi-autonomous and automation functions in excavators. A representative example is the Auto-Grading function, where the boom, arm, and bucket cylinders must move in a coordinated manner to achieve a smooth and level work surface.
Such functionality relies on accurate kinematic models, trajectory planning, and coordinated multi-actuator control to track a desired tool trajectory with high precision. Research shows that trajectory control and multi-actuator coordination are fundamental enablers for automated excavation and grading tasks, particularly given the nonlinear and strongly coupled dynamics of electro-hydraulic excavator systems.
By abstracting low-level actuator coordination, semi-autonomous functions allow the operator to control the excavation task using fewer inputs, thereby reducing cognitive load and physical fatigue while maintaining high operational performance.
To be competitive with manual operation, such semi-autonomous functions must:

- Achieve trajectory accuracy comparable to that of skilled operators.
- Maintain high productivity by enabling smooth and efficient motion without sacrificing precision.
- Provide high reproducibility, resulting in low variance in task execution.
- Ensure safe operation under varying working conditions and external disturbances.

2.1.2.3 Challenges in autonomous excavator control

Unlike industrial robotic manipulators operating in highly structured factory environments, excavators operate in dynamic, unstructured settings where external disturbances and internal system parameters exhibit high variance. The primary challenges in designing robust autonomous functions stem from three main sources:

- **Varying terramechanics:** The excavator bucket interacts with distinct geological materials, such as loose sand, compacted clay, or solid rock. Each material presents different resistive forces and friction coefficients during the penetration and cutting phases. These unpredictable soil-tool interactions act

as stochastic load disturbances on the hydraulic actuators, altering the required control effort.
- **Thermal fluctuations:** The operating temperature of the hydraulic fluid continuously shifts depending on the ambient environment and the operational load. Temperature variations directly alter the fluid's viscosity and effective bulk modulus, which in turn dynamically changes the system's volumetric efficiency, damping characteristics, and dynamic stiffness.
- **Mechanical degradation:** Over a machine's lifetime, physical wear and tear causes the fundamental dynamic parameters of the excavator to drift. Phenomena such as increased seal friction, widened valve dead-zones, and hydraulic pump degradation transform the excavator into a complex, time-varying dynamic system.

The culmination of these factors makes it highly challenging to derive a single fixed-parameter analytical model capable of accurately representing the excavator across all operational regimes. Because the underlying dynamics are constantly shifting due to temperature, wear, and unpredictable soil disturbances, traditional rigid control strategies struggle to maintain trajectory precision.
Therefore, achieving reliable autonomous functionality requires a combination of highly expressive plant models and advanced control strategies capable of capturing environmental disturbances and adapting to unseen operational states in real time.

2.2 Data-driven system identification

System identification is the process of building mathematical models of dynamical systems using measurements of input and output signals [15].

2.2.1 Classical system identification

The traditional workflow consists of four essential phases: collecting persistently exciting data, selecting a suitable model architecture, estimating parameters by minimizing a prediction loss function, and validating the model's generalization on unseen data.
Fundamentally, the objective is to approximate a function f that maps past system outputs y and control inputs u to future states:

$$y_{t+1} = f(y_t, y_{t-1}, \ldots, u_t, u_{t-1}, \ldots; \theta) + e_t$$

where θ represents the trainable parameters and e_t is the modeling error. Unlike white-box approaches derived strictly from first-principles physics equations, system identification typically employs black-box or grey-box architectures to capture internal dynamics that are too complex to model analytically.
Classical system identification relies heavily on linear or simple nonlinear parametric structures. Although linear models (such as the standard ARX model) offer computational efficiency and strong convergence guarantees, they generally fail to capture the severe nonlinearities of hydraulic machinery. To address this, the Nonlinear ARX model generalizes the mapping via a nonlinear function F:

$$y(t) = F(y(t-1), \ldots, y(t-n_a), u(t-1), \ldots, u(t-n_b)) + e(t)$$

In classical methods, F is typically approximated using polynomial basis functions or wavelets. However, for high-dimensional, cross-coupled systems like an excavator, the required number of polynomial terms explodes, leading to severe overfitting. This limitation, combined with the universal approximation theorem, motivates the transition to deep learning, where neural networks are utilized as highly scalable and expressive function approximators for F.

2.2.2 Deep learning for dynamics

Deep learning introduces the use of deep neural networks (DNN) as universal function approximators for systems identification. Unlike classical polynomial methods, DNNs can learn highly complex, discrete nonlinear representations.

2.2.2.1 Recurrent neural networks

While standard feedforward neural networks approximate static functions, dynamical systems exhibit memory, where current state depends on past inputs. Recurrent Neural Networks (RNNs) address this by maintaining an internal hidden state $h_t \in \mathbb{R}^{n_h}$ which represents the systems past. At each timestep t, the hidden state is updated according to

$$\mathbf{h}_t = \sigma(W_h \mathbf{h}_{t-1} + U_h \mathbf{x}_t + \mathbf{b}_h) \tag{2.1}$$

where $\mathbf{x}_t \in \mathbb{R}^{n_x}$ is the current input, $W_h \in \mathbb{R}^{n_h \times n_h}$ and $U_h \in \mathbb{R}^{n_h \times n_x}$ are learned weight matrices, and $b_h \in \mathbb{R}^{n_h}$ the bias vector. The weighted linear combination of the previous state and current input, followed by the nonlinear sigmoid activation function $\sigma(\cdot)$, shapes the internal state dynamics.

2.2.2.2 Long short-term memory

Specialized RNN structures, such as the LSTM networks, are designed to model temporal sequences with long-range dependencies [16]. Standard RNNs suffer from the vanishing gradient problem when trained using backpropagation through time, which limits their ability to learn long-term temporal dependencies. LSTMs address this limitation by introducing an internal memory cell $c_t \in \mathbb{R}^{n_h}$ and three multiplicative gates that regulate the temporal flow of information:

- **Forget Gate** (f_t): Scales the previous cell state c_{t-1}, controlling the decay of stored state information.
- **Input Gate** (i_t): Modulates the incorporation of new candidate information $\tilde{c}_t$ into the cell state.
- **Output Gate** (o_t): Determines how much of the internal cell state is exposed through the hidden state h_t.

Given an input vector $x_t \in \mathbb{R}^{n_{\text{in}}}$ at timestep t, these gates and states define the standard LSTM cell updates as:

$$\mathbf{f}_t = \sigma(W_f \mathbf{x}_t + U_f \mathbf{h}_{t-1} + \mathbf{b}_f) \tag{2.2}$$
$$\mathbf{i}_t = \sigma(W_i \mathbf{x}_t + U_i \mathbf{h}_{t-1} + \mathbf{b}_i) \tag{2.3}$$
$$\tilde{\mathbf{c}}_t = \tanh(W_c \mathbf{x}_t + U_c \mathbf{h}_{t-1} + \mathbf{b}_c) \tag{2.4}$$
$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \tag{2.5}$$
$$\mathbf{o}_t = \sigma(W_o \mathbf{x}_t + U_o \mathbf{h}_{t-1} + \mathbf{b}_o) \tag{2.6}$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{2.7}$$

where $\odot$ denotes element wise multiplication, $\sigma(\cdot)$ is the sigmoid function, and $W, U, \mathbf{b}$ are the learned weights and biases. The additive structure of the update (eq. (2.5)) enables linear gradient propagation over a long horizon, combating the issue of vanishing gradient caused by the multiplicative state evolution in standard RNNs. On the other hand, the input sequence is processed sequentially, meaning that LSTMs can struggle to efficiently extract highly local structural features across broad temporal windows.

2.2.2.3 Temporal convolutional networks

Alternative approaches to temporal modeling shifts away from recurrence towards fully convolutional structures, which capture long-term dependencies via stable, parallelized deep training. Temporal Convolutional Networks (TCNs) are a neural network architecture designed for sequence modeling and system identification, utilizing convolutions to handle sequential data as an alternative to recurrent architectures [17].
TCNs rely on 1D causal convolutions to enforce the temporal ordering of inputs: the output at time t depends only on inputs from time t and earlier. For an input sequence $\mathbf{X} \in \mathbb{R}^{T \times n_{\text{in}}}$, the output sequence element $\mathbf{z}_t^{(l)}$ at layer l is given by:

$$\mathbf{z}_t^{(l)} = \sum_{i=0}^{k-1} W_i^{(l)} \mathbf{z}_{t-d \cdot i}^{(l-1)} \tag{2.8}$$

where k represents kernel size, d dilation factor, $W^{(l)}$ the convolutional weights at layer l and $z^{(l-1)}$ is the input vector from the previous layer (with $z^{(0)} = \mathbf{x}$).
The dilation factor d allows the receptive field to grow exponentially with network depth, allowing the network to capture long-term dependencies efficiently. The total receptive field size R across L layers is defined as:

$$R = 1 + \sum_{l=1}^{L} (k-1) \cdot d_l \tag{2.9}$$

To improve stability during training, TCNs incorporate residual connections such that each block outputs:

$$z^{(l)} = z^{(l-1)} + F(z^{(l-1)}; \theta^{(l)}) \tag{2.10}$$

where $F(\cdot)$ represents the residual mapping learned by the block and $\theta^{(l)}$ its parameters.

While dilated convolutions provide stable training and an exact bounded temporal window, the lack of an explicit, evolving recurrent state causes TCNs to sometimes lack high precision sequential tracking capabilities inherent to recurrent memory cells.

2.3 Data quality and persistent excitation

In both classical system identification and modern deep learning-based modeling, the predictive capability of the learned model is constrained by the quality of the training data. While deep neural networks excel at interpolating within the distribution of their training data, their performance degrades when forced to extrapolate to unseen states. Therefore, acquiring a large volume of data is insufficient; the data must be informative and cover the full operational domain of the system. In control theory, this requirement is described by the concept of persistent excitation (PE) [18].
An input signal $u(t)$ is said to be persistently exciting of order n if it excites all the dynamic modes of the system up to that order. If the input lacks sufficient spectral richness the resulting data will lie in a lower-dimensional subspace, preventing unique identification of the system dynamics. Consequently, the optimization algorithm (in this case gradient descent for a neural network) will not have enough information to uniquely identify the true parameters of the underlying plant model.
In the context of learning a complex and nonlinear plant model for a hydraulic excavator, persistent excitation translates to the necessity of dynamic and varied data collection. The excavator's hydraulic actuators exhibit frequency-dependent behaviors, state-dependent dead-zones, and pressure saturation. If the training dataset only contains slow, steady-state movements, the neural network will fail to learn the high-frequency transient responses of the hydraulic valves.
Furthermore, to learn the interaction dynamics between the excavator and the environment, the input signals must drive the machine through varied load conditions. Training a model exclusively on free-air motion will result in an uninformative dataset regarding contact forces, making it hard for the network to accurately predict the system's behavior during actual digging or grading operations. Therefore, the design of the excitation trajectories is an important prerequisite for successful data-driven modeling. Methods to achieve PE data often utilizes steps, multi-sine waves, or amplitude-modulated pseudo-random binary sequences (APRBS) on the joystick commands [19].

2.4 Feature selection and evaluation metrics

To effectively reduce dimensionality and select the most informative predictors, several statistical and information-theoretic metrics are employed. These metrics allow for a quantitative assessment of the linear, nonlinear, and temporal relationships between input features and target variables.

Pearson correlation coefficient: The Pearson correlation coefficient (r) measures the linear relationship between two continuous variables. It is defined as the

covariance of the two variables divided by the product of their standard deviations [20]:

$$r_{xy} = \frac{\sum_{t=1}^{T}(x_t - \bar{x})(y_t - \bar{y})}{\sqrt{\sum_{t=1}^{T}(x_t - \bar{x})^2}\sqrt{\sum_{t=1}^{T}(y_t - \bar{y})^2}} \tag{2.11}$$

The resulting value ranges from -1 to 1, where values near the extremes indicate a strong linear dependency, while a value of 0 implies no linear correlation.

Mutual information: Rooted in Shannon's Information Theory, Mutual Information (MI) quantifies the amount of information obtained about one random variable through the observation of another [21]. For two continuous random variables X and Y with joint probability density function $p(x, y)$ and marginal probability density functions $p(x)$ and $p(y)$, MI is defined as:

$$I(X;Y) = \int_X \int_Y p(x,y) \log\left(\frac{p(x,y)}{p(x)p(y)}\right) dxdy \tag{2.12}$$

Unlike the Pearson coefficient, MI does not assume linearity. It is capable of capturing complex nonlinear dependencies by explicitly measuring the reduction in uncertainty (entropy) between variables.

Cross-correlation and temporal lag: Cross-correlation is a metric that measures the degree of similarity between two discrete time-series as a function of the displacement (lag) of one relative to the other [22]. The cross-correlation sequence $R_{xy}(\tau)$ at displacement τ is given by:

$$R_{xy}(\tau) = \sum_t x_t y_{t+\tau} \tag{2.13}$$

In dynamic time-series analysis, this is used to identify the specific optimal lag $\tau^* = \arg\max_\tau R_{xy}(\tau)$ at which two signals are most highly synchronized, revealing the physical latency inherent within a dynamic system.

Permutation feature importance: Permutation feature importance is a model-agnostic method used to calculate the post-hoc contribution of a specific feature to the predictive power of a trained model. Let $L(y, \hat{y})$ be the loss function of a model scored on a validation data set $\mathcal{D}$. The importance I_j of feature j is computed by measuring the increase in the model's prediction error after the values of feature j are randomly shuffled (permuted) across the data set [23]:

$$I_j = \mathcal{L}(\mathcal{D}_{\text{permuted},j}) - \mathcal{L}(\mathcal{D}) \tag{2.14}$$

This shuffling process breaks the underlying physical relationship between the feature and the target variable. Evaluating the resulting drop in performance allows for an assessment of the necessity of the feature in a multivariate context.

2.5 Optuna framework

Optuna is an open-source hyperparameter optimization framework designed for machine learning workflows [24]. It automates the search for optimal hyperparameter configurations by framing the problem as the minimization (or maximization) of a black-box objective function.

2.5.1 Define-by-run API

A distinguishing design principle of Optuna is its define-by-run API, in contrast to the define-*and*-run paradigm used by earlier frameworks such as Hyperopt. In a define-and-run framework, the entire search space must be declared statically before optimization begins. In Optuna's define-by-run approach, the search space is constructed dynamically within the objective function itself: each hyperparameter is sampled on demand during a trial's execution. This enables conditional and hierarchical search spaces without requiring the user to enumerate all possible combinations upfront. For example, the number of layers in a network can be sampled first, and then per-layer parameters are sampled only for the layers that exist.

2.5.2 Tree-structured Parzen Estimator

By default, Optuna uses the Tree-structured Parzen Estimator (TPE) as its sampling algorithm. TPE is a sequential model-based optimization method that models the objective function indirectly. Rather than fitting a surrogate $p(y \mid \theta)$ that predicts the objective value y given hyperparameters θ, TPE models two separate densities:

$$p(\theta \mid y) = \begin{cases} \ell(\theta) & \text{if } y < y^*, \\ g(\theta) & \text{if } y \geq y^*, \end{cases} \tag{2.15}$$

where y^* is a quantile threshold that separates "good" from "poor" trials, and $\ell(\theta)$ and $g(\theta)$ are kernel density estimates fitted to the hyperparameter vectors of each group, respectively. New candidate configurations are proposed by maximizing the ratio $\ell(\theta)/g(\theta)$, which can be shown to be proportional to the expected improvement acquisition function. The tree-structured extension handles conditional dependencies between hyperparameters by factoring the density estimates along the dependency tree.

2.5.3 Automated early stopping of trials

Optuna provides a pruning mechanism that terminates unpromising trials before they complete, substantially reducing the total computational cost of a search. During training, the objective function periodically reports intermediate values (e.g., the validation loss at each epoch) to the framework. A pruning algorithm then decides whether to stop the trial early based on its trajectory relative to previously completed trials.
Several pruning strategies are available. The median pruner terminates a trial at step t if its intermediate value is worse than the median of intermediate values

reported by all previous trials at the same step. More sophisticated alternatives, such as the Hyperband pruner [25], allocate resources using successive halving: a large number of trials are started with a small budget, and only the most promising fraction is promoted to the next budget level, iterating until a single configuration remains.

2.5.4 Study and trial abstraction

Optuna organizes an optimization session into a study, which manages a collection of trials. Each trial corresponds to one evaluation of the objective function with a specific hyperparameter configuration. The study persists the history of all completed, pruned, and failed trials, enabling the sampler to make increasingly informed suggestions as the search progresses. Studies can optionally be backed by a relational database, allowing optimization to be paused, resumed, or distributed across multiple workers.

2.6 Statistical framework and foundational theory

This section details the theoretical concepts used to validate the experimental results of this study. It addresses the principles of data partitioning, variance decomposition, and sample size determination to ensure statistical significance. Furthermore, it introduces the linear mixed-effects models used for confirmatory validation.

2.6.1 Data partitioning and variance decomposition

To guarantee unbiased generalization estimates in models trained on sequential or episodic data, data sets must be isolated strictly at the trajectory or session level. Let the complete data set $\mathcal{D}$ be partitioned into a development set $\mathcal{D}_{\text{dev}}$ and a completely frozen test set $\mathcal{D}_{\text{test}}$, defined such that:

$$\mathcal{D} = \mathcal{D}_{\text{dev}} \cup \mathcal{D}_{\text{test}}, \quad \mathcal{D}_{\text{dev}} \cap \mathcal{D}_{\text{test}} = \varnothing \tag{2.16}$$

When evaluating baseline stability across stochastic training runs, total performance variance can be decomposed into between-split variance ($\hat{\sigma}_{\text{split}}$) and within-split seed variance ($\hat{\sigma}_{\text{seed}}$). For a design evaluating J data partitions and S random seeds per partition, these components are defined mathematically as:

$$\hat{\sigma}_{\text{split}} = \text{std}(\bar{\mu}_j)\,, \quad \text{where } \bar{\mu}_j = \frac{1}{S}\sum_{s=1}^{S} \text{MAE}_{js} \tag{2.17}$$

$$\hat{\sigma}_{\text{seed}} = \frac{1}{J}\sum_{j=1}^{J} \text{std}(\text{MAE}_{j1}, \ldots, \text{MAE}_{jS}) \tag{2.18}$$

2.6.2 Statistical power and sample size determination

To control Type II errors (missing a genuine improvement), the required number of data partitions n must be determined via statistical power analysis [26]. Under a repeated-split design, between-split variance is controlled via pairing, rendering the within-split seed variance the primary source of experimental noise. For a targeted true effect size δ, the expected difference Δ_j across partitions is modeled as a normal distribution via Monte Carlo estimation frameworks [27]:

$$\Delta_j \sim \mathcal{N}\left(-\delta, \frac{\sigma^2_{\text{seed}}}{S}\right), \quad j = 1, \ldots, n \tag{2.19}$$

The empirical statistical power over M simulated Monte Carlo experiments is computed by evaluating the rejection rate of a one-tailed z-test under the alternative hypothesis $H_1 : \mu_\Delta < 0$:

$$\text{Power}(n_{\text{MC}}, \delta) = \frac{1}{M} \sum_{k=1}^{M} \mathbb{I}\left(p^{(k)} < \alpha\right) \tag{2.20}$$

where $\mathbb{I}(\cdot)$ is the indicator function and $p^{(k)}$ is the p-value of the k-th simulation. Alternatively, a conservative closed-form analytical bound for a one-sided test is approximated by:

$$n_{\text{analytical}} \approx \left(\frac{(z_{1-\alpha} + z_{1-\beta})\, \sigma_{\text{seed}}/\sqrt{S}}{\delta}\right)^2 \tag{2.21}$$

2.6.3 Confirmatory validation via linear mixed-effects models

To evaluate architectural interventions without violating the independence assumptions of standard parametric tests, a Linear Mixed-Effects (LME) model is employed to treat data partitions as random factors [28]. For a given performance metric, the observation y_{ijk} for split j, seed i, and condition $k \in \{0, 1\}$ (representing baseline and modified architectures, respectively) is modeled as:

$$y_{ijk} = \beta_0 + \beta_1 \text{Module}_k + u_j + \varepsilon_{ijk} \tag{2.22}$$

where β_1 isolates the fixed effect of the architectural modification, $u_j \sim \mathcal{N}(0, \sigma^2_{\text{split}})$ represents the random intercept capturing individual partition difficulty, and ε_{ijk} represents the residual stochastic seed variance.
When conducting simultaneous significance tests across a candidate registry, the family-wise error rate is explicitly controlled using the Benjamini-Hochberg False Discovery Rate (FDR) adjustment procedure [29]. To complement statistical significance with practical relevance, effect size magnitudes are calculated and assessed using Cohen's standardized d_z distance metrics [26].

2.7 Knowledge distillation

Knowledge distillation compresses a high-capacity teacher model into a compact student network by forcing the student to mimic the teacher's continuous output

distribution space [30]. The joint objective function balances standard ground-truth supervision with soft-target matching:

$$\mathcal{L} = \alpha \mathcal{L}_{\text{hard}}(\hat{y}, y) + (1 - \alpha)\mathcal{L}_{\text{soft}} \left(\frac{z}{T}, \frac{z_{\text{teacher}}}{T} \right) T^2 \tag{2.23}$$

where $\alpha \in [0, 1]$ balances the loss terms, y is the ground-truth label, and z and z_{teacher} are the unnormalized log-probabilities of the student and teacher models, respectively. The parameter T represents the temperature scaling factor. Raising $T > 1$ smooths the softmax output distribution, amplifying the "dark knowledge" or dark features contained in the negative targets of the teacher. The soft loss $\mathcal{L}_{\text{soft}}$ is implemented as the Kullback-Leibler divergence between these softened distributions. The scaling multiplier T^2 is required to compensate for the fact that the gradients of $\mathcal{L}_{\text{soft}}$ scale down by $1/T^2$ during backpropagation, thereby maintaining a consistent balance between the hard and soft loss components regardless of the chosen temperature.

3
Methods

This chapter details the approach used to develop, optimize, and deploy the excavator dynamics predictor. The workflow spans the entire development cycle, moving from initial data synthesis and model selection to the final optimized model.
First, the data foundation is established in section 3.1, covering collection, feature selection, and temporal windowing. Section 3.2 introduces the structural components of the models, addressing the shared model components and the deep learning backbones. The core of the experimental work is found in section 3.3, which explains the phased search strategy: a multi-step training methodology used to systematically evaluate module interactions and optimize hyperparameters. Furthermore, section 3.4 details the transfer learning strategy for adapting models to real-world data. The chapter concludes with the criteria used for evaluation (section 3.5) and the technical pipeline used to export and integrate (section 3.6) the models into the Simulink environment for practical application.

3.1 Data

The success of the system identification process relies on the quality and informational richness of the training data. For a data-driven model to accurately generalize across the excavator's full dynamical range, the input signals must be sufficiently rich in frequency content and amplitude variation to ensure that all relevant dynamic modes of the system are observable in the recorded response.
Given the extensive set of available signals, a selection process was employed to identify the most influential input features. These signals were subsequently preprocessed and normalized, for standardizing the input space and improving the convergence characteristics of the training algorithms.
The complete data set $\mathcal{D}$ consists of several sessions from the same simulation environment $\mathcal{D} = \{R_1, R_2, ..., R_N\}$, where each run consists of timeseries data, $R_i = \{(x_t, y_t)\}_{t=1}^{T_i}$ where:

- $x_t \in \mathbb{R}^d$: input vector at time t
- $y_t \in \mathbb{R}^m$ target variable at time t

3.1.1 Collection

Data was collected from two sources: a high-fidelity Simulink simulation of the excavator system, and a physical excavator instrumented with a Speedgoat real-time target computer. Using both simulated and physical data serves two purposes.

First, the simulation environment enables the generation of large, systematically excited datasets under controlled conditions. Second, the physical data introduces the unmodeled sensor noise, and environmental disturbances that are absent in simulation, enabling the investigation of the sim-to-real transfer gap.

3.1.1.1 Simulation data

Simulated data was generated using CPAC Systems' proprietary Simulink model of the excavator's mechanical and electro-hydraulic system. The model captures the full hydraulic circuit, including pump dynamics, main control valve pressure-flow characteristics, cylinder chamber pressures, and the mechanical kinematics of the boom, arm, bucket, and swing joints. The simulation model is based on the Volvo EC210F.
The data was collected through manual operation of the excavator within the Simulink environment, using a joystick. To achieve sufficient excitation, the joints were actuated both in isolation and in simultaneous combinations. Isolated single-joint movements were necessary to capture each actuator's individual dynamic response, while coordinated multi-joint trajectories were needed to capture the nonlinear cross-coupling effects between the hydraulic cylinders. Care was also taken to drive the actuators near their mechanical end-stops and through rapid direction reversals, as these operating regions exhibit strongly nonlinear valve and cylinder behavior that the model must learn.
An automated excitation strategy based on APRBS combined with chirp signals was investigated as a means of generating more systematically excited datasets. The APRBS would provide broadband amplitude variation mimicking operator commands, while a superimposed frequency sweep would ensure coverage of higher-frequency hydraulic resonances. However, due to time constraints, a sufficiently well-tuned signal was not realized, and all simulation data used for training was collected through manual operation.
Over the course of the project, several hours of simulation data was recorded, yielding a dataset that spans a wide variety of operating conditions including isolated single-joint movements, coordinated multi-joint trajectories, and extreme actuator positions near the mechanical end-stops.

3.1.1.2 Speedgoat data

To evaluate the model's ability to generalize from simulation to reality, operational data was collected from a physical excavator using a Speedgoat real-time target computer. The Speedgoat executes the same controller model that runs in the Simulink simulation, deployed as a real-time application via Simulink Real-Time. This architectural overlap is a key advantage: the internal controller signals (pilot pressure demands, valve commands, kinematic computations) are available for logging on the physical machine with the same naming convention and signal semantics as in simulation, which greatly simplifies the alignment between the two data domains.
The physical excavator utilized was a Volvo ECR145F. Because this differs from the model used in the simulation, it allowed for evaluation across different excavator platforms in addition to the primary sim-to-real performance investigation.

Several logging sessions were conducted under different operating conditions:

- **Free-air operation:** The excavator was operated in open air without ground contact, matching the simulation conditions.
- **Soil interaction:** The excavator performed digging and grading tasks in soil, introducing the external load disturbances that are absent from the simulation model.

This distinction is central to the sim-to-real investigation objective: the free-air logs provide a direct comparison baseline against the simulation-trained model, while the soil interaction logs test the model's robustness to unmodeled external forces.

3.1.2 Feature selection pipeline

The raw telemetry data collected from the Simulink excavator model yielded a high-dimensional, highly correlated initial dataset. To prevent overfitting, eliminate redundant variables, and minimize computational overhead in the neural network, a multi-stage feature selection pipeline was engineered to isolate an optimal subset of system inputs.

- **Linear and nonlinear dependency filtering:** While hydraulic systems are fundamentally nonlinear, Pearson's correlation and Spearman rank correlation was first applied as a baseline filter to isolate signals exhibiting direct proportional relationships. To capture the nonlinear characteristic of the excavator, mutual information ($I(X;Y)$) scores were computed. Signals that generated low scores across both metrics demonstrated negligible predictive power and were systematically prioritized for removal.
- **Temporal alignment via lag analysis:** To ensure that the sequence-based models correctly captured the excavator's hydraulic memory, cross-correlation functions $R_{xy}(\tau)$ were utilized to calculate the actual physical response lag between control inputs and system outputs. This lag analysis informed the sizing of the look-back window W and ensured that historical time steps provided to the networks were physically synchronized.
- **Cross-file stability assessment:** A prominent risk when utilizing simulation data is selecting features that appear highly relevant due to artifacts of a single test scenario rather than the true generalized behavior of the machine. To mitigate this, the correlation and mutual information metrics were evaluated for cross-file stability across multiple distinct operating datasets. Signals displaying high variance in their importance metrics across differing excavator maneuvers were penalized, ensuring the finalized input set would generalize well to unseen operational profiles.
- **Multivariate coupling analysis:** As a final validation step, a Random Forest regressor was trained on the filtered input candidate set to compute multivariate permutation feature importance (I_j). This stage was important for capturing the cross-coupling effects inherent between the excavator's three interacting hydraulic cylinders. By evaluating features in a multivariate context, signals that appeared redundant in isolation but provided context for cylinder interactions were retained.

3.1.2.1 Feature groups

Using a combination of correlation analysis with the target signal and hydraulic domain knowledge, two input feature groups were constructed. Feature Group 1 (FG1) consists of the signal listing given in table A.1 in chapter A.
Feature Group 2 (FG2) augments FG1 with explicit first-order pressure time-derivatives, defined for each monitored cylinder index $i \in \mathcal{I} = \{1, 2, 3, 5, 6, 7\}$ as

$$\Delta P_i[t] = P_i[t] - P_i[t-1] \tag{3.1}$$

yielding the derived signal set

$$\mathbf{\Delta}_{\text{pressure}} = \left\{ \Delta P_i[\cdot] \right\}_{i \in \mathcal{I}} \tag{3.2}$$

FG2 is then constructed as

$$\text{FG2} = \text{FG1} \cup \mathbf{\Delta}_{\text{pressure}} \tag{3.3}$$

3.1.3 Normalization

A challenge in performing system identification on a hydraulic excavator was the variation in the scales and units of the measured physical quantities. For instance, joint angles were measured in radians, whereas cylinder forces reached tens of thousands of Newton (10^5 N). Feeding this raw, multi-modal data directly into a neural network would lead to that features with naturally large magnitudes would dominate the gradient descent updates during backpropagation. To get stable training and faster convergence, the input and output spaces were normalized.
Two primary scaling techniques were utilized to transform the dataset into a uniform numerical range before training:

- **Min-Max scaling:** This technique linearly transforms the data such that all values fall within a bounded interval, typically $[-1, 1]$ or $[0, 1]$. For a given feature x, the scaled value x_{scaled} is calculated as:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{3.4}$$

  Min-max scaling was used for signals with strictly defined physical limits, such as the current length of the cylinder.
- **Standardization (Z-score normalization):** Instead of strictly bounding the data, standardization rescales the features so they possess a mean (μ) of 0 and a standard deviation (σ) of 1:

$$x_{scaled} = \frac{x - \mu}{\sigma} \tag{3.5}$$

  Standardization was used for unbounded physical states or sensor readings containing natural Gaussian noise, such as hydraulic pressures and cylinder velocities because it is less sensitive to extreme outlier measurements than min-max scaling.

3.1.4 Windowing

To train the model for sequential, real-time prediction, the continuous time-series data was partitioned into fixed-length input sequences using a sliding window approach. Given a transformed feature matrix $\mathbf{X} \in \mathbb{R}^{N \times F}$, where N is the number of time steps and F the number of features, a window of size W is slid across the time axis with a stride of one. At each time step t, the model receives the W most recent observations $\mathbf{X}[t-W+1 : t] \in \mathbb{R}^{W \times F}$ and is trained to predict the target state at the next timestep $t+1$. This transforms the continuous time series into a set of overlapping input–output pairs suitable for supervised learning with mini-batch gradient descent.
The window size W represents the temporal context available to the model at each prediction step, and its selection involves a trade-off. A sufficiently large window is necessary for the model to observe the full transient response of the hydraulic system, including the propagation delay between a joystick command and the resulting cylinder movement, which spans several hundred milliseconds. However, an excessively large window increases the computational cost and memory footprint of each forward pass, which is detrimental to real-time inference. At the default sample rate of 100 Hz, a window size of $W = 100$ corresponds to one second of history.

3.2 Modeling

All models in this work follow a common prediction formulation. Given a windowed input sequence $\mathbf{X} = [\mathbf{x}_{t-W+1}, \ldots, \mathbf{x}_t] \in \mathbb{R}^{W \times d}$ of W time steps and d input features, the model predicts the cylinder velocities one or several steps ahead.
The general prediction takes the form:

$$\hat{\mathbf{y}}_{t+k} = g\Big(\underbrace{\mathbf{z}_{\text{phys}}(\mathbf{X})}_{\text{physics base (optional)}} + \underbrace{f_\theta(\mathbf{X})}_{\text{learned residual}} \Big) \tag{3.6}$$

where f_θ is a neural network backbone with parameters θ, $\mathbf{z}_{\text{phys}}$ is an optional deterministic physics prediction, and $g(\cdot)$ is an output activation function. When the physics base is disabled, the model reduces to $\hat{\mathbf{y}}_{t+1} = g(f_\theta(\mathbf{X}))$.
The remainder of this section describes the shared components used across all architectures (section 3.2.2), followed by the three backbone architectures (section 3.2.3) and lastly regularization techniques (section 3.2.4).

3.2.1 Target formulation

Two target encodings are considered:

- **Absolute mode**: the model predicts the velocity directly, $\hat{\mathbf{y}}_{t+k} = \hat{\mathbf{v}}_{t+k} \in \mathbb{R}^3$.
- **Delta mode**: the model predicts the velocity *change*, $\hat{\mathbf{y}}_{t+k} = \hat{\mathbf{v}}_{t+k} - \mathbf{v}_t \in \mathbb{R}^3$.

Delta mode encodes an inductive bias: by predicting changes rather than absolute values, the model is biased toward learning the system's dynamics rather than memorizing operating-point offsets. This is analogous to residual learning in that the identity mapping is the default prediction. All targets and predictions operate in standardized space.

3.2.2 Shared model components

The components described below can be integrated with any network backbone and are evaluated independently during the hyperparameter optimization process.

3.2.2.1 Physics base

The physics base provides a deterministic, parameter-free prediction $\mathbf{z}_{\text{phys}}$ derived from the excavator's kinematics, which the neural network then corrects. This residual-learning formulation biases the model toward physically plausible solutions and reduces the function the network must learn to unmodeled effects such as friction, valve dynamics, and oil compressibility.
Three physics-base variants are implemented, each corresponding to a different level of physical modeling:

Velocity (kinematic): The simplest variant computes the cylinder velocity directly from joint velocities $\dot{\mathbf{q}}$ and the velocity Jacobian matrix $\mathbf{J}_v(\mathbf{q})$:

$$\hat{v}_i(t + \Delta t) = J_{v,i}(\mathbf{q}_t) \cdot \dot{q}_{i,t}, \qquad i \in \{\text{boom, arm, bucket}\} \tag{3.7}$$

where $J_{v,i}$ is the kinematic leverage ratio mapping joint velocity to cylinder velocity for actuator i. The operation is executed element-wise, as each cylinder displacement corresponds directly to its respective joint angle.

Force integration (Euler step): This variant estimates the change in velocity resulting from hydraulic cylinder forces using a forward Euler integration step under a constant equivalent mass assumption:

$$\hat{v}_i(t + \Delta t) = v_{i,t} + \frac{F_{\text{cyl},i,t} \cdot \Delta t}{\bar{M}_{\text{eq},i}} \tag{3.8}$$

where $F_{\text{cyl},i,t}$ is the net cylinder force, $\Delta t = 0.01\,\text{s}$ is the sampling period ($100\,\text{Hz}$), and $\bar{M}_{\text{eq},i}$ is the time-averaged equivalent mass for actuator i.

Kinematic-dynamic hybrid: The most comprehensive variant combines both kinematic constraints and dynamic force integration via a first-order update equation:

$$\hat{v}_i(t + \Delta t) = \underbrace{J_{v,i}(\mathbf{q}_t)\, \dot{q}_{i,t}}_{\text{Kinematic State}} + \underbrace{\frac{F_{\text{cyl},i,t}}{M_{\text{eq},i}(\mathbf{q}_t)}\, \Delta t}_{\text{Dynamic Correction}} \tag{3.9}$$

where $M_{\text{eq},i}(\mathbf{q}_t)$ is the configuration-dependent equivalent mass matrix evaluated at the current joint positions.

3.2.2.2 Structural modules

The following modular components are toggled independently during hyperparameter screening. They are presented in the order they appear within the computational graph.

Multi-scale TCN: The baseline TCN utilizes a single sequence of dilated convolutional layers with exponentially increasing dilation factors $(1, 2, 4, \ldots)$. The multi-scale variant splits this architecture into two parallel branches: a *fine* branch with a standard dilation sequence $(1, 2, 4, \ldots)$ to capture high-frequency pressure transients and valve transitions, and a *coarse* branch starting at a baseline dilation of 4 $(4, 8, 16, \ldots)$ to cover an expanded receptive field for slow-varying dynamics like load trends and position drift. The outputs of both branches are concatenated along the channel dimension before being forwarded to the recurrent layer.

Attention bridge: An optional transformer-style processing block can be inserted between the TCN and the recurrent layers to process the full TCN output sequence $\mathbf{R} \in \mathbb{R}^{T \times C}$. The block consists of a Multi-Head Self-Attention (MHA) sublayer followed by a position-wise Feed-Forward Network (FFN), utilizing pre-layer normalization (LN) and residual connections:

$$\mathbf{R}' = \mathbf{R} + \mathrm{MHA}\Big(\mathrm{LN}(\mathbf{R}),\, \mathrm{LN}(\mathbf{R}),\, \mathrm{LN}(\mathbf{R})\Big) \tag{3.10}$$

$$\mathbf{R}'' = \mathbf{R}' + \mathrm{FFN}\Big(\mathrm{LN}(\mathbf{R}')\Big) \tag{3.11}$$

where MHA uses h attention heads and FFN is a two-layer network with hidden dimension $2C$ and GELU activations. The output $\mathbf{R}''$ preserves the original sequence length while enriching each time step with global context across the temporal window.

GRU backbone: As an alternative to the standard LSTM, a Gated Recurrent Unit (GRU) can be selected. The GRU combines the forget and input gates into a single update gate and removes the separate cell state:

$$\mathbf{r}_t = \sigma(\mathbf{W}_r \mathbf{x}_t + \mathbf{U}_r \mathbf{h}_{t-1}) \tag{3.12}$$

$$\mathbf{z}_t = \sigma(\mathbf{W}_z \mathbf{x}_t + \mathbf{U}_z \mathbf{h}_{t-1}) \tag{3.13}$$

$$\tilde{\mathbf{h}}_t = \tanh\Big(\mathbf{W}_h \mathbf{x}_t + \mathbf{U}_h(\mathbf{r}_t \odot \mathbf{h}_{t-1})\Big) \tag{3.14}$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t \tag{3.15}$$

This architecture reduces the recurrent parameter footprint by approximately 25% relative to an LSTM of equivalent hidden size, which can accelerate training convergence on moderately long sequences.

Temporal differencing: To explicitly expose rate-of-change indicators to the output layers, the first-order difference of the recurrent hidden states across the final two time steps is concatenated directly to the latent backbone representation $\mathbf{h}$:

$$\mathbf{h}_{\mathrm{aug}} = [\, \mathbf{h} \;;\; \mathbf{h}_t - \mathbf{h}_{t-1} \,] \tag{3.16}$$

This augmentation provides an explicit velocity-equivalent signal in the latent space, reducing the network's reliance on implicitly learned internal temporal derivatives.

Squeeze-and-excitation block: To explicitly model channel-wise inter-dependencies, a Squeeze-and-Excitation (SE) block adaptively recalibrate the TCN backbone features. First, a global channel descriptor $\mathbf{z}$ aggregates the feature sequence $\{\mathbf{u}_t\}_{t=1}^{T}$ via temporal average pooling:

$$\mathbf{z} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{u}_t \tag{3.17}$$

A two-layer bottleneck network then computes channel modulation weights $\mathbf{s}$ to scale the original features ($\widetilde{\mathbf{u}}_t = \mathbf{s} \odot \mathbf{u}_t$):

$$\mathbf{s} = \sigma(\mathbf{W}_2\delta(\mathbf{W}_1\mathbf{z})) \tag{3.18}$$

This adaptive gating allows the network to emphasize globally informative feature channels while actively suppressing less relevant ones prior to the recurrent stage.

Global average pooling branch: To complement the last-step and attention-pooled features that comprise the standard backbone representation $\mathbf{h}$, an optional Global Average Pooling (GAP) branch computes a parameter-free temporal summary across the entire recurrent output sequence $\{\mathbf{h}_t\}_{t=1}^{T}$:

$$\mathbf{g} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{h}_t \tag{3.19}$$

The vector $\mathbf{g} \in \mathbb{R}^{d_h}$ is projected to a dimension of $d_h/2$ and concatenated with $\mathbf{h}$. GAP captures the mean operational state over the input window, which provides a slow-varying feature reference that acts as an implicit regularizer.

Cylinder feature gate: A specialized gating mechanism is implemented to modulate the shared backbone representation $\mathbf{h}$ individually for each cylinder output head:

$$\mathbf{h}_i^{\text{gated}} = \boldsymbol{\sigma}_i \odot \mathbf{h} \tag{3.20}$$

where $\boldsymbol{\sigma}_i \in (0,1)^D$ is a learned sigmoid gating vector. Two structural variants are evaluated:

$$\text{Linear Variant:} \quad \boldsymbol{\sigma}_i = \sigma(\mathbf{W}_i^g\mathbf{h} + \mathbf{b}_i^g) \tag{3.21}$$

$$\text{Diagonal Variant:} \quad \boldsymbol{\sigma}_i = \sigma(\mathbf{s}_i \odot \mathbf{h} + \mathbf{b}_i^g) \tag{3.22}$$

The linear variant maps a full $\mathbf{W}_i^g \in \mathbb{R}^{D\times D}$ weight matrix per cylinder at a computational cost of $\mathcal{O}(n_{\text{cyl}} \cdot D^2)$, while the diagonal variant optimizes a per-feature scale vector $\mathbf{s}_i \in \mathbb{R}^D$ at a reduced cost of $\mathcal{O}(n_{\text{cyl}} \cdot D)$. For both configurations, the bias $\mathbf{b}_i^g$ is initialized to 4.0 ($\sigma(4) \approx 0.98$). This ensures the gate is transparent at initialization, matching the performance of the ungated baseline at the beginning of training.

Cross-cylinder attention: To capture the mechanical coupling inherent to the multi-link kinematic chain (boom → arm → bucket), a multi-head self-attention layer can be applied across the cylinder dimension. This layer enables information sharing across individual cylinder streams after feature extraction but prior to final scalar projections:

$$\mathbf{C}' = \mathrm{LayerNorm}\Big(\mathbf{C} + \mathrm{MHA}(\mathbf{C}, \mathbf{C}, \mathbf{C})\Big) \tag{3.23}$$

where $\mathbf{C} \in \mathbb{R}^{3 \times d_{\text{head}}}$ is the stacked feature representation of the three cylinder channels, and MHA represents a standard single-head attention mechanism.

Velocity skip connection: An optional residual link adds the current observed velocity vector directly to the final network output:

$$\hat{\mathbf{v}}_{t+1} = \mathbf{v}_t + g(f_\theta(\mathbf{X})) \tag{3.24}$$

This structure configures the absolute execution mode as a residual learning problem, biasing the neural network $f_\theta(\mathbf{X})$ to predict small delta modifications to the current velocity state. This module is automatically disabled when running in delta mode to avoid structural redundancy.

3.2.2.3 Per-cylinder output heads

The hydraulic excavator features $C = 3$ independently actuated subsystems: the boom, arm, and bucket cylinders. Mapping a shared backbone feature space $\mathbf{h}$ to all actuators simultaneously via a single linear layer would force the network to compromise between distinct physical subsystems. Instead, the model employs a decoupled multi-head architecture, instantiating a dedicated MLP head for each individual cylinder $c \in \{1, \ldots, C\}$:

$$\hat{y}_c = \mathrm{Head}_c(\mathbf{h}), \quad c = 1, \ldots, C \tag{3.25}$$

This design allows the shared backbone to isolate globally relevant, system-wide dynamics, while permitting individual output heads to specialize in the unique load characteristics, pressure profiles, and hydraulic behaviors of their respective actuators. For example, the boom cylinder operates under significantly higher load forces and lower velocities than the highly transient bucket cylinder.
In the baseline configuration, each head is parameterized as a two-layer feed-forward network with a single hidden layer:

$$\hat{y}_c = \mathbf{w}_c^\top \,\mathrm{ReLU}\Big(\mathbf{W}_c\,\mathbf{h} + \mathbf{b}_c\Big) + b'_c \tag{3.26}$$

where $\mathbf{W}_c \in \mathbb{R}^{d' \times D}$ and $\mathbf{b}_c \in \mathbb{R}^{d'}$ represent the weights and biases of the hidden projection layer with dimension d', while $\mathbf{w}_c \in \mathbb{R}^{d'}$ and $b'_c \in \mathbb{R}$ define the final linear output mapping.
To evaluate how head capacity impacts the tracking of severe physical non-linearities, two alternative architectural variants are implemented for Head_c:

Gated Linear Unit (GLU) head: The standard ReLU activation is replaced by a GLU mechanism. The initial projection width is doubled to $2d'$, mapping the backbone representation to a vector $\mathbf{u} = \mathbf{W}_c\mathbf{h} \in \mathbb{R}^{2d'}$. The output is then obtained via element-wise sigmoid gating of the split vector halves:

$$\mathrm{GLU}(\mathbf{u}) = \mathbf{u}_1 \odot \sigma(\mathbf{u}_2) \tag{3.27}$$

where $\mathbf{u}_1, \mathbf{u}_2 \in \mathbb{R}^{d'}$, and $\sigma(\cdot)$ is the logistic sigmoid function. The data-driven gating mechanism selectively suppresses or amplifies components of the latent representation. This behavior is physically well-suited to capturing hydraulic dead-band nonlinearities, where valve spool movement yields near-zero actuator response until a critical control input threshold is surpassed.

Deeper MLP head: An additional hidden layer of width d' is inserted prior to the output mapping, yielding a deeper feed-forward structure. This increases the head's capacity to model nonlinear input–output relationships at the cost of additional parameters.

3.2.2.4 Output activation functions

The output activation $g(\cdot)$ in eq. (3.6) controls the range of predictions from the per-cylinder heads. Three options are considered:

Identity (linear). $g(x) = x$. No bound on the output range. The simplest option but allows unbounded predictions, which can be problematic during early training.

Tanh. $g(x) = \tanh(x)$. Bounds outputs to $(-1, 1)$ in normalized space, corresponding to approximately $\pm 1\sigma$ of the target distribution. This can be too restrictive for target distributions with heavy tails, as rare high-velocity events are clipped.

Scaled Tanh (STanh). A learnable generalization of tanh with per-cylinder amplitude:

$$g_i(x) = a_i \cdot \tanh(x), \qquad a_i \in [a_{\min},\, a_{\max}] \tag{3.28}$$

where a_i is a learnable parameter clamped to $[a_{\min}, a_{\max}]$. This bounds each cylinder's output to $(-a_i, a_i)$ while allowing the model to adapt the range per cylinder. Clamping prevents degenerate solutions where $a_i \to 0$ or $a_i \to \infty$.

3.2.2.5 Loss function

Two base loss functions are considered. Mean Squared Error (MSE):

$$\mathcal{L}_{\mathrm{MSE}} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{w}^\top(\hat{\mathbf{y}}_n - \mathbf{y}_n)^2 \tag{3.29}$$

and Huber loss, which transitions from quadratic to linear behavior at a threshold δ:

$$\ell_\delta(e) = \begin{cases} \frac{1}{2}e^2 & \text{if } |e| \le \delta \\ \delta\left(|e| - \frac{1}{2}\delta\right) & \text{otherwise} \end{cases} \tag{3.30}$$

The Huber loss provides robustness to outliers, since large errors from sensor noise or transient spikes receive a linear rather than a quadratic penalty, reducing their influence on gradient updates.

Per-output weighting. Both losses are scaled by per-output weights $\mathbf{w} \in \mathbb{R}^3$ initialized from inverse target variance:

$$w_i \propto \frac{1}{\sigma_i^p}, \qquad \sum_i w_i = 3 \tag{3.31}$$

where σ_i is the standard deviation of target i and p is a configurable power. This ensures that cylinders with lower variance (typically boom) receive proportionally higher weight, preventing the loss from being dominated by the high-variance bucket cylinder.

3.2.3 Backbone architectures

Three backbone architectures for f_θ are evaluated. All share the output structure described above (per-cylinder heads, optional modules, activation, physics base) and differ only in how they process the input sequence $\mathbf{X}$ into a fixed-dimensional representation $\mathbf{h} \in \mathbb{R}^D$.

3.2.3.1 LSTM backbone (with attention pooling)

The LSTM backbone processes the sequential input recurrently to capture the system's underlying temporal dependencies. The scaled input features $\mathbf{X}_{\text{scaled}} \in \mathbb{R}^{W \times d_s}$ are fed through a multi-layer LSTM network to generate a sequence of hidden states:

$$\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_W] = \text{LSTM}(\mathbf{X}_{\text{scaled}}) \in \mathbb{R}^{W \times D} \tag{3.32}$$

where W is the input window size and D is the hidden state dimension.
Rather than relying solely on the terminal hidden state $\mathbf{h}_W$, which can introduce a severe recency bias and cause the model to forget early sequence events, a temporal attention pooling layer is integrated. This layer computes a learned, weighted summary across the entire sequence of hidden states to capture globally salient features:

$$\text{AttnPool}(\mathbf{H}) = \tilde{\mathbf{h}} = \sum_{t=1}^{W} \alpha_t \mathbf{h}_t \tag{3.33}$$

The attention weights α_t represent the relative importance of each timestep and are calculated using a softmax activation over the inner product of the hidden states with a learnable parameter vector:

$$\alpha_t = \frac{\exp(\mathbf{w}_a^\top \mathbf{h}_t)}{\sum_{s=1}^{W} \exp(\mathbf{w}_a^\top \mathbf{h}_s)} \tag{3.34}$$

where $\mathbf{w}_a \in \mathbb{R}^D$ acts as a learnable template of "importance" in the feature space. During training, the network optimizes $\mathbf{w}_a$ to align with the specific hidden states that precede significant structural changes in the excavator's operating state. This mechanism explicitly allows the model to attend to highly informative time steps

rather than relying entirely on the recurrent mechanics to propagate that information over long horizons.
The final fixed-dimensional backbone representation $\mathbf{h}$ is constructed by concatenating the attention-pooled summary with the terminal hidden state:

$$\mathbf{h} = \left[\tilde{\mathbf{h}} \; ; \; \mathbf{h}_W \right] \in \mathbb{R}^{2D} \tag{3.35}$$

This dual aggregation strategy ensures that the downstream output heads preserve both the globally salient event context ($\tilde{\mathbf{h}}$) and the absolute end-of-sequence boundary conditions ($\mathbf{h}_W$).

3.2.3.2 TCN backbone

The TCN backbone shifts away from recurrent state updates, utilizing a deep stack of causal dilated 1D convolutions to extract sequential features in parallel. The baseline TCN processes the scaled input sequence $\mathbf{X}_{\text{scaled}} \in \mathbb{R}^{W \times d_s}$ through L sequential temporal blocks. To build upon the generic residual structure defined in eq. (2.10), each of the temporal blocks explicitly instantiates two consecutive layers of dilated convolutions. These layers are configured with causal padding, ReLU activations, and dropout regularizers:

$$\mathbf{Z}^{(l)} = \text{TemporalBlock}_l\Big(\mathbf{Z}^{(l-1)}\Big) + \mathbf{Z}^{(l-1)}, \quad l = 1, \ldots, L \tag{3.36}$$

where $\mathbf{Z}^{(0)} = \mathbf{X}_{\text{scaled}}$. Weight normalization is applied to all convolutional filters within the blocks to stabilize the distribution of internal activations and accelerate network convergence during stochastic gradient descent.
By utilizing an exponentially increasing dilation factor $d_l = 2^{l-1}$ at layer l, the network expands its history window efficiently without the parameter explosion or computational overhead associated with large kernel sizes. Because each residual block contains two internal convolutional operations, expanding the generic receptive field definition from eq. (2.9) yields the total effective receptive field R:

$$R = 1 + 2(k-1)\Big(2^L - 1\Big) \tag{3.37}$$

where k represents the kernel size. During network training and deployment, the hyperparameters k and L are automatically configured to ensure that the receptive field spans or exceeds the designated input window size ($R \geq W$).
While the LSTM baseline requires an explicit attention pooling mechanism to combat memory degradation over time, the standalone TCN backbone extracts its final latent representation $\mathbf{h}$ solely from the terminal timestep of the final layer:

$$\mathbf{h} = \mathbf{z}_W^{(L)} \in \mathbb{R}^{C_L} \tag{3.38}$$

where C_L represents the feature channel width of the final layer. This single-point feature extraction is mathematically justified by the structural constraint $R \geq W$. Because the receptive field covers the entire sequence window, the individual feature vector $\mathbf{z}_W^{(L)}$ at the absolute boundary timestep W inherently possesses a complete effective history, having integrated information from all previous hydraulic states through the hierarchy of dilated connections.

3.2.3.3 Hybrid TCN-LSTM

The proposed hybrid architecture sequentially combines both backbones to address the complementary physical limitations of each individual model topology. While the TCN component efficiently parses raw, high-dimensional inputs and extracts local high-frequency patterns across a wide temporal window via parallelized convolutions, it lacks a recurrent hidden state capable of tracking state evolution across long sequences. Conversely, the LSTM tracks continuous state trajectories well but struggles to efficiently process long, uncompressed structural feature representations over extended horizons. By stacking these topologies, the hybrid model utilizes the TCN as a localized semantic compressor, smoothing out complex high-frequency transients before passing a condensed sequence representation to a downstream recurrent layer.
In practice, for a scaled input sequence $\mathbf{X}_{\text{scaled}} \in \mathbb{R}^{W \times d_s}$, the TCN backbone applies L stacked residual blocks configured according to the structure described in section 3.2.3.2. The resulting output sequence matrix $\mathbf{Z}^{(L)} \in \mathbb{R}^{W \times C_L}$ encodes the localized temporal context of the input into C_L distinct feature channels, capturing rapid physical system variations.
The sequence of feature vectors $\mathbf{z}_t^{(L)} \in \mathbb{R}^{C_L}$ (comprising the rows of $\mathbf{Z}^{(L)}$) is subsequently fed sequentially into a single-layer LSTM. The hidden-state sequence matrix $\mathbf{H}$ evolves over the sequence window according to the exact cell gating mechanisms established in the theoretical foundation:

$$\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_W] = \text{LSTM}\Big(\mathbf{Z}^{(L)}\Big) \in \mathbb{R}^{W \times D} \tag{3.39}$$

Because $\mathbf{z}_t^{(L)}$ represents a rich, lower-dimensional encoding of local operating history, the effective complexity of the sequence passed to the recurrent block is dramatically reduced. This enables the single-layer LSTM to maintain longer-range tracking capabilities while maintaining low additional computational and recurrent overhead. The final fixed-dimensional representation $\mathbf{h}$ is formed identically to the standalone LSTM baseline, combining the absolute boundary context with the temporal attention pooling weights derived across the sequence:

$$\mathbf{h} = \Big[\ \text{AttnPool}(\mathbf{H})\ ;\ \mathbf{h}_W\ \Big] \in \mathbb{R}^{2D} \tag{3.40}$$

To evaluate how multi-scale feature parsing affects downstream estimation, an optional multi-scale TCN variant is implemented. This structural variant runs two parallel TCN branches over the raw input with differing initial dilation bases. The fine branch initializes with a standard dilation factor ($d_0 = 1$) to capture immediate high-frequency changes, while the coarse branch initializes with an expanded dilation factor ($d_0 = 4$) to capture broader temporal contexts. The resulting matrix outputs from both parallel branches are concatenated along the channel dimension prior to entering the LSTM layer, doubling the hidden representation width to $2C_L$ entering the recurrent network.

3.2.4 Regularization techniques

Regularization is applied at multiple levels throughout the training pipeline to control over-fitting and to improve generalization across the operating conditions repre-

sented in the simulation dataset. The techniques can be divided into three groups: architectural regularization that is built into the model, optimizer-level regularization applied during gradient descent, and loss-function regularization that shapes the learning objective itself. Additional techniques that are specific to fine-tuning on real machine data are described separately in section 3.4.5.

3.2.4.1 Architectural regularization

Dropout: Standard dropout is applied to the hidden activations of all temporal encoder layers with a rate p_{drop} that is included in the Optuna hyperparameter search space. During each forward pass in training mode, each hidden unit is independently zeroed with probability p_{drop}, preventing co-adaptation between units and forcing the model to learn redundant representations of the hydraulic state. Dropout is disabled during inference.

Zoneout: The LSTM architecture can replace standard dropout with zoneout. Rather than zeroing activations, zoneout stochastically retains the previous hidden state h_{t-1} and cell state c_{t-1} instead of updating them. During training, each unit independently keeps its previous value with a probability p_{zone}; during evaluation the states are replaced by the expectation $p_{\text{zone}}\, h_{t-1} + (1 - p_{\text{zone}})\, h_t$. This preserves temporal continuity across time steps and maintains unobstructed gradient flow through time, mitigating the vanishing-gradient problem.

Layer normalization: Layer normalization is applied to the gate pre-activations inside the LSTM cells. By normalizing over the hidden dimension at each time step, it prevents the internal covariate shift that arises from the highly non-stationary hydraulic pressure and velocity signals, which can otherwise cause gate saturation and gradient instability in standard LSTM cells.

3.2.4.2 Optimizer-level regularization

Decoupled weight decay: All models are trained with the AdamW optimizer, which applies weight decay directly to the parameters rather than coupling it to the gradient update as in standard ℓ_2 regularization. The weight decay coefficient λ is included in the Optuna search space, allowing the search to select the strength of the implicit ℓ_2 norm penalty appropriate for each architecture.

Gradient clipping: Gradients are clipped to a maximum global ℓ_2 norm before every parameter update. This prevents the occasional large gradient spikes that arise from abrupt hydraulic transients in the data from destabilizing training, and is particularly important for the recurrent architectures where back-propagation through time can amplify gradient magnitudes.

Learning-rate scheduling: The learning rate is decayed over training using either a cosine annealing schedule or a *ReduceLROnPlateau* scheduler, with the scheduler type and patience determined by the Optuna search. Decaying the learning rate as training progresses allows the optimizer to take large steps in the early exploration phase and progressively refine parameter estimates as

the model converges, reducing the risk of oscillating around a minimum.

Early stopping: Training is terminated if the validation loss does not improve for a fixed number of consecutive epochs. This prevents the model from memorizing the specific simulation trajectories in the training split while failing to generalize to the validation split, and acts as an implicit upper bound on model complexity by limiting the number of gradient updates seen.

Stochastic weight averaging: Stochastic Weight Averaging (SWA) is available as an optional regularizer during simulation pre-training. In the final 20% of training epochs, a running average of the model weights is maintained under a constant low learning rate η_{SWA}. The averaged weights typically lie closer to the center of a flat loss basin than any single checkpoint, improving generalization without additional inference cost.

3.2.4.3 Loss-function regularization

Huber loss: The primary training objective uses the Huber loss (see section 3.2.2.5). Below a learnable threshold δ the loss is quadratic (matching MSE), and above δ it grows only linearly, making the objective robust to occasional large errors caused by sudden valve actuation or sensor glitches in the simulation data.

Smoothness regularization: An optional smoothness penalty weighted by λ_s is added to the primary loss. It penalizes the second-order variation of consecutive predictions, discouraging rapid oscillations in the output sequence that are inconsistent with the inertia of the hydraulic actuators. The weight λ_s is included in the Optuna search space; its optimal value converges to zero for most architectures and simulation datasets, and it is therefore not applied by default.

3.3 Proposed phased search strategy

To mitigate the risk of information leakage, uncontrolled variance, and biased model comparison, this study adopts a multi-phased search strategy. By separating hyperparameter exploration from formal confirmatory validation, we ensure computational efficiency while maintaining statistical rigor.

- **Phase 0 – Data isolation & variance estimation:** Establishes data structures, freeze the test set $\mathcal{D}_{\text{test}}$ according to eq. (2.16), calculates baseline deviations ($\hat{\sigma}_{\text{split}}$, $\hat{\sigma}_{\text{seed}}$), and selects feature group to propagate downstream.
- **Phase 1 – Baseline calibration:** Utilizes an Optuna TPE search with warmup trials to establish a well-calibrated reference configuration, selected via an efficiency heuristic.
- **Phase 2 – Exploratory module screening:** Assesses candidate architectural enhancements within a localized, elastic hyperparameter window against a multi-seed median baseline.
- **Phase 2.5 – Power analysis:** Employs Monte Carlo simulations and analytical bounds to compute the minimum repeated data splits J required to ensure statistical power for subsequent confirmatory testing.

- **Phase 3 – Confirmatory validation:** Subjects surviving modules to a paired, repeated-split experimental design over J partitions. Decisions are adjudicated via an LME model evaluated against strict significance, effect-size, and stability gates.
- **Phase 4 – Final synthesis:** Explores potential antagonistic or destructive interactions among individually confirmed modules by jointly tuning continuous hyperparameters and binary module activation flags.
- **Phase 5 – Unconstrained explanatory search:** Re-opens the entire module registry to a global combinatorial search, identifying complex, non-linear module synergies that failed standalone validation.
- **Phase 6 – Knowledge distillation:** Compresses the verified optimal architecture into a computationally lightweight student model using structured pruning, distillation objectives, and low-precision quantization.

3.3.1 Phase 0 – Data partition and variance estimation

To prevent leakage from temporal dependencies within trajectories, $\mathcal{D}$ is split at the session level into development ($\mathcal{D}_{\text{dev}}$) and test ($\mathcal{D}_{\text{test}}$) sets, following eq. (2.16). A fixed, low complexity, configuration baseline model is trained across n_{split} partitions and n_{seed} initialization seeds. The partition seed is explicitly fixed within each split to guarantee file consistency across runs.
Variance components are calculated according to eq. (2.17) and eq. (2.18). To maintain conservative safety margins, when evaluating multiple candidate backbones, the maximum observed $\hat{\sigma}_{\text{split}}$ is propagated to the sample-size calculations, providing a robust sample-size guarantee for the most variable model configuration.

3.3.2 Phase 1 – Baseline calibration and HPO

The reference configuration is established via an Optuna TPE search with a MedianPruner over 28 total trials on $\mathcal{D}_{\text{dev}}$. The optimization explores the following mixed search space:

- Learning rate: $\eta \in [10^{-4}, 5 \times 10^{-3}]$ (log uniform)
- Weight decay: $\lambda \in [10^{-5}, 10^{-2}]$ (log uniform)
- Dropout rate: $\delta \in [0.05, 0.40]$ (linear uniform)
- Hidden unit capacity: $C \in \{64, 128, 256, 512\}$
- Layer depth: $L \in \{1, 2, 3, 4\}$
- Temporal/Hybrid kernel size: $K \in \{3, 5, 7\}$
- Loss function formulation: $\mathcal{L}_{\text{type}} \in \{\text{MSE}, \text{Huber}\}$

To prevent TPE from favoring the largest units before smaller sizes are sampled, a warmup cycles through all four unit sizes for $w = 6$ rounds (24 warmup trials). The optimizer then exploits freely for the remaining 4 trials, minimizing the following composite score:

$$S_{\text{composite}} = 0.4 \cdot MAE + 0.15 \cdot RMSE + 0.25 \cdot P_{95} + 0.20 \cdot P_{99} \tag{3.41}$$

The optimal baseline configuration is chosen via an efficiency heuristic: candidate unit configurations are evaluated in ascending order of size, selecting the smallest parameter footprint that avoids diminishing performance returns.

3.3.3 Phase 2 – Exploratory module screening

Candidate architectural modules are integrated and evaluated within a constrained, elastic hyperparameter window centered around the Phase 1 calibrated baseline values $(\eta_0, \lambda_0, \delta_0)$:

$$\begin{aligned} \log_{10}(\eta) &\in \log_{10}(\eta_0) \pm 1 \\ \log_{10}(\lambda) &\in \log_{10}(\lambda_0) \pm 1 \\ \delta &\in [\delta_0 - 0.15,\ \delta_0 + 0.15] \cap [0, 1] \end{aligned}$$

Each candidate module runs for 10 trials using the frozen hidden unit capacity determined in Phase 1.

3.3.3.1 Phase 2.5 – Monte Carlo power analysis

To target an experimental power of $1 - \beta = 0.80$ at a significance level of $\alpha = 0.05$, the minimum required repeated splits J is calculated using the variance metrics derived in Phase 0. Assuming a paired repeated-split design, σ_{split} cancels out in the paired differences. For candidate split size n, $M = 3000$ synthetic experiments are drawn from the distribution defined in eq. (2.19), targeting a minimum detectable effect size of $\delta = 0.02 \times \text{MAE}_{\text{baseline}}$.
The empirical power recommendation (n_{MC}) derived from eq. (2.20) is cross-checked against the analytical approximation ($n_{\text{analytical}}$) from eq. (2.21). The definitive split allocation J is chosen via:

$$J = \min\left(\max(n_{\text{MC}},\ n_{\text{analytical}}),\ 25\right) \tag{3.42}$$

where the upper bound of 25 is determined by the maximum number of independent recording sessions available within $\mathcal{D}_{\text{dev}}$.

3.3.4 Phase 3 – Confirmatory validation

Modules surviving the screening stage undergo a paired repeated-split evaluation over J data partitions and S seeds, producing $N = J \times S$ observations. The four error metrics composing eq. (3.41) are tracked simultaneously. An LME model is fitted for each module and metric according to eq. (2.22), and a one-tailed z-test is applied to the fixed effect coefficient under $H_1 : \beta_1 < 0$.
To control the family-wise error rate across multiple simultaneous module evaluations, raw p-values are adjusted globally via the Benjamini-Hochberg false discovery rate procedure. A module is confirmed if it satisfies three strict validation gates for at least one metric:

1. **Statistical significance:** $p_{\text{BH}} < \alpha$.
2. **Effect size magnitude:** A medium Cohen's effect size ($|d_z| \geq 0.5$) along with an improvement direction ($\hat{\beta}_1 < 0$).
3. **Rank stability:** The candidate module outperforms the baseline in $\geq \frac{3}{4}$ of the N paired observations.

3.3.5 Phase 4 – Final synthesis and final evaluation

Although modules confirmed in Phase 3 have demonstrated individual benefit against the baseline, joint confirmation does not guarantee synergy: two modules each improving MAE may interact destructively when enabled simultaneously. Phase 4 therefore re-opens the confirmed modules as binary on/off candidates within the final HPO search.
An HPO step of 15 trials is performed over $\mathcal{D}_{\text{dev}}$ minimizing $S_{\text{composite}}$, fixing the structural backbone and allowing the TPE optimizer to jointly tune binary module selections alongside continuous parameters (η, λ, δ). Modules rejected at this stage are classified as antagonistic and omitted.
The highest-performing joint configuration is then initialized from scratch, trained using the full training epoch budget, and evaluated exactly once on the frozen hold-out set $\mathcal{D}_{\text{test}}$. Because $\mathcal{D}_{\text{test}}$ plays no role in hyperparameter tuning or architectural selection, this step yields an unbiased, generalization performance estimate free from optimistic selection bias.

3.3.6 Phase 5 – Unconstrained joint optimization

To catch synergistic relationships between modules that failed standalone Phase 3 validation but perform well in combination, Phase 5 expands the search to the entire module registry. This optimization unifies B binary module activation toggles, conditional hyperparameters for active modules, and global continuous parameters (η, λ, δ). The core structural backbone remains frozen to the Phase 1 configuration. To accelerate search convergence, the optimization is warm-started using the winning Phase 4 configuration as a prior, executing the joint objective:

$$\mathbf{m}^*, \boldsymbol{\theta}^* = \arg \min_{\mathbf{m} \in \{0,1\}^B, \boldsymbol{\theta}} S_{\text{composite}} \left(f(\cdot; \mathbf{m}, \boldsymbol{\theta}) \right) \tag{3.43}$$

where $\mathbf{m}$ represents the binary module selection vector, and $\boldsymbol{\theta}$ represents the vector of continuous hyperparameters. The resulting model configuration and its point-estimate performance are passed forward as the teacher framework for subsequent distillation.

3.3.7 Phase 6 – Knowledge distillation and compression

To prepare the system for resource-constrained deployments, the verified optimal teacher model (selected from Phase 4 or Phase 5 based on minimum variance and mean performance) undergoes compression. Three alternative compression pipelines are systematically evaluated:

1. **Knowledge Distillation (KD):** A low-capacity student architecture featuring reduced layer depth and hidden units is optimized using the dual-objective loss formulation defined in eq. (2.23).
2. **KD and Post-Training Quantization (PTQ):** A distilled student network undergoes dynamic INT8 quantization targeted at the linear layers to compress runtime footprint without retraining.

3. **KD and Quantized Aware Training (QAT):** A KD-supervised model is fine-tuned with fake-quantization observers targeting both temporal convolutional and linear components, preserving teacher-level behavior under low-precision inference.

All three compressed variants are evaluated on $\mathcal{D}_{\text{test}}$ to map the Pareto efficiency frontier between model scale and accuracy.

3.4 Transfer learning

To evaluate the performance of the EDP model under real-world conditions, the pre-trained simulation models described in section 3.2 were evaluated on and fine-tuned using real machine data. Although the simulation captures the dominant physics of the system, a reality gap exists between simulated and real machine behavior: sensor noise characteristics, unmodeled friction, hydraulic fluid temperature variation, and manufacturing tolerances all introduce distributional differences that the pre-trained model has never observed. At the same time, real logged data from the machine were not available in large quantities, while the simulation data could be generated in larger amounts. Transfer learning bridges this gap by re-using the rich feature representations learned from plentiful simulation data while adapting the model to the target distribution with the limited real data that were available.

3.4.1 Source domain: simulation pre-training

The starting point for every fine-tuning run is a checkpoint produced by the main training pipeline (see section 3.3). A checkpoint serializes the full model state, the experiment configuration, and the `DatasetState` object. The `DatasetState` contains all necessary information regarding the specific model modules and parameters described in section 3.2. Carrying the fitted scaler inside the checkpoint is essential: it ensures that real data can be transformed to exactly the same normalized space that the model was exposed to during simulation training, without refitting any statistics on the small real dataset.

3.4.2 Target domain: real machine data

Real data are collected from the excavator using the Speedgoat real-time target machine (see section 3.1.1.2), which logs all input and output signals at the same sampling rate used in the simulation.
Each log file is processed by the same data loading pipeline used in simulation training, selecting the identical signal set. The pre-trained scaler is then applied to normalize each file.

3.4.2.1 Temporal train/validation split

While the simulation data allowed for a file-level train/validation split, the limited real machine data required a different approach. Because individual log files cap-

tured distinct operating conditions, a file-level split would risk entirely omitting specific regimes from either the training or validation sets.

To ensure the model was both trained and evaluated across the entire operational envelope of the physical excavator, an intra-file temporal split was implemented. Each individual log file was partitioned into an 80/20 train/validation split. This approach guarantees that every recorded operating condition is represented in both phases, accurately reflecting the model's capacity to generalize across diverse operational scenarios.

After splitting, sliding-window sequences of length W are constructed with the same stride logic used during simulation training, including the k-offset construction when $k > 1$.

3.4.3 Backbone–head decomposition

All supported architectures (LSTM, TCN, Hybrid TCN-LSTM) share the same functional decomposition:

Backbone: The temporal encoder layers that transform the input sequence into a compact latent representation: the LSTM or TCN stack, attention pooling, and layer normalization. These layers require large amounts of data to train and are expected to capture generic hydraulic dynamics that transfer well from simulation to real operation.

Prediction head: The shallow layers that map the latent representation to cylinder-wise output predictions: per-cylinder feature extractors and projectors, the optional cylinder feature gate and cross-cylinder attention, and the final output activations. These layers are architecture-specific in name but consistent in role across all models.

3.4.4 Two-phase fine-tuning

The fine-tuning follows a two-phase schedule designed to prevent catastrophic forgetting [31] of the simulation-learned representations while still allowing the backbone to adapt to the real data distribution.

3.4.4.1 Phase 1: Head-only warmup

In Phase 1, the backbone weights are frozen, and only the prediction head is trained. This ensures that the head parameters converge to a stable state before the backbone is exposed to gradient updates from the limited real dataset.

Consequently, Phase 1 is configured to run for fewer epochs and with a smaller learning rate than the main simulation training. This conservative approach prevents the head from generating large, destabilizing gradients early on, thereby protecting the pre-trained feature extractors from premature distortion. Furthermore, a weight decay is applied during this phase to act as a regularizer, preventing the shallow head layers from overfitting to the small sample size of real machine data.

3.4.4.2 Phase 2: Discriminative fine-tuning

In Phase 2, all parameters are unfrozen and trained simultaneously, but with discriminative learning rates [32] applied to the respective sub-modules. The prediction head retains a higher learning rate, while the backbone is updated using a substantially smaller rate.

This distinct step-down ratio between the two learning rates allows the head to adapt freely to the target domain while strictly limiting the magnitude of gradient updates to the backbone. By constraining the backbone's optimization pace, the model preserves the generalized feature representations learned over the much larger simulation dataset while still allowing for subtle domain adaptation. Reflecting this localized tuning approach, Phase 2 is executed for a moderate duration to prevent over-optimization on the real machine data.

The two-phase procedure is summarized in table 3.1.

Table 3.1: Fine-tuning phase summary.

Phase	Epochs	Head LR	Backbone LR	Backbone frozen
1: Head warmup	15	3×10^{-4}	—	Yes
2: Discriminative FT	25	3×10^{-4}	1×10^{-5}	No

3.4.5 Regularization for fine-tuning

Over-fitting is the primary risk when fine-tuning on a small real dataset. Although several regularization mechanisms are shared with simulation pre-training (gradient clipping, weight decay, early stopping, and a learning-rate scheduler), two strategies are applied specifically during fine-tuning and are not active in the pre-training pipeline.

Reduced batch size: Fine-tuning uses a batch size of $B = 32$, considerably smaller than the batch sizes of 128–512 used during simulation pre-training. Because the real dataset contains fewer samples, a large batch would span a substantial fraction of the training set in each step, producing very few gradient updates per epoch. The smaller batch produces more frequent, noisier gradient updates that act as an additional implicit regularizer and allow the model to visit more of the loss landscape curvature before early stopping terminates training.

Light smoothness penalty A smoothness regularization term with weight $\lambda_s = 0.05$ is added to the Huber loss during fine-tuning. This term penalizes the second-order variation of consecutive predictions, discouraging the model from producing rapid oscillations that are artifacts of noisy gradient estimates on the small dataset rather than genuine hydraulic dynamics. In the pre-training Optuna search this weight was included in the search space but its optimized value converged to zero for most architectures, reflecting the fact that the large, clean simulation dataset provides sufficient information without needing to suppress prediction roughness explicitly.

3.5 Evaluation

To assess the quality and practical viability of trained models, an evaluation procedure was established, covering single-step accuracy, multi-step prediction stability, and computational feasibility for real-time deployment. All evaluation metrics are computed on a validation set that the model has not seen during training.

3.5.1 Single-step accuracy

The primary evaluation of a trained model is its ability to predict the next state of the excavator given the current input window. For each output channel, the following complementary metrics are computed:

- **Mean squared error** quantifies the average squared deviation between predicted and true values. It penalizes large errors disproportionately, making it sensitive to outlier predictions, which is desirable since large deviations in cylinder states could represent physically dangerous situations.
- **Mean Absolute Error (MAE)** provides a more robust and interpretable measure of the average prediction error in the same units as the target variable. Unlike MSE, it is less sensitive to occasional large errors and gives a direct sense of the typical prediction offset.
- **Coefficient of determination (R^2)** measures the proportion of variance in the ground truth that is explained by the model's predictions. An R^2 value of 1.0 indicates perfect prediction, while a value of 0.0 indicates that the model performs no better than predicting the mean. This metric contextualizes the error relative to the signal's variability, which is important when comparing channels with different dynamic ranges.
- **95th and 99th Percentile Errors (p95, p99)** characterize the tail behavior of the absolute error distribution. While MAE reflects typical performance, p95 and p99 quantify how large errors become in the worst 5% and 1% of predictions respectively. These metrics are particularly relevant for safety-critical deployment, where the frequency and magnitude of rare but large deviations matter.

These metrics are computed per output channel to identify whether the model struggles with particular state variables. Additionally, residual distributions are plotted per channel as histograms. A well-performing model should exhibit residuals that are tightly centered around zero with approximately Gaussian shape and low p95/p99 values relative to the signal's operating range. Heavy tails or systematic bias in the residual distribution indicate modeling deficiencies for particular operating modes.

3.5.2 Embedded deployment feasibility

To evaluate whether the trained models can be deployed on the target embedded hardware, an analysis of their computational and memory footprints is performed. Using the `thop` library, the number of multiply-accumulate (MAC) operations and parameters is profiled for each architecture. Memory requirements are estimated for both FP32 and INT8 quantized representations of the model weights. Additionally,

peak activation memory during a forward pass is quantified by instrumenting each network layer with forward execution hooks.
These profiled metrics are benchmarked directly against the constraints of the physical vehicle controller hardware: an industrial microcontroller operating at a total clock frequency of 180 MHz, equipped with 2 MB of internal Flash and 512 KB of SRAM. To ensure deterministic execution alongside safety and vehicle control tasks, the deployment framework assumes a strict allocation budget. Specifically, the model is constrained to operate within a 10 ms task cycle, where it is allocated a maximum processing budget of 4 ms. This temporal constraint restricts the model to a maximum computational allowance of 0.72 MHz (equivalent to 7.2×10^5 clock cycles per execution loop). Furthermore, the model memory allocation is budgeted to fit within approximately 500 KB of total available RAM, with a target threshold of roughly 100 KB dedicated specifically to model parameters.

3.6 Simulink Deployment

A key requirement of this work was integrating the trained neural network plant model into the existing MATLAB Simulink environment. This section outlines the export pipeline developed to facilitate this transition.

3.6.1 Export Pipeline: PyTorch to ONNX

Because PyTorch models cannot be loaded directly into MATLAB, the Open Neural Network Exchange (ONNX) format was selected as an interoperability layer. PyTorch's built-in exporter (`torch.onnx.export`) was used to trace the model's forward pass and generate the computation graph.
To ensure the Simulink block could operate directly on raw, unscaled sensor signals, a Python wrapper module (`SimulinkWrapper`) was implemented to embed data transformations directly into the ONNX graph. This eliminates the need to replicate Python preprocessing logic within MATLAB. The wrapper executes four operations in sequence:

1. **Column gather:** Replicates and reorders raw input columns using registered index buffers to match the sequence expected by the preprocessing pipeline.
2. **Input scaling:** Applies the training-phase normalization (z-score and min-max scaling) as element-wise arithmetic using registered parameter buffers.
3. **Model inference:** Passes the scaled input window through the trained network backbone, including any physics-based layers.
4. **Output inverse transform:** Converts predictions back to physical units. For absolute targets, an inverse z-score normalization is applied. For delta targets, the normalized output is scaled by the target standard deviation σ_t to yield a physical-unit delta.

The export script targets specific ONNX opsets to maintain compatibility across MATLAB versions (e.g., R2021b and R2025b), applying compatibility patches where necessary to replace unsupported fused operators.
Alongside the model, the script outputs a metadata file (JSON and MAT formats) containing feature names, window size, target counts, and architecture de-

tails. This metadata enables automated validation and ensures the Simulink-side wiring matches the training configuration.

3.6.2 MATLAB and Simulink integration

The ONNX model was imported into MATLAB using the Deep Learning Toolbox Converter, utilizing either `importONNXFunction` or `importNetworkFromONNX` depending on the MATLAB environment version.
Within Simulink, a persistent first-in, first-out buffer of size $W \times N_{\text{features}}$ manages the sliding window of historical time steps required by the network. At each simulation step, the oldest data point is discarded, the newest feature vector is appended, and the complete window is passed to the imported model function.
If the model is configured in delta target mode, the Simulink block completes the transformation by adding the predicted physical delta to the current measured cylinder velocity, yielding the final absolute prediction.

4
Results

This chapter presents the empirical findings obtained from the development, optimization, and evaluation of plant models. The results are structured chronologically according to the engineering pipeline. First, the data preprocessing outcomes are detailed through data saturation analyzes and the feature selection process, including sensor noise estimations. Next, the performance of the model architectures is tracked across the multi-phase hyperparameter optimization suite, culminating in a detailed evaluation of the optimal Hybrid TCN-LSTM configuration. Finally, the chapter presents the cross-model generalization results when evaluating the baseline and fine-tuned models on physical excavator data, concluding with a feasibility analysis regarding final onboard microcontroller deployment.
Throughout this chapter, the signal names follow the mapping convention outlined in table A.1.

4.1 Feature selection

To optimize model performance and minimize computational overhead, the initial input feature space was pruned to a subset of highly informative variables. This feature reduction process was driven by analyzing the statistical correlation, mutual information, and temporal lag between candidate input signals and the target features. Redundant features exhibiting high collinearity were eliminated, retaining only the primary drivers of system dynamics. The finalized subset of features selected for model training is summarized in table 4.1.

Feature group	**Description**
Cylinder pressure	Piston- and rod-side pressures for each primary actuator.
Pilot pressure	Primary pilot line pressures driving the main control valves.
Conflux pilot pressure	Secondary pilot pressures modulating flow-combinator valves for high-speed operation.
Cylinder velocity	Linear velocity of each actuator, establishing the baseline kinematic state.
Cylinder acceleration	Linear acceleration of each actuator, capturing higher-order dynamic transients.

Table 4.1: Finalized input feature configuration for model training.

Initially, linear and monotonic dependencies within the data were evaluated using Pearson, Spearman, and Kendall rank correlation metrics. All three coeffi-

cients yielded highly consistent trends. The resulting monotonic relationships are illustrated via the Spearman correlation matrix in fig. 4.1. The matrix reveals a pronounced correlation between the primary and conflux pilot pressures and their respective target cylinder velocities, confirming their strong predictive weight.

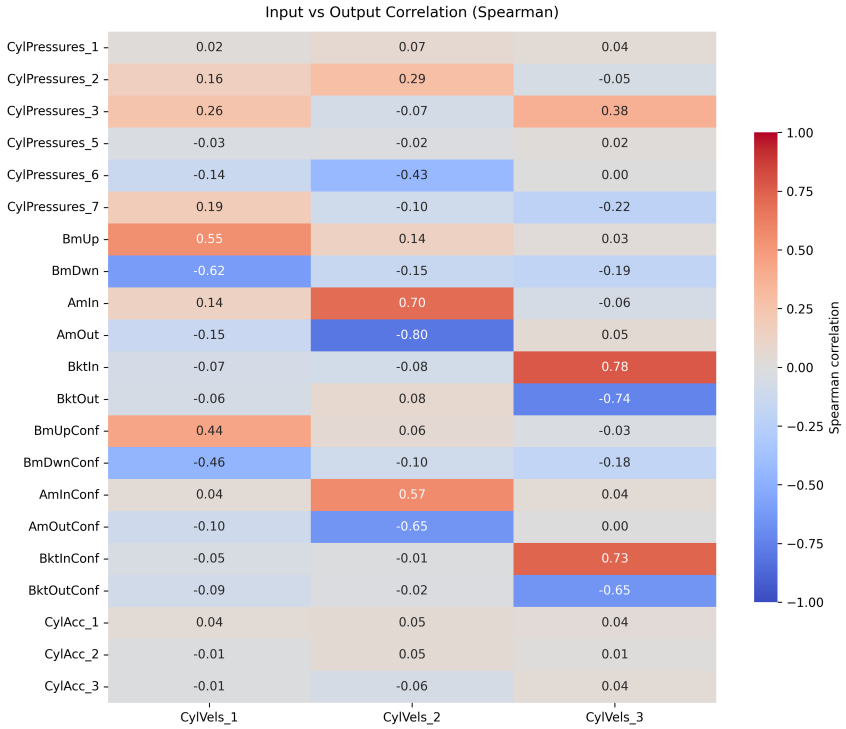


Figure 4.1: Spearman rank correlation matrix between input and output features.

To capture potential non-monotonic and nonlinear dependencies that correlation coefficients might overlook, MI was computed. The resulting MI score matrix is presented in fig. 4.2. The matrix reveals a pronounced informational dependency between cylinder pressures and their corresponding cylinder velocities.
Additionally, cylinder accelerations exhibit a notably higher MI score relative to the previously evaluated linear and rank correlation metrics, where they displayed minimal dependency. While the primary pilot pressures maintain a distinct informational coupling with their respective cylinders, the main actuator cylinder pressures demonstrate the highest overall mutual information in this analysis.
A temporal lag analysis was conducted to determine the time offsets that yield the maximum correlation between the input features and target outputs. The resulting optimal lag ranges are summarized in table 4.2. The results indicate that the primary and conflux pilot pressures exhibit the earliest optimal lag times relative to the system outputs. This temporal precedence confirms that pilot pressure signals provide early indicators of upcoming system dynamics, making them highly informative for predicting future cylinder velocities.
Furthermore, these lag profiles established a practical guideline for defining the input window size of the neural networks. Because the maximum relevant historical information is contained within approximately 20 samples, an extended window size of 100 samples was determined to be redundant. Consequently, a more compact

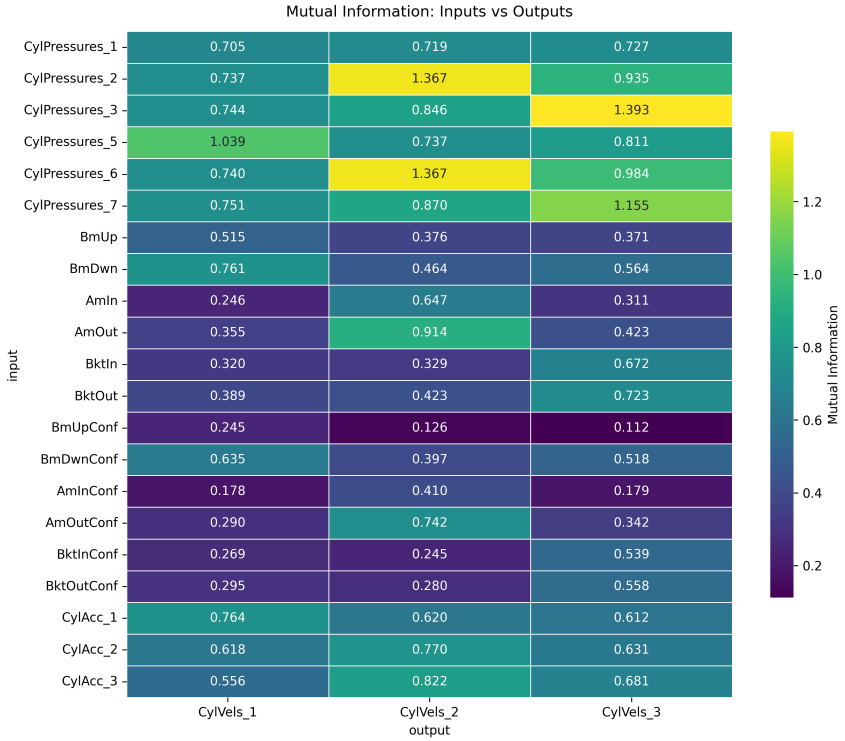


Figure 4.2: Mutual Information matrix quantifying nonlinear dependencies between input and output features.

window size of 50 samples was selected, reducing computational overhead without omitting critical temporal dependencies.

Input Feature Group	Optimal Lag Range (Samples)
Cylinder pressure	-8 to -4
Pilot pressure	-18 to -12
Conflux pilot pressure	-20 to -13
Cylinder acceleration	-12 to -5

Table 4.2: Optimal lag time ranges for maximizing correlation between candidate input features and kinematic outputs.

4.2 Optimization suite results

The following sections outline the results and observations obtained by running the proposed phase based search.

4.2.1 Phase 0: Baseline and variance decomposition

To establish the statistical foundation required for subsequent HPO, baseline evaluations isolated the dataset split variance (σ_{split}) from initialization seed variance (σ_{seed}).

As visualized in fig. 4.3 and detailed in table 4.3, σ_{split} dominates over σ_{seed} across all architectures, frequently by an order of magnitude. This sensitivity to sequence composition indicates that single-split evaluations introduce substantial evaluation bias, justifying the multi-split validation protocol described in section 3.3.4.

While the longer forecasting horizon ($k = 10$) scales the absolute mean absolute error ($\overline{\text{MAE}}$) by approximately one order of magnitude due to cumulative physical complexity, the relative variance dynamics remain highly consistent across both offsets.

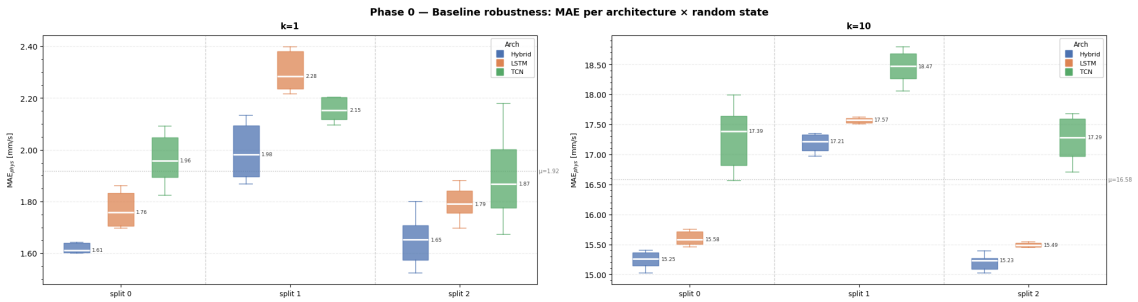


Figure 4.3: Performance variance across architectures, data splits, and initialization seeds.

Table 4.3: Phase 0 — Variance decomposition comparison for offset $k = 1$ and $k = 10$, MAE (mm/s).

		Offset $k = 1$			**Offset** $k = 10$		
Arch	**FG**	$\overline{\textbf{MAE}}$	$\sigma_{\textbf{split}}$	$\sigma_{\textbf{seed}}$	$\overline{\textbf{MAE}}$	$\sigma_{\textbf{split}}$	$\sigma_{\textbf{seed}}$
Hybrid	FG1	1.791	0.253	-	15.968	1.163	-
	FG2	**1.694**	**0.160**	0.037	**15.766**	1.103	0.127
LSTM	FG1	2.039	0.296	-	16.244	1.172	-
	FG2	1.909	0.287	**0.023**	16.141	1.170	**0.098**
TCN	FG1	1.984	0.103	-	17.899	0.610	-
	FG2	1.904	0.241	0.159	17.369	**0.837**	0.301

Beyond variance decomposition, these baseline runs reveal early architectural and feature preferences. The Hybrid architecture utilizing Feature Group 2 (FG2) consistently achieves the lowest overall error for both short-term ($k = 1$, $\overline{\text{MAE}}_{\text{phys}} = 1.694\,\text{mm/s}$) and long-term ($k = 10$, $\overline{\text{MAE}}_{\text{phys}} = 15.766\,\text{mm/s}$) horizons. Moreover, FG2 systematically outperforms FG1 across all tested models (Hybrid, TCN, and LSTM). This uniform improvement confirms that integrating differential pressure ($\boldsymbol{\Delta}_{\text{pressure}}$), as defined in section 3.1.2.1, provides a predictive signal that generalizes well across distinct model mechanics.

4.2.2 Phase 1: Hyperparameter optimization and architecture screening

Phase 1 executes joint HPO spanning: learning rate, weight decay, dropout, loss function, and backbone size, using Optuna with TPE sampling across 28 trials per architecture. Each trial is evaluated via $S_{\text{composite}}$ defined in eq. (3.41), with performance metrics summarized in table 4.4.

Table 4.4: Phase 1 — Composite score and Mean MAE (mm/s) per architecture × feature group.

(a) Offset $k = 1$

Arch	FG	$\overline{S}_{\textbf{composite}} \pm \sigma$	Min	$\overline{\textbf{MAE}} \pm \sigma$
Hybrid	FG1	0.0126 ± 0.0013	0.0108	-
	FG2	$\mathbf{0.0092 \pm 0.0014}$	**0.0079**	$\mathbf{1.4472 \pm 0.2648}$
LSTM	FG1	0.0132 ± 0.0011	0.0120	-
	FG2	$\mathbf{0.0108 \pm 0.0022}$	**0.0086**	-
TCN	FG1	0.0134 ± 0.0017	0.0118	-
	FG2	$\mathbf{0.0107 \pm 0.0008}$	**0.0096**	-

(b) Offset $k = 10$

Arch	FG	$\overline{S}_{\textbf{composite}} \pm \sigma$	Min	$\overline{\textbf{MAE}} \pm \sigma$
Hybrid	FG1	0.0913 ± 0.0015	0.0895	-
	FG2	$\mathbf{0.0896 \pm 0.0007}$	**0.0887**	$\mathbf{12.9480 \pm 0.1382}$
LSTM	FG1	0.0912 ± 0.0018	0.0893	-
	FG2	$\mathbf{0.0905 \pm 0.0017}$	**0.0888**	-
TCN	FG1	0.0920 ± 0.0032	0.0892	-
	FG2	$\mathbf{0.0899 \pm 0.0015}$	**0.0880**	-

Three key insights emerged to guide subsequent phases:

Feature group superiority: Reflecting Phase 0 trends, FG2 consistently outperforms FG1 across all architectures and horizons. This uniform reduction in composite score confirms that the explicit pressure time-derivatives ($\boldsymbol{\Delta}_{\text{pressure}}$) from eq. (3.3) provide robust predictive utility irrespective of the backbone. Consequently, FG2 is frozen as the default feature configuration for all remaining phases.

Architectural leadership: The Hybrid TCN-LSTM architecture achieves the lowest composite score in both forecasting horizons. For the short-term horizon ($k = 1$), the Hybrid establishes a distinct margin of $\approx$ 0.0015–0.0016 composite units over the standalone LSTM and TCN. However, for the long-term horizon ($k = 10$), performance across the three backbones converges to within 0.0009 units,

indicating that the structural inductive bias of the Hybrid yields diminishing returns over extended horizons.

Pareto-optimal model selection: While the HPO search space evaluates four backbone widths (64, 128, 256, and 512 units), scaling backbone width introduces trade-offs regarding parameter counts and training latency. As illustrated in fig. 4.4, the delta in composite score across widths is marginal, prompting a heuristic trade-off analysis. For $k = 1$, a 128-unit width is selected; expanding to 512 units yields a minor $\approx 13\%$ reduction in MAE at the cost of a $15\times$ parameter explosion. Conversely, the elevated complexity of the $k = 10$ horizon justifies a 256-unit backbone, capturing critical performance gains before substantial efficiency degradation occurs.

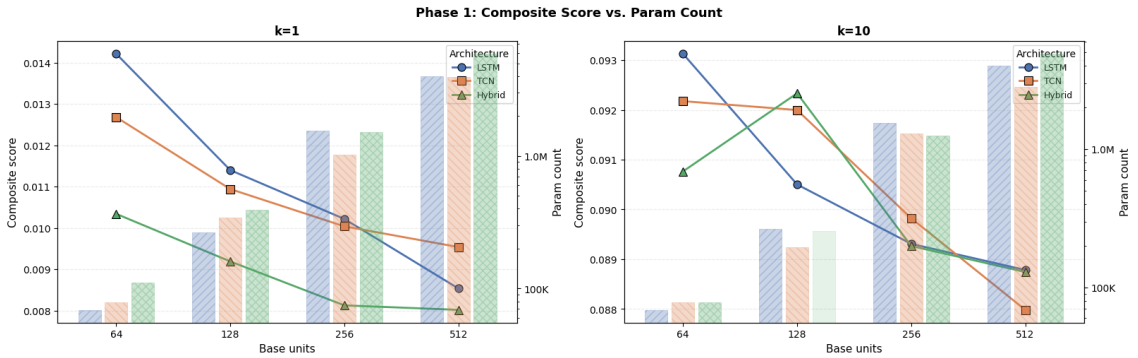


Figure 4.4: Model performance (left y-axis, composite score) versus model complexity (right y-axis, log-scaled parameter count) across varying backbone widths to evaluate Pareto-optimal configurations.

4.2.3 Phase 2 & 3: Screening and confirmation

Phase 2 evaluates architectural modules against a baseline median using a restricted hyperparameter budget designed for the elastic tuning window, described in section 3.3.3. The screening threshold proved highly permissive: only *Cross-Cylinder Attention* failed to outperform the baseline at $k = 1$, while all fourteen candidates advanced at $k = 10$ (table 4.5).

Phase 3 subjects these candidates to cross-validated confirmation (section 3.3.4), revealing a performance divergence between forecasting horizons. For short-term horizons ($k = 1$), only *Multiscale TCN* maintains statistical significance ($d_z = -0.79, p < 0.01$). On the other hand, long-term horizons ($k = 10$) yield five highly significant structural improvements ($p < 0.001$): *STanh Activation* ($d_z = -6.30$), *LayerNorm LSTM* ($d_z = -2.00$), *Temporal Diff* ($d_z = -1.50$), *Squeeze-Excitation* ($d_z = -1.47$), and *GLU Head* ($d_z = -1.21$).

The distinction between Phase 2 pass rates (93–100%) and Phase 3 confirmation rates (7% for $k = 1$; 36% for $k = 10$) highlights substantial single-trial selection noise in the screening phase. Most notably, *Physics Residual* easily cleared Phase 2 but severely degraded accuracy under repeated cross-validation ($d_z \approx +18.5$ and $+22.6$). These dynamics motivate the dual-phase validation pipeline as an essential filter against false positives introduced by single-optimization variance.

Table 4.5: Phase 2 & 3 — Module screening metrics and confirmatory significance ($k = 1 : n_{\text{splits}} = 10 \mid k = 10 : n_{\text{splits}} = 7$). Performance improvements are indicated by $\Delta p_2 > 0$(composite units) and Cohen's $d_z < 0$.

	Offset $k = 1$			**Offset** $k = 10$		
Module Component	Δp_2	p-value	Cohen's d_z	Δp_2	p-value	Cohen's d_z
Structural Gating						
Cyl Gate Diagonal	0.374	0.7442	0.208	0.503	0.3372	-0.159
Temporal Diff	0.347	0.5914	0.073	0.444	< 0.001	**-1.496*****
Sequence Architecture						
Cross Cyl Attn	-0.364	—	—	0.391	0.9985	1.121
GRU Backbone	0.884	0.9846	0.683	1.045	> 0.999	2.285
LayerNorm LSTM	0.347	> 0.999	1.869	0.873	< 0.001	**-1.998*****
Multiscale TCN	1.131	0.0066	**-0.785****	0.967	0.9999	1.387
Zoneout	0.009	> 0.999	4.360	0.064	0.8285	0.358
Channel Attention						
Attention Bridge	1.049	0.1101	-0.388	0.746	> 0.999	1.492
Global Avg Pool	0.924	0.5116	0.009	0.925	0.9863	0.834
Squeeze-Excitation	0.805	> 0.999	1.959	0.303	0.0001	**-1.468*****
Physics Constraint						
Physics Residual	0.220	> 0.999	18.548	0.446	> 0.999	22.638
Output Head						
Deeper Head	1.717	0.6110	0.089	0.215	0.9824	0.796
GLU Head	1.227	> 0.999	2.212	0.252	0.0007	**-1.213*****
STanh Activation	1.342	> 0.999	1.852	29.148	< 0.001	**-6.300*****

$^{**}p < 0.01$, $^{***}p < 0.001$ (one-tailed paired t-test). Bold entries indicate confirmed positive architectural components.

4.2.4 Phases 4 & 5: Joint optimization and unconstrained discovery

To evaluate the efficacy of the individual-merit filtering used in Phase 3, the pipeline divides into two distinct architectural synthesis strategies. Phase 4 executes a constrained joint optimization, re-opening only the verified components from Phase 3 alongside deferred hyperparameters to test for mutual synergy. Conversely, Phase 5 bypasses the Phase 3 filter entirely, exposing the complete unconstrained 2^N module space to the TPE. This tests if individually discarded components form highly synergistic pairs when toggled jointly, utilizing a warmup trial initialized from the Phase 4 winner to guide the search toward established high-performance regions.

Offset-dependent component selection: As shown in table 4.6, component selection frequencies reveal structural preferences heavily dependent on the prediction horizon k. Multiscale TCN (M_2) serves as a foundational component for short-horizon tasks ($k = 1$), retaining a dominant selection affinity across both Phase 4

(70.6%) and Phase 5 (79.3%). Conversely, Temporal Differencing (M_3) and Stanh Activation govern long-horizon tasks ($k = 10$). This division aligns with each module's intrinsic inductive bias: M_2 captures high-frequency local dynamics essential for next-step accuracy, whereas M_3 enforces low-frequency physical anchoring to mitigate cumulative drift over extended sequences. Notably, auxiliary loss functions (Correlation and Focal Loss) achieved moderate selection frequencies under the constrained space of Phase 4 but dropped to minimal or zero affinity in Phase 5. This suggests these loss modules acted as performance surrogates to compensate for an artificially restricted architectural space rather than providing fundamental optimization utility.

Table 4.6: Architectural component selection frequency among all candidate models explored in Phases 4–5. Values denote the percentage of runs containing the corresponding component at each prediction horizon. Mean affinity represents the average selection frequency across all four evaluation groups.

	Phase 4		**Phase 5**		
Component	$k = 1$	$k = 10$	$k = 1$	$k = 10$	**Mean Affinity**
Temporal Diff	0.0%	76.5%	68.3%	83.0%	**56.9%**
Stanh Activation	0.0%	76.5%	29.3%	73.0%	44.7%
SWA	70.6%	23.5%	40.2%	41.1%	43.9%
Multiscale TCN	70.6%	0.0%	79.3%	17.0%	41.7%
Attention Bridge	23.5%	0.0%	73.2%	42.6%	34.8%
Correlation Loss	23.5%	23.5%	20.7%	17.0%	21.2%
Focal Loss	29.4%	23.5%	14.6%	15.6%	20.8%
LayerNorm LSTM	0.0%	23.5%	51.2%	7.8%	20.6%
Squeeze-Excitation	0.0%	23.5%	20.7%	18.4%	15.7%
Deeper Head	0.0%	0.0%	35.4%	14.9%	12.6%
Zoneout	0.0%	0.0%	14.6%	35.5%	12.5%

Unconstrained synergy vs. incremental filtering: Evaluating the final configurations on the holdout set ($\mathcal{D}_{\text{holdout}}$) reveals the clear advantage of unconstrained search profiles (table 4.8). For the long-term horizon ($k = 10$), both phases independently converged on identical parameter counts and highly similar architectures leveraging M_3, confirming that the Phase 3 significance filter successfully isolates dominant long-horizon modules. However, the limitation of incremental filtering is exposed at $k = 1$. In this domain, Phase 5 discovered a highly synergistic four-module topology (M_1, M_2, M_3, M_6) that was structurally inaccessible to Phase 4. This complex composition drove a 52.3% reduction in the composite score ($S_{\text{composite}}$ falling from 0.001145 to 0.000546), proving that multi-component interactions can vastly outperform individually optimal selections.

Table 4.7: Module identifiers

ID	Module	ID	Module
M_1	Attention bridge	M_4	Squeeze excitation
M_2	Multiscale TCN	M_5	Zoneout
M_3	Temporal differencing	M_6	LayerNorm

Table 4.8: Consolidated architecture, size, and performance metrics for Holdout models. Percentages in parentheses indicate the relative change (parameter growth vs. error reduction) compared to the baseline mean values.

(a) Offset $k = 1$

Metric	$\textbf{Holdout}_{\textbf{P4}}$	$\textbf{Holdout}_{\textbf{P5}}$
Active Modules	M_2	M_1, M_2, M_3, M_6
Parameters	677,572 (+70.7%)	703,527 (+77.3%)
FP32 (MB)	2.7103	2.8141
$S_{\text{composite}}$	0.001145 (−87.6%)	**0.000546 (−94.1%)**
MAE (mm/s)	1.117 (−22.8%)	**1.083 (−25.2%)**

(b) Offset $k = 10$

Metric	$\textbf{Holdout}_{\textbf{P4}}$	$\textbf{Holdout}_{\textbf{P5}}$
Active Modules	M_3	M_3, M_5
Parameters	1,197,319 (+0%)	1,197,319 (+0%)
FP32 (MB)	4.7893	4.7893
$S_{\text{composite}}$	0.0593 (−33.8%)	**0.0535 (−40.3%)**
MAE (mm/s)	10.33 (−20.2%)	**10.28 (−20.6%)**

Point accuracy vs. trajectory regularization: The holdout results reveal an engineering trade-off between absolute point-wise error and physical consistency, particularly at $k = 10$. Individual trials within Phase 5 (e.g., Trial 71) achieved the lowest raw physical error ($\text{MAE}_{\text{phys}} = 10.10\,\text{mm/s}$) by employing M_3 in isolation, outperforming the final holdout model by a marginal 0.18 mm/s. However, the cross-validated Holdout model combines temporal differencing with zoneout (M_3, M_5), securing a significantly superior composite score (0.0535 vs. 0.0602). This demonstrates that the inclusion of M_5 introduces a stochastic regularization effect that compromises only a sub-millimeter fraction of raw point-wise accuracy. In turn, it promotes structurally smoother, physically consistent trajectory derivatives over extended forecasting windows.

Crucially, the raw physical error magnitude scales near-linearly with the temporal horizon, rising from $\approx 1\,\text{mm/s}$ at $k = 1$ to $\approx 10\,\text{mm/s}$ at $k = 10$ across the holdout evaluations. This uniform scaling highlights a predictable baseline of temporal error compounding across both optimization strategies.

Structural complexity and downstream memory footprints: An analysis of model footprints across phases reveals that performance gains do not scale quadratically with physical parameter weights or memory overheads (table 4.8). At $k = 1$, the four-module synergy discovered in Phase 5 increases the parameter baseline by only 3.8% (677, 572 to 703, 527 parameters) while cutting the overall composite score in half. This decoupling demonstrates that the unconstrained search space promotes a superior regularized topology, capturing trajectory and derivative stability with a completely negligible difference in raw point-wise MAE_{phys}.

4.2.5 Phase 6: Knowledge distillation and compression

As hardware limitations imposed by edge ECUs are strict the evaluation of KD, LPQAT and QAT are highly interesting to get an indication of wether hardware deployment is feasible. Utilizing the holdout winners from the two respective target offsets k=1 and k=10, presented in table 4.8, as teacher models to evaluate the tradeoff between performance and complexity. The complete results are presented in table 4.9, while the trade-offs between parameter scale, MAE and $S_{\text{composite}}$ are visualized below in fig. 4.5 and 4.6

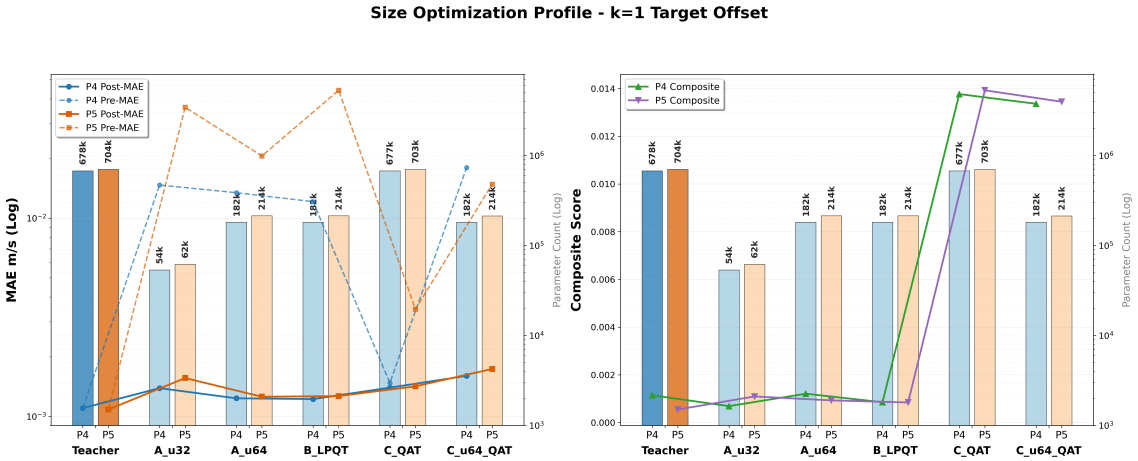


Figure 4.5: Size optimization profile for the $k = 1$ target offset formulation, illustrating the trade-off between model parameter count (bars, log scale), pre- and post-compression MAE (left subplot, log scale), and the resulting multi-objective $S_{\text{composite}}$ (right subplot).

Figure 4.6: Size optimization profile for the $k = 10$ target offset formulation, detailing the scaling behavior of MAE (left, log scale) alongside $S_{\text{composite}}$ highlighting Pareto-optimal deployment candidates (right).

Evaluation of target offset k=1 For the short-term horizon k=1 the compression suite yields promising results. As the errors for both Teacher P4 (MAE = 0.001117) and P5 (MAE = 0.001083) are exceptionally low, the relative performance decrease incurred via distillation and quantization translate to negligible absolute deviations in mm/s.

As illustrated in fig. 4.5(left), the student variations learns to operate within the same region as the teacher, clearly visible by the pre-and post MAE illustrations. The A_u64 and B_LPTQ variants distilled from the P4 teacher emerge as competitive options, enforcing a $\sim 73.1\%$ reduction in parameter density and a $\sim 74.5\%$ reduction in storage size at a minimal relative error penalty of $+10.47\%$.

Architecturally, the P4 topology demonstrates superior distillation stability at this horizon, restricting MAE inflation more effectively than the P5 counterparts ($+10.47\%$ vs. $+15.97\%$). Conversely, the P5 students exhibits superior computational efficiency where the P5 A_u32 variant minimizes computational overhead to a group-bound minimum of 2.093×10^6 MACs.

A structural insight is highlighted when analyzing computational scaling across horizons: the $k = 1$ student networks are significantly more computationally intensive than their $k = 10$ equivalents, demanding approximately $2.6\times$ more MACs for the P4 A_u32 model and nearly $4\times$ for the P5 variant. This indicate that tighter temporal constraints demand structurally or computationally denser modules to map the underlying system dynamics accurately. However, the flat profiles observed in the composite scores (Figure 4.5, right) indicate that if an error increase of $\sim 40\%$ is acceptable, these representations can be trimmed down further to match the $k = 10$ compute signatures.

Table 4.9: Merged P6 family-winner results. MACs are reported in millions (M). Relative MAE change (Δ%) is calculated against each respective Teacher baseline. For student variants: **bold** indicates the best variant for that specific teacher; **bold + underline** indicates the absolute best variant across the entire k group.

(a) Results for $k = 1$

Variant	MAE	Δ MAE (%)	Params	Size (MB)	Prec.	MACs (M)
Teacher P4	0.001117	Baseline	677572	2.7103	FP32	31.281
A_u32	0.001388	+24.26	**53604**	**0.2144**	FP32	**2.486**
A_u64	**0.001234**	**+10.47**	181956	0.7278	FP32	8.409
B_LPTQ	**0.001234**	**+10.47**	181956	0.6905	INT8	8.409
C_QAT	0.001287	+15.22	676676	0.9793	INT8	31.281
C_u64_QAT	0.001529	+36.88	181572	0.2834	INT8	8.409
Teacher P5	0.001083	Baseline	703527	2.8141	FP32	26.488
A_u32	0.001562	+44.23	**61996**	0.2480	FP32	**2.093**
A_u64	**0.001256**	**+15.97**	213972	0.8559	FP32	6.834
B_LPTQ	**0.001256**	**+15.97**	213972	0.6290	INT8	6.834
C_QAT	0.001293	+19.39	702628	0.7178	INT8	26.488
C_u64_QAT	0.001652	+52.54	213588	**0.2220**	INT8	6.834

(b) Results for $k = 10$

Variant	MAE	Δ MAE (%)	Params	Size (MB)	Prec.	MACs (M)
Teacher P4	0.010333	Baseline	1197319	4.7893	FP32	45.198
A_u32	0.015068	+45.82	**24228**	**0.0969**	FP32	**0.958**
A_u64	0.013891	+34.43	84292	0.3372	FP32	3.257
B_LPTQ	0.013891	+34.43	84292	0.2814	INT8	3.257
C_QAT	**0.013161**	**+27.37**	1196804	2.7793	INT8	45.198
C_u64_QAT	0.015792	+52.83	84164	0.1849	INT8	3.257
Teacher P5	0.010275	Baseline	1197319	4.7893	FP32	18.984
A_u32	0.015471	+50.57	**24228**	**0.0969**	FP32	**0.549**
A_u64	0.014072	+36.95	84292	0.3372	FP32	1.619
B_LPTQ	0.014072	+36.95	84292	0.2814	INT8	1.619
C_QAT	**0.013147**	**+27.95**	1196804	2.7793	INT8	18.984
C_u64_QAT	0.016125	+56.93	84164	0.1849	INT8	1.619

Evaluation of target offset k=10 At the long term offset k=10, the interplay between network capacity, accuracy, and execution cost becomes nuanced, creating distinct optimization trajectories based on deployment criteria:

1. **Lean edge deployment:** For scenarios where storage footprint is the primary constraint, the A_u32 variant represents the absolute minimum bounding case. It shears away $\sim 98\%$ of the baseline parameter matrix, yielding an lightweight footprint of 0.0969 MB, though this pruning incurs a degradation in accuracy (+45.82% MAE for P4, +50.57% MAE for P5).
2. **Accuracy-preserving deployment:** When predictive capacity cannot be

compromised, the C_QAT paradigm proves optimal. By retaining the baseline structural capacity while quantizing to INT8, it limits MAE inflation to +27.37% (P4) and +27.95% (P5), as reflected by the dips in the post-MAE curve in Figure 4.6 (left).

The most prominent architectural finding at $k = 10$ is the structural efficiency introduced by the Phase 5 baseline. Teacher P5 trims baseline MAC requirements by over 58% relative to P4 (18.984 M vs. 45.198 M). This computational reduction propagates directly to its compressed descendants, allowing the P5 A_u32 variant to achieve a minimal computational load of 0.549×10^6 MACs.

Compatibility and hardware verification A part of this study was evaluating compliance with edge hardware constrained by memory and computational budget. While the non-pruned C_QAT configurations (2.7793 MB) exceed memory threshold at $k = 10$, intermediate compression configurations successfully reconcile this constraint. Specifically, the B_LPTQ (0.2814 MB) and C_u64_QAT (0.1849 MB) variants satisfy several hardware-induced limitations while maintaining balanced error characteristics. Simultaneously, the computational throughput constraints are considered, as the drastic reduction in MACs across both horizons—culminating in the ultra-low 0.549×10^6 MAC footprint for $k = 10$, significantly eases the runtime scheduling overhead on the edge hardware.

At the $k = 1$ horizon, compliance is even less constrained. High-fidelity models such as B_LPTQ (0.6290-0.6905 MB) and the accuracy-focused C_QAT model derived from Teacher P5 (0.7178 MB) are close to the hardware allocation.

Crucially, the performance advantages realized by the Phase 5 architecture do not impose downstream deployment overhead. The architectural centerpiece of Phase 5—Zoneout regularization—functions strictly as a train-time constraint. It guides gradient convergence and optimizes the underlying structural representation during training, but adds zero inference variables or computational latency during edge execution. Consequently, the optimal compressed variants developed in Phase 6 successfully preserve the modeling advantages of Phase 5 within the target hardware constraints.

4.3 Evaluation of the optimal hybrid configuration

Based on the comparative analysis, the Hybrid architecture demonstrated the highest potential due to its balance of superior prediction accuracy, compact model size, and low computational overhead. This section provides a detailed evaluation of the top-performing Hybrid model configurations across two distinct time horizons (offsets k=1 and k=10). Detailed specifications of the model configuration and architecture can be found in section B.1. The model's performance is analyzed using cylinder-specific prediction versus ground truth trajectories, error distribution profiles, and model complexity metrics.

The prediction profiles for the free-air digging duty cycle are illustrated in fig. 4.7a, showing cylinder velocity over elapsed time. The trajectory begins with independent

joint actuation, followed by simultaneous multi-axis movement. The Hybrid model demonstrates robust tracking accuracy under both operating conditions, closely aligning with the ground truth during isolated and concurrent cylinder maneuvers. Similarly, fig. 4.7b displays the corresponding tracking performance during an auto-grading operation. This duty cycle induces significant velocity oscillations, particularly within the bucket cylinder. Despite these dynamic disturbances, the model maintains high predictive fidelity and accurately tracks the oscillatory behavior.

4.4 Evaluation on physical machine data

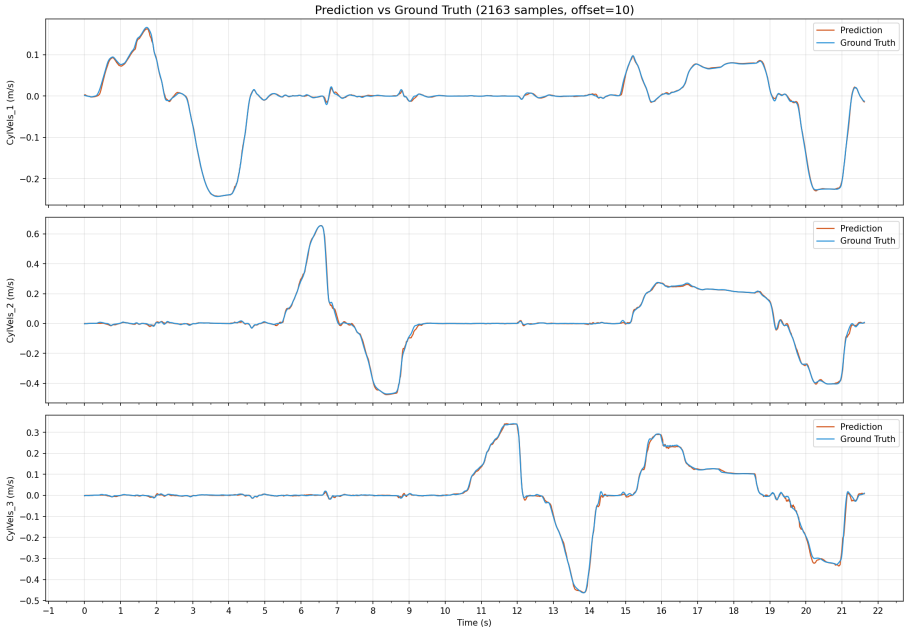
The models were evaluated on physical machine data logged via the Speedgoat target computer (section 3.1.1.2). The objective was twofold: to assess model performance on real-world data containing complex soil-tool interactions, and to evaluate generalization capabilities across different excavator platforms. This cross-model evaluation was necessary because the simulation environment modeled a Volvo EC210F, whereas the real-world data was collected from an ECR145F. These evaluations directly address RQ3. The following subsections present the performance of the standard Hybrid model (section 4.4.1) followed by an evaluation of the fine-tuned variant using transfer learning (section 4.4.2).

4.4.1 Standard Hybrid TCN-LSTM (offset=10)

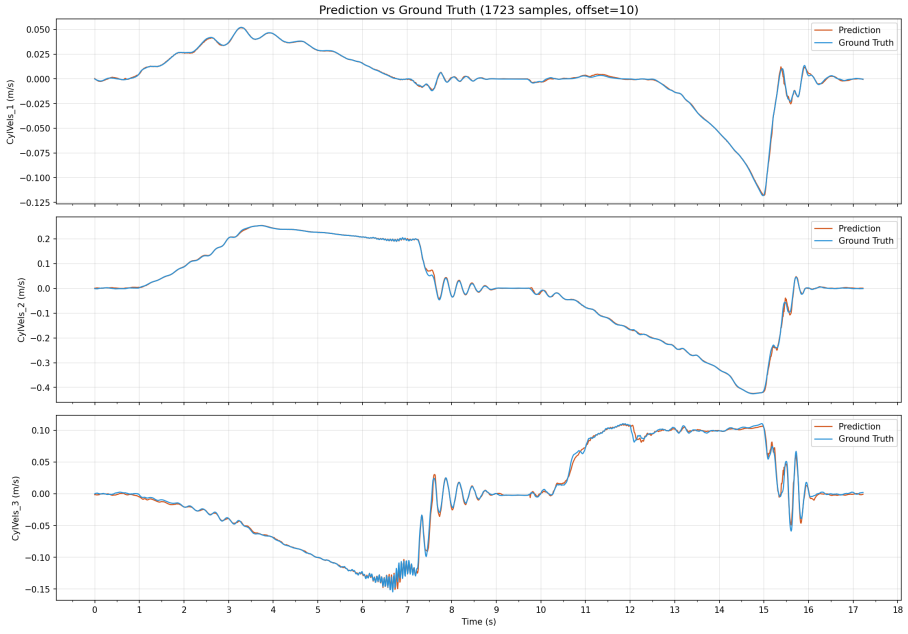
The architecture selected for this evaluation was the top-performing Hybrid TCN-LSTM configuration previously analyzed in section 4.3 (detailed specifications are provided in section B.1). In this section, this model is referred to as the "standard Hybrid" denoting that it was trained exclusively on simulation data. Running inference with this model on real machine data introduces both a sim-to-real gap and a platform mismatch. The results indicate that the standard Hybrid model struggles to accurately predict the dynamics of the physical excavator, yielding significantly lower performance compared to its evaluation on pure simulation data. This behavior is illustrated in the prediction versus ground truth comparison (fig. 4.8) and the corresponding error distribution plot (fig. 4.9). The error distribution reveals a high frequency of both minor and significant tracking errors. Furthermore, the distribution curve is noticeably skewed, indicating that the model suffers from a persistent steady-state error.

4.4.2 Fine-tuned hybrid model

The subsequent results evaluate the Hybrid model after it was fine-tuned on physical machine data using transfer learning (section 3.4). These results demonstrate a substantial improvement in tracking real-world excavator dynamics. This indicates that the Hybrid architecture can successfully generalize across different excavator models, confirming that viable sim-to-real deployment is achievable when leveraging a limited amount of real-world data for fine-tuning. As shown in the prediction versus ground truth plot (fig. 4.10), the predictive capability of the model improves substantially after fine-tuning. This is further supported by the error distribution plot



(a) Free-air digging operation.



(b) Auto-grading operation.

Figure 4.7: Prediction versus ground truth trajectories for the Hybrid TCN-LSTM model (offset=10) under simulated conditions.

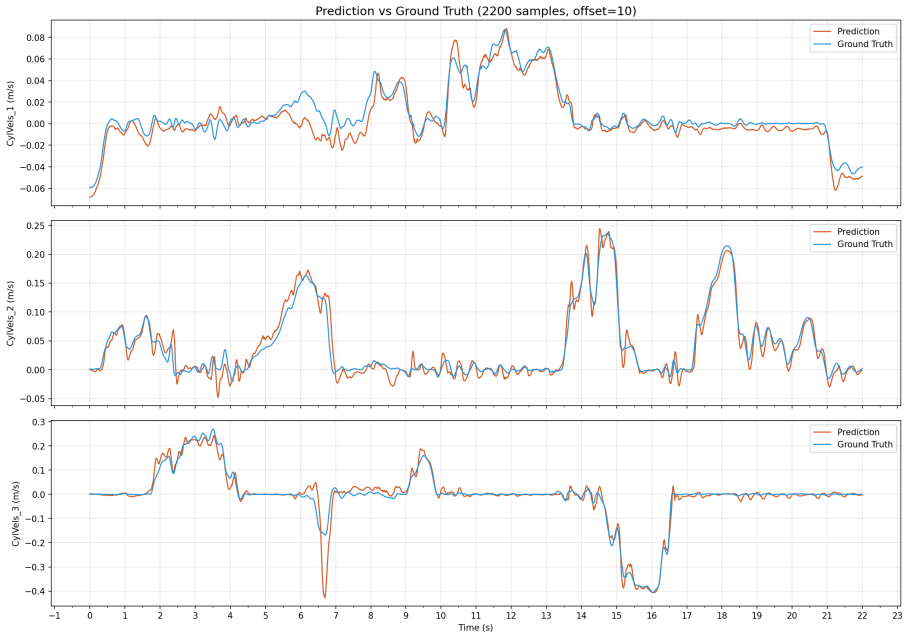


Figure 4.8: Inference on real world data, standard Hybrid (offset=10)

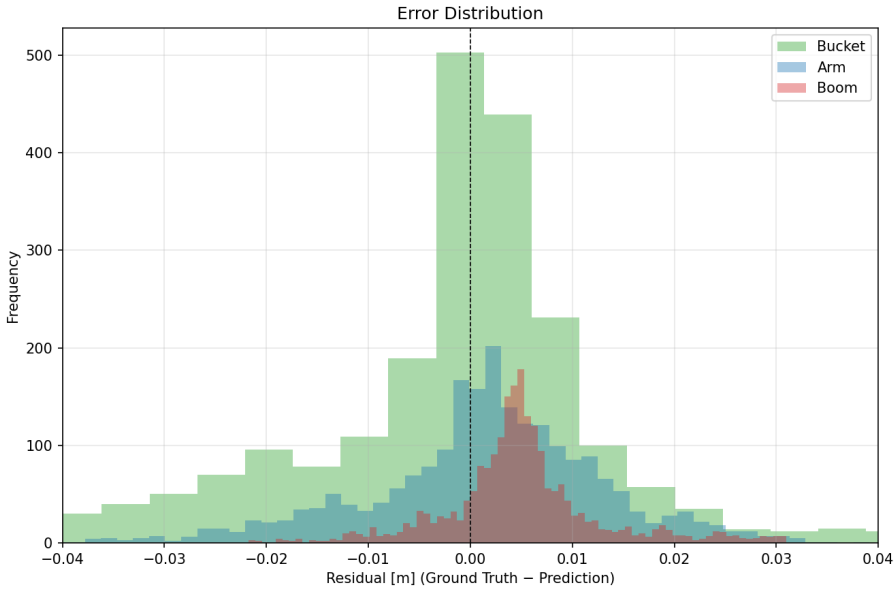


Figure 4.9: Error distribution, standard Hybrid (offset=10)

(fig. 4.11), where the residuals are tightly concentrated around zero with negligible skew, confirming that the systematic tracking bias has been largely mitigated.

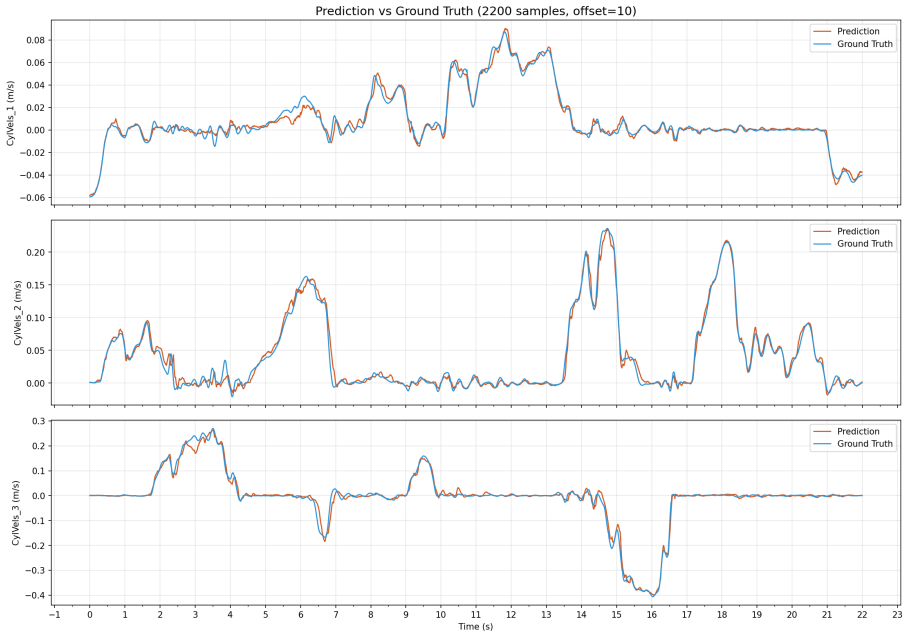


Figure 4.10: Inference on real world data, fine-tuned Hybrid (offset=10)

4.4.3 Model variant comparison

A direct performance comparison between the Standard Hybrid model and the Fine-Tuned Hybrid variant is compiled in table 4.10. Both architectures were evaluated using an identical, previously unseen dataset captured directly from physical machine operations to ensure an unbiased validation of their generalization capabilities. The empirical results reveal that the Fine-Tuned Hybrid model achieves a sweeping and significant performance advantage across all evaluated metrics and cylinder channels. By utilizing a constrained subset of physical operational data for localized domain adaptation, tracking errors are significantly mitigated. Specifically, the MAE for the Boom, Arm, and Bucket is restricted to just 1.7 mm/s, 3.1 mm/s, and 4.0 mm/s, respectively. This represents an error reduction of over 50% across the board compared to the baseline Standard Hybrid configuration.
The impact of fine-tuning is most pronounced in the tracking stability of the high-inertia components. The Boom channel's coefficient of determination (R^2) sees a increases significantly, climbing from 0.7862 to 0.9737. Furthermore, the drastic compression of the 95th and 99th percentile errors (P95 and P99) demonstrates that fine-tuning effectively eliminates severe transient prediction spikes during unexpected operational maneuvers. These findings indicate that while base physics-informed models establish a reasonable foundation, a targeted sim-to-real calibration phase is mandatory to achieve the millimeter- and centimeter-scale tracking precision required for real-world automated excavator deployment.

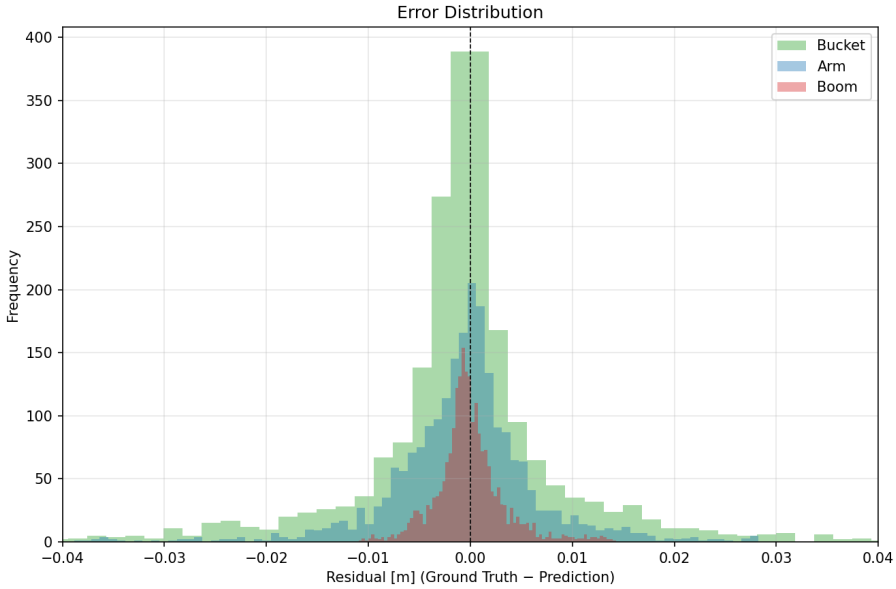


Figure 4.11: Error distribution, fine-tuned Hybrid (offset=10)

Table 4.10: Performance metric comparison between the fine-tuned hybrid and standard hybrid (offset=10).

Model Variant	**Cylinder**	**MSE** (m^2/s^2)	**MAE** (m/s)	R^2	**P95** (m/s)	**P99** (m/s)
Standard	Boom	0.000078	0.0066	0.7862	0.0162	0.0281
	Arm	0.000167	0.0083	0.9476	0.0270	0.0519
	Bucket	0.000236	0.0091	0.9097	0.0321	0.0517
Fine-Tuned	Boom	**0.000010**	**0.0017**	**0.9737**	**0.0062**	**0.0130**
	Arm	**0.000030**	**0.0031**	**0.9906**	**0.0108**	**0.0222**
	Bucket	**0.000059**	**0.0040**	**0.9774**	**0.0154**	**0.0312**

5
Discussion

This chapter evaluates the empirical findings of this study, analyzing the engineering trade-offs, performance constraints, and practical deployment implications of the developed data-driven frameworks. The discussion details the outcomes of the feature selection and training phases, followed by an analysis of the EDP regarding its performance and capabilities. Finally, the chapter addresses the operational constraints encountered during virtual Simulink integration and physical testing on the excavator.

5.1 Feature selection

Determining the optimal input feature configuration required balancing model complexity against predictive capability. To accurately predict future excavator velocities over an extended look-ahead horizon, the input space had to contain not only the instantaneous kinematic state but also predictive indicators of upcoming dynamic changes. Initially, evaluating a high-dimensional feature set containing over 100 candidate signals resulted in excessive training times without yielding measurable improvements in prediction accuracy. This finding motivated a targeted feature reduction strategy focused on eliminating highly collinear variables and isolating the primary drivers of system dynamics.
Identifying signals with high informational density is critical for expanding the model's look-ahead horizon. While the current analysis confirms that pilot pressures provide the most significant predictive information regarding future cylinder velocities, alternative control signals could offer additional foresight. For instance, utilizing direct joystick control currents was considered; however, this signal is inherently limited to manual operations. Because manual joystick inputs are bypassed or modified during active semi-autonomous functions, pilot pressures represent a more versatile and reliable foundation for the EDP, as they consistently reflect the actual hydraulic commands regardless of whether the machine is operated manually or autonomously.

5.2 Training and optimization suite

The progression from baseline variance decomposition to unconstrained multi-module discovery highlights a coupling between data composition, forecasting horizons, and architectural capacity. Synthesizing the empirical findings across all six optimization phases yields core insights for high-dimensional physical time-series modeling

and edge deployment.

5.2.1 Selection noise and multi-split validation

Phase 0 established that data-split variance (σ_{split}) systematically dominates initialization variance (σ_{seed}) by an order of magnitude. This volatility explains the sharp performance divergence observed between the Phase 2 screening and Phase 3 confirmation stages. While the permissive, single-trial screening in Phase 2 passed 93% to 100% of the candidate modules, the cross-validated confirmation rate in Phase 3 collapsed to just 7% for $k = 1$ and 36% for $k = 10$.

The *Physics Residual* module exemplifies this vulnerability. Although favored during individual optimization runs, it severely degraded model accuracy under rigorous cross-validation ($d_z \approx +18.5$ to $+22.6$). Contrary to the theoretical expectations, forcing the network to isolate unmodeled physical residuals suggests that domain-specific inductive biases caused overfitting to the localized sequence compositions of individual data splits. This underscores the necessity of multi-split gating over single-split HPO to prevent structural false positives.

5.2.2 Horizon-dependent inductive biases

Optimal network topology is intrinsically bound to the target forecasting horizon (k). In Phase 1, the structural advantage of the Hybrid TCN-LSTM backbone over standalone models at $k = 1$ diminished to a negligible ≤ 0.0009 composite units at $k = 10$, suggesting that fixed baseline inductive biases offer decaying utility as cumulative physical complexity scales over time.

Phase 3 results confirm this trend. For short-term predictions ($k = 1$), the baseline Hybrid model operates near its local performance ceiling, with only the *Multiscale TCN* module achieving statistical significance. Conversely, the long-term horizon ($k = 10$) yielded five significant structural improvements ($p < 0.001$). This shift indicates that long-range forecasting cannot rely solely on raw backbone capacity, instead it demands explicit structural regularization, such as the amplitude bounding of *STanh Activation* ($d_z = -6.30$) or the feature stabilization of *LayerNorm LSTM* ($d_z = -2.00$), to mitigate error compounding across extended temporal spans.

5.2.3 Feature versus architecture-driven generalization

A compelling point of synthesis is the decoupled success of Feature Group 2 (FG2). While architectural modifications and backbone scaling remained sensitive to selection noise and forecasting horizons, integrating explicit pressure time-derivatives ($\mathbf{\Delta}_{\text{pressure}}$) delivered a robust performance premium across all models and evaluation phases.

This behavior reveals a clear optimization hierarchy:

1. Maximizing the predictive quality of the input signal via targeted feature engineering provides a baseline of generalization that transcends specific model mechanics.

2. Architectural fine-tuning acts as a secondary layer of optimization that must be strictly tailored to the temporal constraints of k, minimizing the need to scale raw architectural complexity.

5.2.4 Evaluating regularization trade-offs via composite scoring

The near-linear scaling of physical error relative to the forecasting horizon—increasing from ≈ 1 mm/s at $k = 1$ to ≈ 10 mm/s at $k = 10$—uncovers a limitation in standard autoregressive or direct multi-step training paradigms. This drift indicates that relying exclusively on point-wise loss functions (such as standard MAE or MSE) fails to sustain trajectory continuity over longer horizons.
This tracking deficit validates the utility of the multi-objective composite score ($S_{\text{composite}}$). Traditional validation metrics frequently favor configurations like Phase 5's Trial 71, which optimized raw point-wise accuracy ($\text{MAE}_{\text{phys}} = 10.10$ mm/s) at the expense of trajectory continuity. By incorporating regularized derivative constraints (P_{95}, P_{99}), the optimization suite selected the $M_3 + M_5$ (*Temporal Differencing* and *Zoneout*) topology. Though this pairing sacrificed a nominal, sub-millimeter margin of instantaneous point accuracy (0.18 mm/s), it preserved the macro-level physical validity of the predicted trajectory.

5.2.5 Nonlinear synergy and the pitfalls of sequential filtering

The structural divergence between Phase 4 and Phase 5 exposes a defect in sequential ablation pipelines, which assumes that a module must demonstrate standalone merit to justify its inclusion. Phase 5's discovery of the four-module topology (M_1, M_2, M_3, M_6) at $k = 1$ refutes this linear assumption. Individually, modules like the *Attention Bridge* (M_1) and *LayerNorm* (M_6) failed to reach significance in Phase 3, yet when optimized concurrently in an unconstrained search space, they formed a cooperative network that halved the composite error score.
This nonlinear synergy reveals a "masking effect" in isolated testing, where an inductive bias is suppressed unless paired with complementary feature stabilization or attention mechanisms. This interplay is underlined by the behavior of the auxiliary loss components. In Phase 4's restricted search space, *Correlation Loss* and *Focal Loss* maintained moderate selection affinities, acting as optimization surrogates to artificially compensate for a limited topology. Once Phase 5 removed these structural boundaries, affinity for these complex loss configurations collapsed to near-zero. Specialized optimization objectives often act as structural crutches; when a network achieves authentic topological synergy, the underlying learning dynamics are naturally regularized.

5.2.6 Topological coordination versus pure capacity scaling

Early baseline developments highlighted severe diminishing returns when scaling network size: doubling the hidden layer capacity (from 128 to 256 units) inflated

the memory footprint while yielding an engineering premium of only ≈ 1 mm/s. This dynamic reflects a common pitfall: attempting to resolve unmodeled physical variations by increasing raw capacity.
The unconstrained search in Phase 5 provides a powerful alternative to brute-force scaling. At $k = 1$, rather than inflating the backbone dimensions, the framework introduced a multi-module coordination that expanded the parameter baseline by a negligible 3.8% (677,572 to 703,527 parameters). Despite this minimal footprint expansion, the model realized a 52.3% reduction in the composite error metric while keeping raw point-wise MAE flat. This decoupling demonstrates that physical time-series networks do not necessarily require more parameters to capture system dynamics; instead, they require strategically arranged pathways capable of routing temporal features across varying scales and attention boundaries.

5.2.7 Compression dynamics and edge deployment frontiers

Phase 6 translates these coordinated topologies into hardware-realizable configurations, mapping the Pareto efficiency frontier between model scale and metric accuracy. The evaluation of alternative compression pipelines demonstrates that the structural choices made during HPO directly dictate the downstream compression success.
Standard knowledge distillation successfully condenses the macro-level behavior of the optimal teacher model into a low-capacity student architecture, verifying that the student can absorb the regularized temporal routing discovered in Phase 5. However, PQT highlights a precision vulnerability: restricting quantization strictly to linear operators introduces localized quantization noise that disrupts the sub-millimeter tracking accuracy.
This limitation is overcome by quantization-aware training. By incorporating fake-quantization observers across both temporal convolutional and linear layers during fine-tuning, the optimization suite actively learns to counter precision drop-offs. QAT forces the network to preserve teacher-level trajectory stability and composite scores under a rigid INT8 execution format. This data-driven pipeline proves that model scale can be aggressively compressed without sacrificing physical validity, provided that structural coordination is optimized globally before hardware-precision boundaries are enforced.

5.3 Excavator dynamics predictor performance

This section evaluates the predictive accuracy of the trained architectures across varying temporal horizons, analyzes link-specific kinematic behaviors, and assesses the model's adaptability when transferring from simulation to real-world operational data.

5.3.1 Look-ahead horizon and predictive limitations

The models demonstrated high accuracy when predicting a single time step ($k = 1$, representing a 10 ms look-ahead horizon), routinely achieving a combined MAE

of $\leq$ 1 mm/s. However, single-step prediction rely heavily on the instantaneous cylinder velocity as a dominant indicator, since physical system states cannot change significantly within a 10 ms window. From a control systems perspective, a 10 ms predictive horizon is too short to counteract the inherent hydraulic actuation delays. To be practically viable for predictive control loops, the look-ahead horizon must extend beyond these core system delays.
Consequently, multi-step predictions were evaluated using both direct k-step ahead prediction and sequential horizon tracking, with the direct approach yielding the highest accuracy. The models trained on a 10-step look-ahead horizon (100 ms) maintained a reliable precision of $\leq$ 10 mm/s MAE. However, extending the horizon to 20 steps (200 ms) resulted in a rapid accuracy degradation. This performance drop aligns with the prior temporal lag analysis, which indicated that the informational density of the input signals decays significantly beyond a 20-sample threshold. While a 10-step prediction provides a viable window for enhancing vehicle control capabilities, longer prediction steps remain a key objective for future optimization.

5.3.2 Inertial delays and temporal precedence

The inherent time delay between control signal command and physical actuator movement varies significantly across the excavator's kinematic chain. The VAC team at CPAC Systems has documented an actuation delay longer than our current prediction horizon (100 ms), specifically for the boom cylinder, primarily due to the large physical mass and inertia it must manipulate. In theory, this suggests that boom dynamics could be anticipated further into the future. The large physical lag of this cylinder likely explains why the model achieved its highest predictive accuracy on the boom channel, as the control inputs possess a longer window of temporal precedence relative to the physical output.
However, this systemic delay is nonlinear and highly inconsistent during operations. The dead time varies depending on the operational context, such as whether the actuator is stationary, accelerating continuously, or undergoing rapid directional reversals.While these transient variations complicate the mapping function, the neural network architectures implicitly capture these nonlinearities during training.

5.3.3 Actuator variance and kinematic predictability

A clear hierarchy emerged regarding prediction accuracy across the individual links, with the boom cylinder achieving the highest precision, followed by the arm, and lastly the bucket. This performance gap is directly attributable to the varying kinematic profiles of each component. The boom experiences the lowest average velocities and accelerations because it is constrained by a massive mechanical inertia. This slower, more continuous motion naturally increases its predictability over longer look-ahead horizons.
In contrast, the bucket cylinder operates with minimal inertia, enabling rapid velocity transients and high accelerations. Because the bucket undergoes sudden, high-frequency state changes, its future trajectories are significantly more difficult for a data-driven model to accurately project over extended time horizons.

5.3.4 Transfer learning and fine-tuning

Utilizing transfer learning to fine-tune the models on real-world data resulted in accurate predictions when running inference on unseen physical excavator data. This demonstrates that the model possesses strong generalization capabilities, as a foundation trained purely on simulation data from one machine could be adapted to a different machine model using a limited amount of real-world data.
However, a direct consequence of this fine-tuning was that the models' performance on the original simulation data degraded. Whether this performance drop is caused by the discrepancy between the two excavator models or the broader differences between simulation and reality remains uninvestigated, and it is likely a combination of both. The results clearly indicate that a model trained solely on simulation data generalizes poorly to real-world data, while a model optimized for real-world conditions loses its accuracy when placed back into the simulation environment.
Additionally, the simulation environment completely omitted soil-tool interactions, thereby excluding a significant portion of a physical excavator's operational dynamics. These interactions are highly nonlinear and unpredictable, varying drastically depending on the specific characteristics of the material being manipulated. While the simulation training data was restricted entirely to "free-air" motions, the physical logged data included a combination of both free-air maneuvers and active soil operations to capture these missing external load characteristics.
However, because the available real-world dataset was recorded using only a single, uniform type of soil, the current results do not definitively confirm whether the EDP can maintain its predictive performance across diverse terrain compositions. Consequently, it remains unverified whether a single, generalized data-driven model can achieve high accuracy across a broad spectrum of material properties, or if the fine-tuning process inherently specializes the model weights to the specific material profile encountered during data logging.
This divergence highlights that a simulation environment, despite its high fidelity, cannot perfectly replicate all real-world characteristics and unmodeled disturbances. These results underscore the importance of sim-to-real transfer and demonstrate that testing in physical environments remains a necessity for validation. Ultimately, the findings confirm that a viable model can be established in simulation and subsequently adapted for physical deployment with minimal real-world data.

5.4 Simulink deployment and execution efficiency

The model was integrated into the Simulink environment by exporting and importing the architecture via the ONNX format, as described in section 3.6, enabling the model to generate predictions directly within the simulation loop. However, utilizing the MATLAB 2021b environment introduced significant execution speed limitations. While the Deep Learning Toolbox has advanced in newer software iterations, the legacy version available in MATLAB 2021b caused the simulation to execute slowly when the prediction model was active. Because this Simulink environment serves as the primary testing platform for the VAC team at CPAC Systems to validate control features, this reduction in simulation speed presents a practical challenge for

daily development workflows.
A clear correlation was observed between model size and execution latency. Large models with higher parameter counts and multiply-accumulate operations (MACs) caused substantial simulation delays. Conversely, shallower architectures with lower computational footprints had a significantly smaller impact on execution speed. Combined with the strict hardware constraints of onboard microcontrollers, these results highlight the critical importance of model compression techniques, such as quantization and knowledge distillation. Implementing these methods is essential not only to maintain viable simulation speeds during desktop development and testing but also to meet the execution requirements for final deployment on physical machine hardware.

5.5 Future work

While the developed EDP successfully demonstrates the viability of data-driven plant modeling, several engineering challenges must be addressed before real-world deployment. Future development should focus on closing the sim-to-real gap, optimizing the model for embedded hardware, and integrating the framework into active control and diagnostic loops. The following sections outline the key research avenues proposed to advance this work:

5.5.1 Domain adversarial training for balanced simulation and reality

While fine-tuning the EDP on real machine data corrects tracking errors, it degrades the model's performance in the Simulink simulation. To prevent the model from specializing too much in one environment, future work could explore Domain Adversarial Neural Networks. This training method forces the neural network to learn features that are common to both environments. During training, the model is penalized if its internal layers can distinguish whether the data originates from the simulator or the real machine. Implementing this technique would ensure that a single set of model weights stays accurate and useful for both virtual development and real-world deployment.

5.5.2 Onboard adaptive fine-tuning via meta-learning

The current transfer learning setup relies on offline calibration using a static dataset. However, physical excavators operate in dynamic environments where soil conditions change continuously. To allow the model to adapt in real time, future work should explore meta-learning frameworks like Model-Agnostic Meta-Learning (MAML). Instead of training the EDP for one specific setup, MAML optimizes the model so it can adapt to new dynamics with only a few updates. Deployed onboard, a MAML-initialized model could look at a brief window of recent sensor data and quickly adjust its weights on the fly, maintaining high accuracy as the machine switches between different soil types.

5.5.3 Hardware-aware optimization

The current optimization suite uses Optuna to minimize prediction errors without fully considering hardware limits. To ensure that the model can run on resource-constrained embedded ECUs, future work should implement Hardware-Aware Neural Architecture Search (HW-NAS). This approach modifies the optimization framework to use a multi-objective cost function that balances accuracy against execution speed and model size. By evaluating networks this way, the system can find a Pareto-optimal front. This allows engineers to easily choose a model that provides the highest possible accuracy while still running fast enough to fit within the machine's strict real-time control loops.

5.5.4 Anomalous load and fault detection diagnostics

An accurate, fine-tuned EDP can also be used for onboard vehicle diagnostics. Because the model learns exactly how a healthy machine behaves during standard digging, it can act as a real-time anomaly detector. Future research could implement a monitoring system that constantly compares actual sensor readings with the EDP's predictions. Normally, the error between the two should stay close to zero. A sudden, large jump in this error would indicate a problem. This could be used to flag external issues, like striking buried infrastructure, or internal hardware faults, such as hydraulic leaks, sticking valves, and sensor drift.

5.5.5 Integration into model predictive control frameworks

The ultimate goal of the EDP is to serve as the internal plant model for MPC. To move the model from a standalone predictor into an active control loop, future work must focus on extending its look-ahead horizon. Since hydraulic cylinders have significant delays, the EDP needs to reliably predict states further into the future than the current 100 ms window. Future research should explore stable multi-step rollout methods that match the controller's full optimization horizon without accumulating tracking errors.

Additionally, standard linear MPC solvers cannot handle the non-linear dynamics captured by a neural network. Future development will require integrating the EDP with a Non-linear MPC solver. This involves setting up the control loop so the solver can quickly run forward predictions through the TCN-LSTM model, check them against safety constraints, and output optimal cylinder commands. Successfully merging the EDP with an MPC loop will allow the machine to proactively compensate for hydraulic lag and dead-zones, leading to smoother and more efficient automated digging.

6
Conclusion

This thesis successfully demonstrated the development, optimization, and testing of a data-driven Excavator Dynamics Predictor for heavy machinery automation. Developed in collaboration with CPAC Systems AB, this framework addresses a key challenge in modern electro-hydraulic control design: the need for accurate, fast plant models that can predict complex, nonlinear machine behavior in real time.

By testing a variety of deep learning models for sequence data, this research identified an optimal neural network setup for hydraulic plant modeling, connecting to research question RQ3. A hybrid TCN-LSTM architecture achieved the best overall performance in prediction, while compacting the network to a manageable size. The hybrid architecture combines the local feature extraction of temporal convolutions with the continuous sequence-tracking capabilities of recurrent memory cells, which successfully kept an accuracy of a few mm/s (MAE) over 100 ms look-ahead horizons.

The automated optimization suite developed with Optuna simplified the model selection workflow, finding efficient network configurations within a large hyperparameter search space. This multi-phase search strategy handled data noise by using strict cross-validation over multiple data splits. This testing prevented the selection of over-fitted components which looked promising in single-trial tests but actually hurt the model's ability to generalize to new excavator variants. Additionally, the global search showed that neural networks do not need to be scaled up blindly with more hidden units to capture complex physical behaviors. Instead, optimizing with baseline complexity constraints to cut the composite prediction error by $94.1-40.3\%$ (k=1 and k=10 respectively).

The inclusion of physics-based modules (RQ1) did not inherently improve the performance of the EDP, and in some cases, actively reduced its predictive accuracy. This outcome does not necessarily mean that physics-aware or grey-box models generally perform worse in this domain. Instead, it likely indicates that the specific physical properties and simplified equations used did not provide the network with enough high-fidelity information. Despite these findings, incorporating physics-based predictions remains a promising area for future development. Grounding data-driven models in first-principles physics can provide useful structural direction, potentially enhancing the system's ability to generalize across unseen operating conditions and different excavator variants.

The practical value of the EDP was validated through real-world data benchmarks, providing insights into how the model handles unseen operating conditions and new excavator variants (RQ2). Although a reality gap exists due to unpredictable soil-tool interactions and simulation errors, a two-phase fine-tuning schedule proved

effective. This transfer learning strategy showed that a base model trained on simulated free-air data can be adapted to physical data using a limited amount of operational logs, which speeds up calibration for a new excavator variant (RQ4).
Importantly, the study mapped out the trade-offs between model size and prediction accuracy to meet the strict hardware limits of an industrial excavator microcontroller. While full-sized models would cause computational delays on the vehicle, compressing the chosen network using knowledge distillation combined with quantization-aware training resulted in a model with only 24 K parameters (down from 1.2 M). This implementation fits within the target 500 KB RAM allocation and performs with an accuracy of 1.6 cm/s (MAE) on simulation data. However, the number of MACs would still be too large to fit within the amount of clock cycles available on the current microcontroller unit.
Ultimately, this work confirms that well-optimized, data-driven plant models offer a strong alternative to traditional system identification methods. When paired with hardware-aware compression, these models provide a fast, scalable foundation for digital twins and future predictive control systems, marking a step forward for autonomous excavation.

Bibliography

[1] Fortune Business Insider. *Excavators Market Size, Share, Trends | Growth Report [2034]*. URL: https://www.fortunebusinessinsights.com/industry-reports/excavators-market-100861 (visited on 04/13/2026).

[2] Volvo CE. *Volvo Active Control | Dig-Assist | Volvo CE*. URL: https://www.volvoce.com/europe/en/volvo-services/dig-assist/volvo-active-control/ (visited on 04/14/2026).

[3] Gianluigi Pillonetto et al. "Deep networks for system identification: A survey". In: *Automatica* 171 (Jan. 2025), p. 111907. ISSN: 00051098. DOI: 10.1016/j.automatica.2024.111907. URL: https://linkinghub.elsevier.com/retrieve/pii/S0005109824004011 (visited on 01/23/2026).

[4] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. "Discovering governing equations from data: Sparse identification of nonlinear dynamical systems". In: *Proceedings of the National Academy of Sciences* 113.15 (Apr. 12, 2016), pp. 3932–3937. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.1517384113. arXiv: 1509.03580[math]. URL: http://arxiv.org/abs/1509.03580 (visited on 04/15/2026).

[5] Shinji Ishihara, Ryu Narikawa, and Toshiyuki Ohtsuka. "Automated Loading Operation for Mass-Production Hydraulic Excavators by Nonlinear Model Predictive Control". In: *IFAC-PapersOnLine* 56.2 (2023), pp. 5793–5798. ISSN: 24058963. DOI: 10.1016/j.ifacol.2023.10.554. URL: https://linkinghub.elsevier.com/retrieve/pii/S2405896323009217 (visited on 04/15/2026).

[6] Liangjun Zhang et al. "An autonomous excavator system for material loading tasks". In: *Science Robotics* 6.55 (June 23, 2021), eabc3164. ISSN: 2470-9476. DOI: 10.1126/scirobotics.abc3164. URL: https://www.science.org/doi/10.1126/scirobotics.abc3164 (visited on 04/15/2026).

[7] Jonas Weigand et al. "Hybrid Data-Driven Modelling for Inverse Control of Hydraulic Excavators". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). ISSN: 2153-0866. Sept. 2021, pp. 2127–2134. DOI: 10.1109/IROS51168.2021.9636269. URL: https://ieeexplore.ieee.org/document/9636269/ (visited on 01/26/2026).

[8] Hao Feng et al. "Data-driven hydraulic pressure prediction for typical excavators using a new deep learning SCSSA-LSTM method". In: *Expert Systems with Applications* 275 (May 2025), p. 127078. ISSN: 09574174. DOI: 10.1016/j.eswa.2025.127078. URL: https://linkinghub.elsevier.com/retrieve/pii/S0957417425007006 (visited on 01/28/2026).

[9] Ruiyang Zhang, Yang Liu, and Hao Sun. "Physics-informed multi-LSTM networks for metamodeling of nonlinear structures". In: *Computer Methods in Applied Mechanics and Engineering* 369 (Sept. 2020), p. 113226. ISSN: 00457825. DOI: 10.1016/j.cma.2020.113226. URL: https://linkinghub.elsevier.com/retrieve/pii/S0045782520304114 (visited on 01/27/2026).

[10] Jaemann Park et al. "Online Learning Control of Hydraulic Excavators Based on Echo-State Networks". In: *IEEE Transactions on Automation Science and Engineering* 14.1 (Jan. 2017), pp. 249–259. ISSN: 1558-3783. DOI: 10.1109/TASE.2016.2582213. URL: https://ieeexplore.ieee.org/document/7559750 (visited on 01/28/2026).

[11] Minhyeong Lee et al. "Precision Motion Control of Robotized Industrial Hydraulic Excavators via Data-Driven Model Inversion". In: *IEEE Robotics and Automation Letters* 7.2 (Apr. 2022), pp. 1912–1919. ISSN: 2377-3766. DOI: 10.1109/LRA.2022.3142389. URL: https://ieeexplore.ieee.org/document/9681181/ (visited on 01/26/2026).

[12] Pascal Egli and Marco Hutter. "Towards RL-Based Hydraulic Excavator Automation". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). ISSN: 2153-0866. Oct. 2020, pp. 2692–2697. DOI: 10.1109/IROS45743.2020.9341598. URL: https://ieeexplore.ieee.org/document/9341598/ (visited on 01/28/2026).

[13] Pascal Egli and Marco Hutter. "A General Approach for the Automation of Hydraulic Excavator Arms Using Reinforcement Learning". In: *IEEE Robotics and Automation Letters* 7.2 (Apr. 2022), pp. 5679–5686. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2022.3152865. URL: https://ieeexplore.ieee.org/document/9743573/ (visited on 01/26/2026).

[14] Weiwei Liu et al. "Review on control systems and control strategies for excavators". In: *Journal of Physics: Conference Series* 2301 (July 1, 2022), p. 012023. DOI: 10.1088/1742-6596/2301/1/012023.

[15] Lennart Ljung et al. "Deep Learning and System Identification". In: *IFAC-PapersOnLine* 53.2 (2020), pp. 1175–1181. ISSN: 24058963. DOI: 10.1016/j.ifacol.2020.12.1329. URL: https://linkinghub.elsevier.com/retrieve/pii/S2405896320317353 (visited on 02/05/2026).

[16] Ralf C. Staudemeyer and Eric Rothstein Morris. *Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks*. Sept. 12, 2019. DOI: 10.48550/arXiv.1909.09586. arXiv: 1909.09586[cs]. URL: http://arxiv.org/abs/1909.09586 (visited on 02/19/2026).

[17] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. *An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling*. Apr. 19, 2018. DOI: 10.48550/arXiv.1803.01271. arXiv: 1803.01271[cs]. URL: http://arxiv.org/abs/1803.01271 (visited on 02/19/2026).

[18] Kamil Nar and S. Shankar Sastry. *Persistency of Excitation for Robustness of Neural Networks*. Nov. 4, 2019. DOI: 10.48550/arXiv.1911.01043. arXiv: 1911.01043[cs]. URL: http://arxiv.org/abs/1911.01043 (visited on 04/15/2026).

[19] Milad Karimshoushtari and Carlo Novara. "Design of experiments for nonlinear system identification: A set membership approach". In: *Automatica* 119 (Sept. 2020), p. 109036. ISSN: 00051098. DOI: 10.1016/j.automatica.2020.109036. URL: https://linkinghub.elsevier.com/retrieve/pii/S000510982030234X (visited on 04/13/2026).

[20] Patrick Schober, Christa Boer, and Lothar A. Schwarte. "Correlation Coefficients: Appropriate Use and Interpretation". In: *Anesthesia & Analgesia* 126.5 (May 2018), pp. 1763–1768. ISSN: 0003-2999. DOI: 10.1213/ANE.0000000000002864. URL: https://journals.lww.com/00000539-201805000-00050 (visited on 04/15/2026).

[21] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. "Estimating mutual information". In: *Physical Review E* 69.6 (June 23, 2004), p. 066138. ISSN: 1539-3755, 1550-2376. DOI: 10.1103/PhysRevE.69.066138. URL: https://link.aps.org/doi/10.1103/PhysRevE.69.066138 (visited on 04/15/2026).

[22] C. Knapp and G. Carter. "The generalized correlation method for estimation of time delay". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 24.4 (Aug. 1976), pp. 320–327. ISSN: 0096-3518. DOI: 10.1109/TASSP.1976.1162830. URL: http://ieeexplore.ieee.org/document/1162830/ (visited on 04/15/2026).

[23] Scikit-learn Developers. *Permutation feature importance*. scikit-learn. URL: https://scikit-learn/stable/modules/permutation_importance.html (visited on 04/15/2026).

[24] Takuya Akiba et al. "Optuna: A Next-generation Hyperparameter Optimization Framework". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD '19: The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. Anchorage AK USA: ACM, July 25, 2019, pp. 2623–2631. ISBN: 978-1-4503-6201-6. DOI: 10.1145/3292500.3330701. URL: https://dl.acm.org/doi/10.1145/3292500.3330701 (visited on 04/29/2026).

[25] Lisha Li et al. *Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization*. June 18, 2018. DOI: 10.48550/arXiv.1603.06560. arXiv: 1603.06560[cs]. URL: http://arxiv.org/abs/1603.06560 (visited on 04/29/2026).

[26] Jacob Cohen. *Statistical power analysis for the behavioral sciences*. 2nd ed. Hillsdale, N.J: L. Erlbaum Associates, 1988. 567 pp. ISBN: 978-0-8058-0283-2.

[27] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. Springer Texts in Statistics. New York, NY: Springer New York, 2004. ISBN: 978-1-4419-1939-7 978-1-4757-4145-2. DOI: 10.1007/978-1-4757-4145-2. URL: http://link.springer.com/10.1007/978-1-4757-4145-2 (visited on 05/20/2026).

[28] Nan M. Laird and James H. Ware. "Random-Effects Models for Longitudinal Data". In: *Biometrics* 38.4 (1982), pp. 963–974. ISSN: 0006-341X. DOI: 10.2307/2529876. URL: https://www.jstor.org/stable/2529876 (visited on 05/20/2026).

[29] Yoav Benjamini and Yosef Hochberg. "Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing". In: *Journal of the Royal*

Statistical Society: Series B (Methodological) 57.1 (Jan. 1, 1995), pp. 289–300. ISSN: 0035-9246. DOI: 10.1111/j.2517-6161.1995.tb02031.x. URL: https://doi.org/10.1111/j.2517-6161.1995.tb02031.x (visited on 05/18/2026).

[30] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. *Distilling the Knowledge in a Neural Network*. Mar. 9, 2015. DOI: 10.48550/arXiv.1503.02531. arXiv: 1503.02531[stat.ML]. URL: http://arxiv.org/abs/1503.02531 (visited on 05/20/2026).

[31] Gido M. van de Ven, Nicholas Soures, and Dhireesha Kudithipudi. "Continual Learning and Catastrophic Forgetting". In: 2025, pp. 153–168. DOI: 10.1016/B978-0-443-15754-7.00073-0. arXiv: 2403.05175[cs.LG]. URL: http://arxiv.org/abs/2403.05175 (visited on 05/20/2026).

[32] Jeremy Howard and Sebastian Ruder. *Universal Language Model Fine-tuning for Text Classification*. May 23, 2018. DOI: 10.48550/arXiv.1801.06146. arXiv: 1801.06146[cs.CL]. URL: http://arxiv.org/abs/1801.06146 (visited on 05/20/2026).

A

Signals mapping reference

This appendix provides a reference for all signals utilized throughout this study. Table A.1 documents the relationship between the raw signals logged in the simulation and through the Speedgoat and their corresponding physical hydraulic components, actuator kinematics, or operator inputs.

Signal Name	**Physical Variable & Component**
Kinematics	
`CylVels_1`	Boom cylinder velocity
`CylVels_2`	Arm cylinder velocity
`CylVels_3`	Bucket cylinder velocity
`CylAcc_1`	Boom cylinder acceleration
`CylAcc_2`	Arm cylinder acceleration
`CylAcc_3`	Bucket cylinder acceleration
Main Actuator Hydraulic Pressures	
`CylPressures_1`	Boom cylinder piston-side pressure (P_A)
`CylPressures_2`	Arm cylinder piston-side pressure (P_A)
`CylPressures_3`	Bucket cylinder piston-side pressure (P_A)
`CylPressures_5`	Boom cylinder rod-side pressure (P_B)
`CylPressures_6`	Arm cylinder rod-side pressure (P_B)
`CylPressures_7`	Bucket cylinder rod-side pressure (P_B)
Primary Joystick Pilot Pressures	
`BmUp`	Boom pilot pressure (Raise)
`BmDwn`	Boom pilot pressure (Lower)
`AmIn`	Arm pilot pressure (Retract)
`AmOut`	Arm pilot pressure (Extend)
`BktIn`	Bucket pilot pressure (Curl)
`BktOut`	Bucket pilot pressure (Dump)
Conflux Pilot Pressures	
`BmUpConf`	Boom conflux pilot pressure (Raise)
`BmDwnConf`	Boom conflux pilot pressure (Lower)
`AmInConf`	Arm conflux pilot pressure (Retract)
`AmOutConf`	Arm conflux pilot pressure (Extend)
`BktInConf`	Bucket conflux pilot pressure (Curl)
`BktOutConf`	Bucket conflux pilot pressure (Dump)

Table A.1: Mapping of logged controller signal names to physical excavator kinematics and hydraulic components.

B

Models presented in Results

B.1 Hybrid TCN-LSTM (offset=10)

```
==========================================================================================================
Layer (type:depth-idx)                                   Input Shape      Output Shape     Param #
==========================================================================================================
ExcavatorHybridTCNLSTM                                   [1, 50, 45]      [1, 3]           --
|-Sequential: 1-1                                        [1, 30, 50]      [1, 32, 50]      --
|    \-TemporalBlock: 2-1                                [1, 30, 50]      [1, 256, 50]     --
|    |    \-Sequential: 3-1                              [1, 30, 50]      [1, 256, 50]     220,672
|    |    \-Conv1d: 3-2                                  [1, 30, 50]      [1, 256, 50]     7,936
|    \-TemporalBlock: 2-2                                [1, 256, 50]     [1, 128, 50]     --
|    |    \-Sequential: 3-3                              [1, 256, 50]     [1, 128, 50]     147,968
|    |    \-Conv1d: 3-4                                  [1, 256, 50]     [1, 128, 50]     32,896
|    \-TemporalBlock: 2-3                                [1, 128, 50]     [1, 64, 50]      --
|    |    \-Sequential: 3-5                              [1, 128, 50]     [1, 64, 50]      37,120
|    |    \-Conv1d: 3-6                                  [1, 128, 50]     [1, 64, 50]      8,256
|    \-TemporalBlock: 2-4                                [1, 64, 50]      [1, 32, 50]      --
|    |    \-Sequential: 3-7                              [1, 64, 50]      [1, 32, 50]      9,344
|    |    \-Conv1d: 3-8                                  [1, 64, 50]      [1, 32, 50]      2,080
|-Sequential: 1-2                                        [1, 30, 50]      [1, 32, 50]      --
|    \-TemporalBlock: 2-5                                [1, 30, 50]      [1, 256, 50]     --
|    |    \-Sequential: 3-9                              [1, 30, 50]      [1, 256, 50]     220,672
|    |    \-Conv1d: 3-10                                 [1, 30, 50]      [1, 256, 50]     7,936
|    \-TemporalBlock: 2-6                                [1, 256, 50]     [1, 128, 50]     --
|    |    \-Sequential: 3-11                             [1, 256, 50]     [1, 128, 50]     147,968
|    |    \-Conv1d: 3-12                                 [1, 256, 50]     [1, 128, 50]     32,896
|    \-TemporalBlock: 2-7                                [1, 128, 50]     [1, 64, 50]      --
|    |    \-Sequential: 3-13                             [1, 128, 50]     [1, 64, 50]      37,120
|    |    \-Conv1d: 3-14                                 [1, 128, 50]     [1, 64, 50]      8,256
|    \-TemporalBlock: 2-8                                [1, 64, 50]      [1, 32, 50]      --
|    |    \-Sequential: 3-15                             [1, 64, 50]      [1, 32, 50]      9,344
|    |    \-Conv1d: 3-16                                 [1, 64, 50]      [1, 32, 50]      2,080
|-LSTM: 1-3                                              [1, 50, 64]      [1, 50, 256]     329,728
|-Dropout: 1-4                                           [1, 50, 256]     [1, 50, 256]     --
|-TemporalAttentionPool: 1-5                             [1, 50, 256]     [1, 256]         --
|    \-Linear: 2-9                                       [1, 50, 256]     [1, 50, 1]       257
|-CylinderFeatureGate: 1-6                               [1, 512]         [1, 512]         3,072
|-ModuleList: 1-7                                        --               --               --
|    \-Sequential: 2-10                                  [1, 512]         [1, 64]          --
|    |    \-Linear: 3-17                                 [1, 512]         [1, 64]          32,832
|    |    \-ReLU: 3-18                                   [1, 64]          [1, 64]          --
|    |    \-Linear: 3-19                                 [1, 64]          [1, 64]          4,160
|    |    \-ReLU: 3-20                                   [1, 64]          [1, 64]          --
|    \-Sequential: 2-11                                  [1, 512]         [1, 64]          --
|    |    \-Linear: 3-21                                 [1, 512]         [1, 64]          32,832
|    |    \-ReLU: 3-22                                   [1, 64]          [1, 64]          --
|    |    \-Linear: 3-23                                 [1, 64]          [1, 64]          4,160
|    |    \-ReLU: 3-24                                   [1, 64]          [1, 64]          --
|    \-Sequential: 2-12                                  [1, 512]         [1, 64]          --
|    |    \-Linear: 3-25                                 [1, 512]         [1, 64]          32,832
|    |    \-ReLU: 3-26                                   [1, 64]          [1, 64]          --
|    |    \-Linear: 3-27                                 [1, 64]          [1, 64]          4,160
|    |    \-ReLU: 3-28                                   [1, 64]          [1, 64]          --
|-ModuleList: 1-8                                        --               --               --
|    \-Linear: 2-13                                      [1, 64]          [1, 1]           65
|    \-Linear: 2-14                                      [1, 64]          [1, 1]           65
|    \-Linear: 2-15                                      [1, 64]          [1, 1]           65
|-STanh: 1-9                                             [1, 3]           [1, 3]           3
==========================================================================================================
Total params: 1,376,775
Trainable params: 1,376,775
Non-trainable params: 0
Total mult-adds (M): 22.50
==========================================================================================================
Input size (MB): 0.01
Forward/backward pass size (MB): 0.49
Params size (MB): 2.19
Estimated Total Size (MB): 2.69
==========================================================================================================
```

Table B.1: Model Architecture - Hybrid TCN-LSTM (offset=10)

B.1.1 Components and Hyperparameters

```
============================================================
DATA CONFIG
============================================================
  window_size                   50
  target_variable               velocity
  target_mode                   delta
  load_physics_features         True
  physics_variant               force
  prediction_horizon            1
  target_offset                 10
  test_size                     0.2
  random_state                  37
  csv_subset                    999
  startup_skip                  600
  feature_groups                feature_groups_lean_acc_dp.json
  augment_enabled               False
  augment_window_shift          False
  window_shift_max              0
  augment_time_warp             False
  time_warp_strength            0.1

============================================================
MODEL CONFIG
============================================================
  arch                          Hybrid
  variant                       hybrid_tcn_lstm
  units                         256
  num_layers                    4
  dropout_rate                  0.016987413933211414
  use_physics_base              False
  lambda_residual               1.0
  lambda_phys                   0.0
  head_hidden_layers            2
  use_cross_cyl_attn            False
  use_cyl_gate                  True
  cyl_gate_type                 diagonal
  use_temporal_diff             False
  use_velocity_skip             False
  kernel_size                   3
  num_channels                  (64, 32)
  use_multiscale_tcn            True
  use_glu_head                  False
  use_gru_backbone              False
  use_layernorm_lstm            False
  use_zoneout                   False
  zoneout_rate                  0.1
  use_se_block                  False
  se_reduction                  4
  use_gap                       False
  use_attention_bridge          False
  bridge_heads                  4

============================================================
TRAINING CONFIG
============================================================
  epochs                        25
  lr                            0.0003
  batch_size                    32
  max_grad_norm                 1.0
  weight_decay                  0.0005
  criterion:
    type                        huber
    huber_delta                 1.5
    weight_power                2.0
    adaptive_weights            False
    adaptive_blend              0.3
    adaptive_max_ratio          3.0
    uncertainty_guided          False
    use_focal                   False
    focal_gamma                 2.0
    smoothness_weight           0.05
    derivative_weight           0.1
    correlation_weight          0.0
  scheduler:
    enabled                     True
    type                        plateau
    factor                      0.5
    patience                    5
    min_lr                      1e-07
    threshold                   0.0001
    warmup_epochs               0
  enable_worst_error_metrics    True
  tracking_metric               phys_mae
  fitness_mode                  robust
  fitness_tail_n                3
  robust_fitness_alpha          0.5
  robust_fitness_beta           0.005
  robust_use_history_proxy      True
  num_workers                   2
  use_amp                       True
  pin_memory                    True
  persistent_workers            True
```

```
  early_stopping_patience      8
  horizon_decay                1.0
  sensor_jitter_std            0.02
  use_swa                      False
  swa_lr                       0.0001

============================================================
ACTIVATION CONFIG
============================================================
  output_activation            stanh
  stanh_init_a                 2.5
  stanh_min_a                  1.01
  stanh_max_a                  5.0
```

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY