Khajeh Nasir Toosi University of Technology

Machine learning final project proposal

Slip Estimation for Mecanum-wheeled robot

Instructor: Dr.Aliyari

Student:

Asal Sanei 40308334

Google colab		
Link 1	Square path	Square
Link 2	Circular path	Circle
Github link		
Github folder		
PowerPoint in Canva		
PowerPoint		

Contents

Figures

Tables

Abstract

Mecanum-wheeled robots are known for their exceptional maneuverability, but their unique design leaves them highly vulnerable to slip, a factor that significantly hinders path tracking accuracy and control system performance. This project aims to address this critical challenge by developing a robust and accurate slip estimation approach. Our core objective is to create a machine learning model that can predict and compensate for slip in real-time under a variety of operating conditions. The proposed methodology involves using two cameras and image processing to gather precise real-time position data, which is then combined with encoder data to calculate actual slip. This discrepancy provides the essential training data for a machine learning model. We plan to explore various approaches for slip modeling, including Regression, polynomial regression, GPR, SVR, MLP, LSTM, NARX-LSTM, CNN-LSTM and CNN-LSTM-Autoencoder. The successful implementation of this project is expected to result in a highly accurate slip estimation module and a notable improvement in the robot's overall path tracking capabilities.

Chapter 1

Introduction

1. Introduction

Mecanum-wheeled robots are a type of omnidirectional mobile robot, which means that can move in any direction without changing their orientation. This unique capability comes from their specialized wheels, which have rollers mounted at a 45-degree angle to the wheel's axis. This design allows the robot to generate forces in multiple directions simultaneously, enabling forward, sideways, diagonal movements, and even on-the-spot rotation.

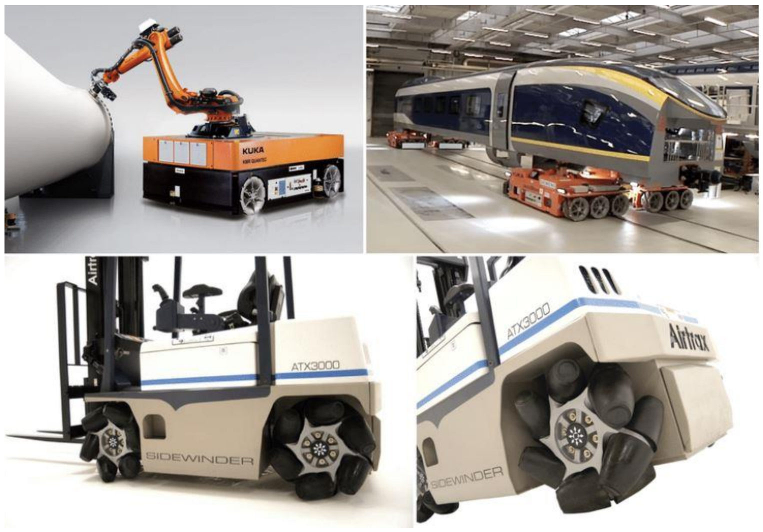


Figure 1: Application of Mecanum wheeled robots

Mecanum wheels offer unparalleled maneuverability in tight and complex environments, making them ideal for tasks requiring precise positioning and navigation. However, their design also makes them particularly susceptible to slip, which is the primary challenge. Accurate slip estimation is crucial for enhancing the path tracking quality and overall control of wheeled mobile robots.

Figure 2: ARAS Lab's robo-omni on the field

2. Problem Statement

Slip in mobile robots, particularly Mecanum robots, leads to inaccuracies in path tracking and degrades control system performance. Existing methods often rely on ideal no-slip assumptions or less accurate sensor data. The objective is to develop a highly accurate slip estimation model that can predict and compensate for slip under various operating conditions.

Figure 3:Mecanum wheels

Neural Networks and machine learning approaches that can be used for estimation include:

- Regression(linear and polynomial)
- GPR
- Decision tree
- SVR
- MLP
- NARX
- LSTM

2.3 Expected Outcomes

- A robust and accurate slip estimation module for Mecanum-wheeled mobile robots.
- Improved path tracking performance of the robot through real-time slip compensation.

Chapter 2

Literature review

(Nourizadeh, Stevens , J, & N, 2021) propose a deep-learning method to estimate the slippage ratio for skid-steering robots in outdoor environments. The technique uses low-cost proprioceptive sensors and is designed to be unaffected by weather conditions. The method consists of two main stages: data collection and training a deep learning model. The model is capable of estimating the slippage for the entire robot, not just individual wheels.

A Pioneer 3-AT robot was used to collect data in a real-world environment on both grass and gravel. Two different types of tires—solid and pneumatic—were used for the experiments. The robot was equipped with a 100Hz Xsens IMU and its default 10Hz wheel encoders. A 5Hz RTK-GPS system was employed to collect ground-truth velocity data, which was used to calculate the slippage ratio for training the model.

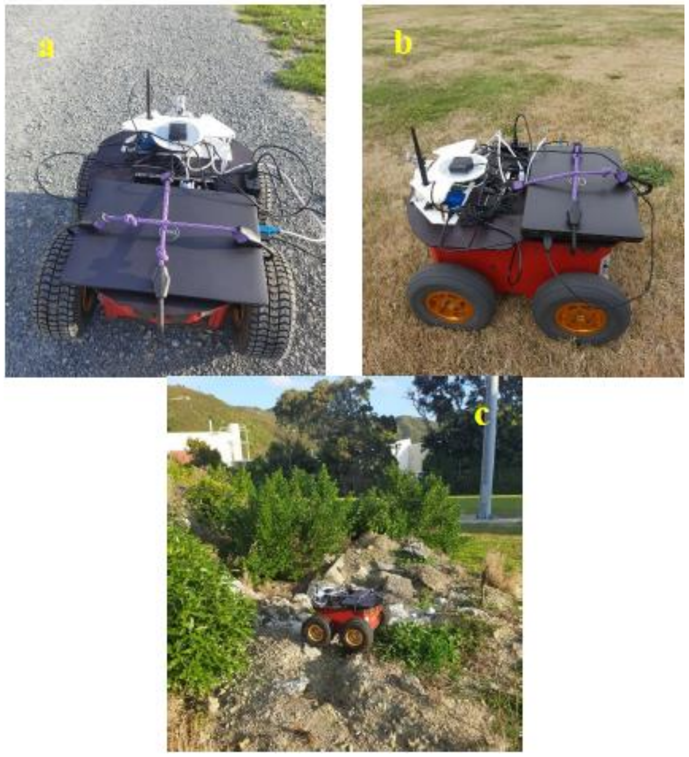
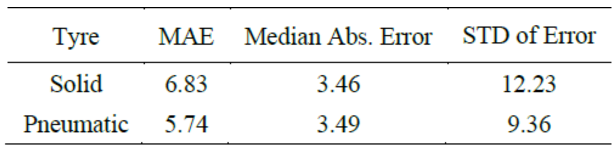


Figure 4: The Pioneer 3-AT driving on (a) gravel with the pneumatic tyre, (b) grass with the solid tyre and (c) rough terrain with the pneumatic tyre.

The paper proposes a CNN-LSTM model designed to estimate the slippage ratio. The model architecture includes 1-D convolutional layers, a bidirectional LSTM network, and dense layers. The convolutional layers are used to extract important features from the time-series data, while the LSTM network is chosen for its ability to detect both short- and long-term dependencies in the data. The model was trained for 5,000 epochs using the NAdam optimizer.

The CNN-LSTM model achieved a Mean Absolute Error (MAE) of 6.83% for the solid tires and 5.74% for the pneumatic tires. The median absolute error for both tire types was less than 3.5%. The performance was lower for the solid tire at slippage levels between 40-70% due to less available data in that range. The study found that estimation error was higher at lower linear velocities (less than 100 mm/sec) but showed no clear correlation with the robot's pitch angle.

Table 1:The performance of the CNN-LSTM model for both tyres on the test set (Abs.: Absolute, STD: Standard Deviation).



Tyre	MAE	Median Abs. Error	STD of Error
Solid	6.83	3.46	12.23
Pneumatic	5.74	3.49	9.36

(Nourizadeh, McFadden, J, & N, 2023) present a novel, real-world feasible method for in-situ slip estimation on wheeled mobile robots (WMRs) in outdoor, uneven terrains. The key innovation is using only two low-cost proprioceptive sensors—an IMU and a wheel encoder—combined with deep learning methods to address the need for in-situ slip estimation without requiring prior knowledge of soil properties. The methodology is categorized as real-world feasible as it is independent of environmental conditions like weather or lighting.

A Pioneer-3AT robot with both solid and pneumatic tires was used for data collection in multiple outdoor environments, including gravel, grass at the Robinson Research Institute (RRI), and clay and volcanic sand at a regional forest. This ensured a wide range of slippage from almost 0% to 100%. An RTK-GPS provided ground-truth velocity data for calculating slip, while a 100Hz Xsens IMU and the robot's default 10Hz wheel encoders were used for input features.

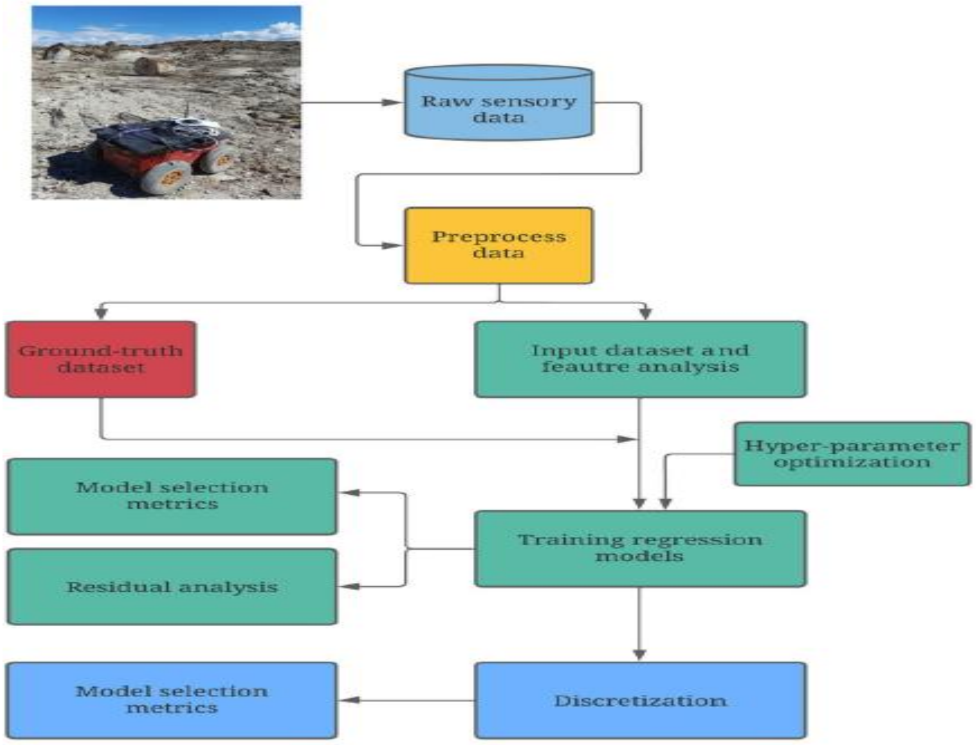


Figure 5: Slip estimation method

The study developed and compared four deep-learning models for a regression task: LSTM, LSTM-Autoencoder (LSTM-AE), CNN-LSTM, and CNN-LSTM-Autoencoder (CNN-LSTM-AE). The CNN-LSTM-AE model combines the advantages of convolutional layers for feature extraction and autoencoders for improved accuracy. The models were also used for a classification task by discretizing the regression output into three slip categories: low (≤30%), moderate (>30% and ≤60%), and high (>60%). A comparison was also made against conventional machine learning algorithms like AdaBoost, SVM, Random Forest, Decision Tree, and ANN.

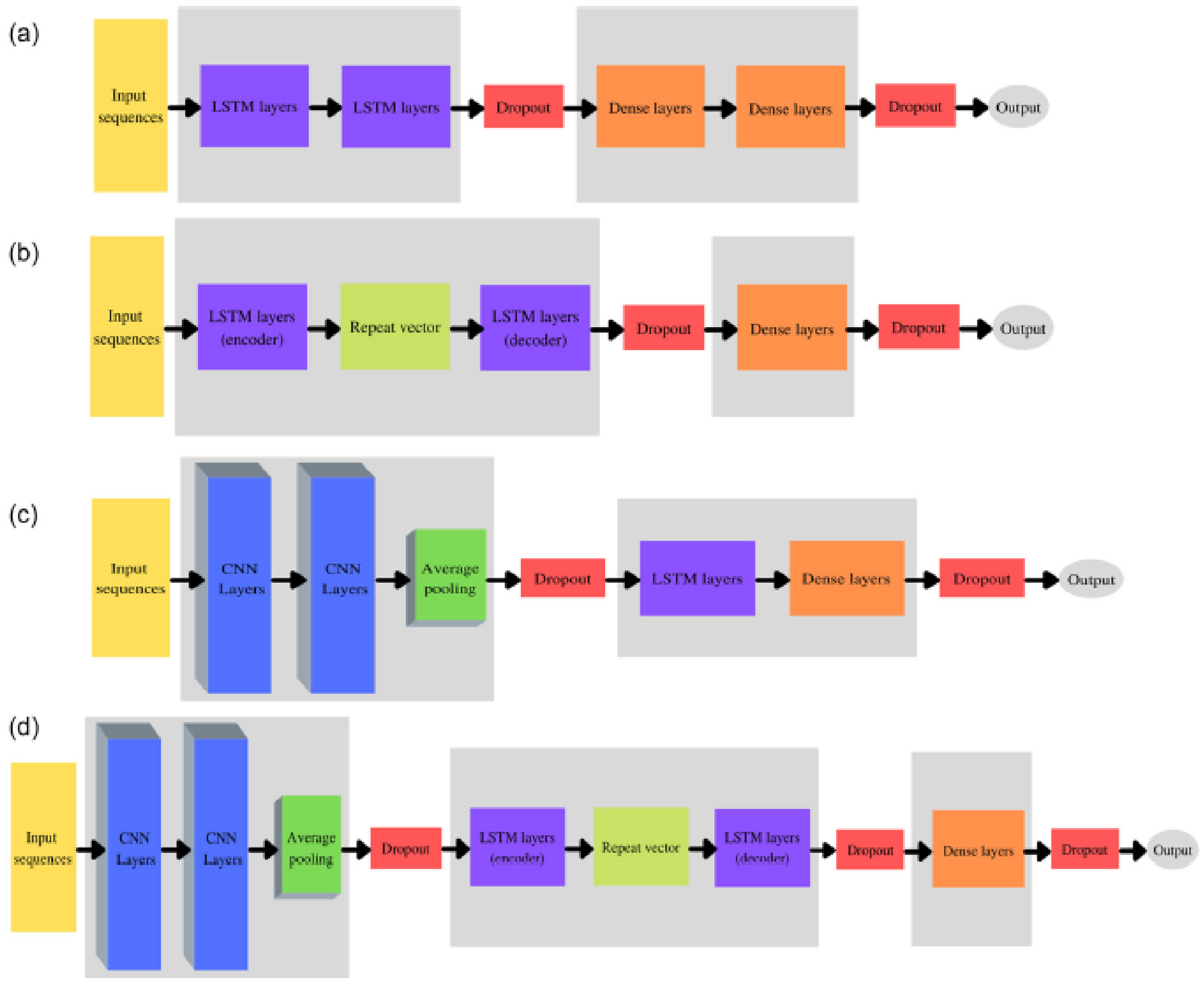


Figure 6: The architecture of proposed models, (a) LSTM, (b) LSTM-AE, (c) CNN-LSTM, and (d) CNN-LSTM-AE

Deep learning models consistently outperformed conventional machine learning algorithms in both regression and classification tasks. For the solid tire, the CNN-LSTM model achieved the best regression performance with an SMAPE of 11.31%, an MAE of 5.77%, and an F1-score of 89.15% in the classification task. For the pneumatic tire, the CNN-LSTM-AE model performed best in regression with an SMAPE of 18.86% and MAE of 5.49%, and an F1-score of 88.01% in the classification task. The study found that estimation errors were higher at low speeds (less than 100 mm/s).

(Ramon , Mirko , & Karl, 2019) provide a comprehensive comparison of eleven well-known machine learning (ML) algorithms for estimating discrete slip events for individual wheels on planetary rovers. The methodology relies solely on proprioceptive sensors and is independent of environmental conditions. The study also analyzes the influence of various factors like rover speed, terrain type, and tire type on the performance of the algorithms.

The Lunar All-Terrain Utility Vehicle (LATUV) rover was used for data collection on uneven terrain. The experiments were conducted at two sites: a concrete factory with thin sand and an abandoned mine with dry, compact gravel. The tests were performed with both off-road and smooth tires. The rover was equipped with an RTK-GPS for ground-truth data, as well as IMU sensors on each wheel and motor current-draw sensors.

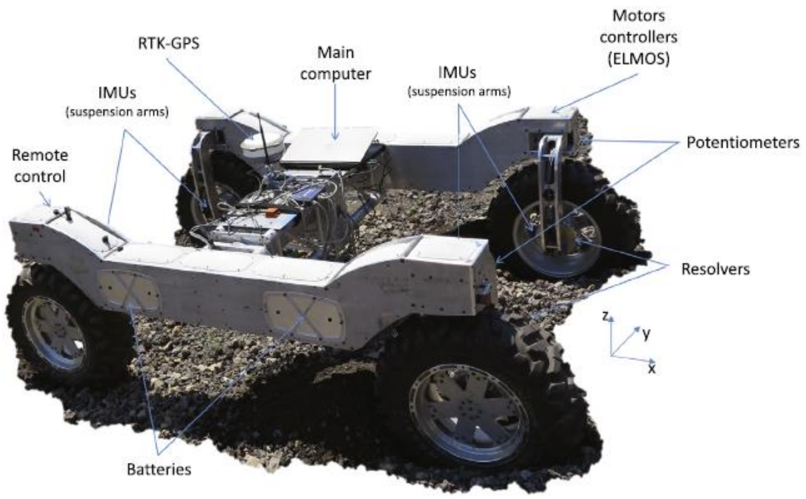


Figure 7: LATUV rover with the sensors and hardware used in this research.

The problem of slip estimation is formulated as a classification problem with three discrete slip classes: low (≤0.3), moderate (>0.3 and ≤0.6), and high (>0.6). The study compared eleven ML algorithms from various categories, including linear models (e.g., Logistic Regression), decision trees, instance-based algorithms (e.g., KNN), and ensemble methods (e.g., Random Forests). Features were derived from the motor current, IMU linear acceleration (X), vertical acceleration (Z), and pitch rate (dy/dt).

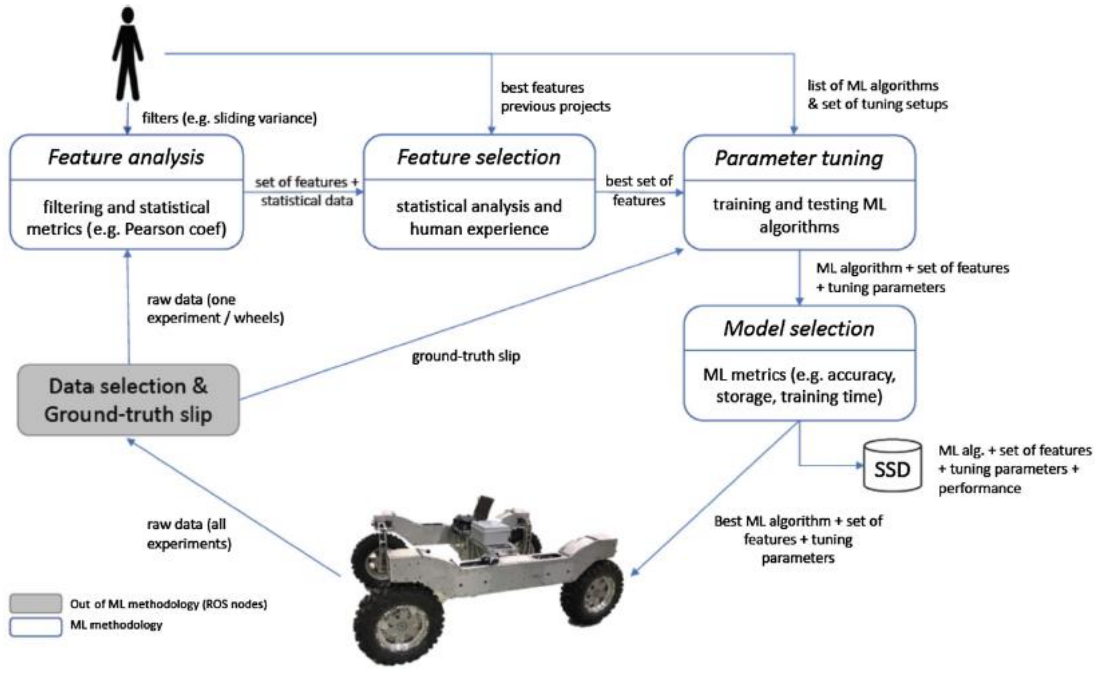

filters (e.g. sliding variance)
best features
previous projects
list of ML algorithms
& set of tuning setups
Feature analysis
filtering and statistical metrics (e.g. Pearson coef)
set of features + statistical data
Feature selection
statistical analysis and human experience
best set of features
Parameter tuning
training and testing ML algorithms
ML algorithm + set of features + tuning parameters
raw data (one experiment / wheels)
Model selection
ML metrics (e.g. accuracy, storage, training time)
ground-truth slip
Data selection & Ground-truth slip
SSD
ML alg. + set of features + tuning parameters + performance
raw data (all experiments)
Best ML algorithm + set of features + tuning parameters
Out of ML methodology (ROS nodes)
ML methodology

Figure 8: Methodology for estimating wheel slip using machine learning.

The study found that the Decision Tree (DT) algorithm provided the highest overall accuracy at 84.15%. However, the AdaBoost (ADA) algorithm, with a slightly lower accuracy of 83.65%, was considered the best choice for deployment on a planetary rover because it required the lowest storage. The research also concluded that ML algorithms performed better at higher wheel speeds and when tested on sand rather than gravel. Correlation between sensor signals and slip was slightly higher with off-road tires compared to smooth tires and for IMUs mounted on the front wheels.

(R & M, 2018) present a methodology for predicting slippage and its associated uncertainty for individual wheels on off-road mobile robots. It frames slip as a random variable and uses machine learning regression algorithms to predict a continuous slip value along with its variance, rather than a discrete class. This approach relies on proprioceptive sensors, making it independent of lighting conditions.

The study used a single-wheel testbed at MIT, which was equipped with a Mars Science Laboratory (MSL) flight spare wheel. The testbed included a soil bin with Mars regolith simulant and was used to induce slip under various conditions. An IMU, a torque sensor, and a displacement sensor were used for data collection. The IMU was mounted directly on the wheel.

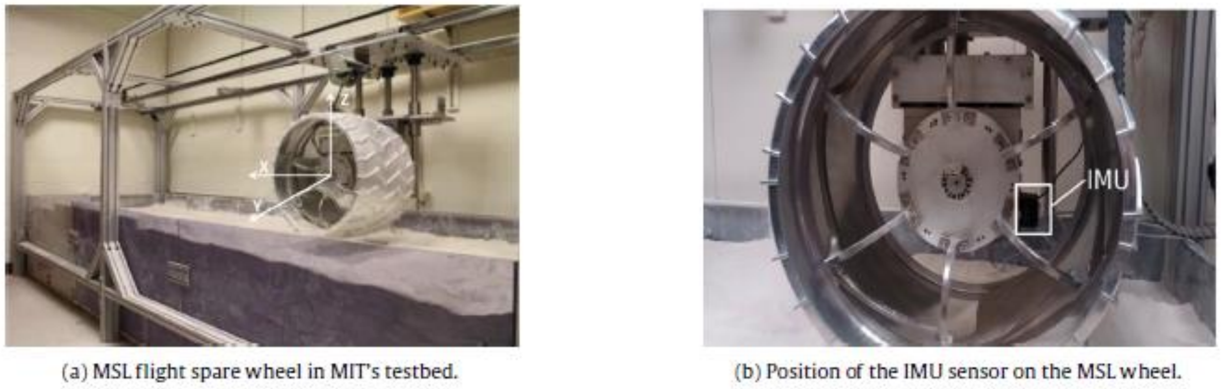
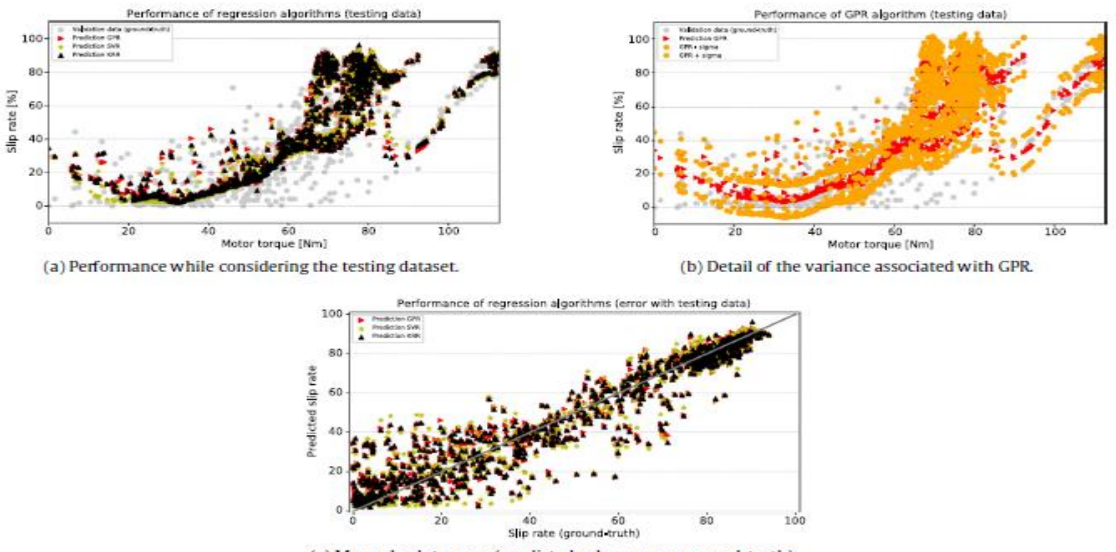

Z
X
Y
IMU
(a) MSL flight spare wheel in MIT's testbed.
(b) Position of the IMU sensor on the MSL wheel.

Figure 9: Single-wheel testbed developed by RMG-MIT and used for collecting experimental data. The IMU constitutes the primary sensor in the proposed methodology.

The methodology uses three machine learning regression algorithms: Gaussian Process Regression (GPR), Support Vector Regression (SVR), and Kernel Ridge Regression (KRR). The input features included the absolute value of the wheel torque and the variance of IMU signals (linear acceleration, pitch rate, and vertical acceleration). The model was trained using 70% of the data, with the remaining 30% used for testing. A comparison was also made against a machine learning classification approach using a Support Vector Machine (SVC) model.

All three regression algorithms performed well, with GPR, SVR, and KRR achieving similar accuracy scores around 80%. The GPR algorithm was highlighted for its ability to provide the variance associated with each prediction, which is useful for tasks like robust motion control and path planning. The study also found that combining IMU and torque-related features resulted in a significantly lower mean absolute error compared to using only torque-related features. While the classification model (SVC) ran faster, the regression models provided a continuous value and uncertainty, offering a different advantage



(a) Performance while considering the testing dataset.

(b) Detail of the variance associated with GPR.

(c) Mean absolute error (predicted values versus ground-truth).

Figure 10: Performance of the machine learning regression algorithms (GPR, SVR, KRR).

Chapter 3

Methodology

3.1 Data collection

The data is collected by Sohrab Allahyari ,Graduate MSc Student at Advanced Robotics & Automated Systems (ARAS) Laboratory at Khaje Nasir university.

google scholar:

Sohrab Allahyari_Google scholar

3.1.1 Approach

The data collection process involves using multiple sensors to compare a robot's planned movement with its actual movement. A key part of this process is measuring the robot's actual position with a high degree of precision.
In this study this was achieved through stereo camera image processing that provided millimeter-level accuracy. The robot's expected movement is determined from its wheel encoders. By calculating the difference between these two data sources, the amount of slip was determined.

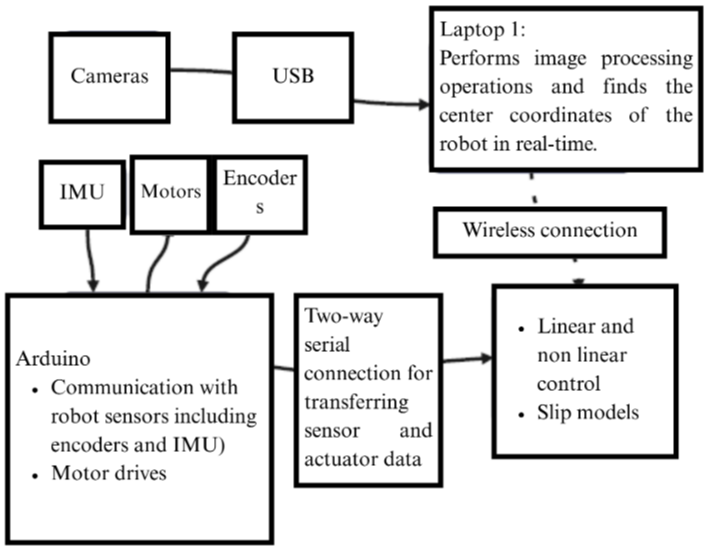

Cameras
USB
Laptop 1:
Performs image processing operations and finds the center coordinates of the robot in real-time.
IMU
Motors
Encoders
Wireless connection
Arduino
• Communication with robot sensors including encoders and IMU)
• Motor drives
Two-way serial connection for transferring sensor and actuator data
• Linear and non linear control
• Slip models

Figure 11: Data collection for slip

3.1.2 Inputs

14 features serve as inputs, derived from the robot's movement data:

- Wheel speed (4 values, one for each wheel)
- Wheel acceleration (4 values, one for each wheel)
- Robot position based on encoder data (3 values: x, y, and rotation ϕ)

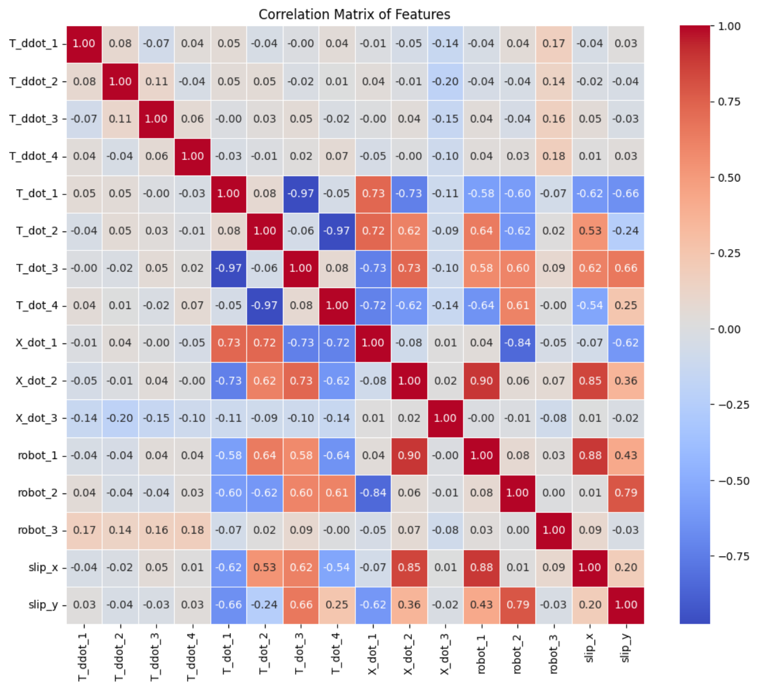
- Robot speed based on encoder data (3 values)

3.1.3 Outputs

- Slip in X direction
- Slip in Y direction

It should be noted that slip estimation in the φ coordinate was ignored because obtaining this angle from image processing was not possible.

3.1.4 Preprocessing



	T_ddot_1	T_ddot_2	T_ddot_3	T_ddot_4	T_dot_1	T_dot_2	T_dot_3	T_dot_4	X_dot_1	X_dot_2	X_dot_3	robot_1	robot_2	robot_3	slip_x	slip_y
T_ddot_1	1.00	0.08	-0.07	0.04	0.05	-0.04	-0.00	0.04	-0.01	-0.05	-0.14	-0.04	0.04	0.17	-0.04	0.03
T_ddot_2	0.08	1.00	0.11	-0.04	0.05	0.05	-0.02	0.01	0.04	-0.01	-0.20	-0.04	-0.04	0.14	-0.02	-0.04
T_ddot_3	-0.07	0.11	1.00	0.06	-0.00	0.03	0.05	-0.02	-0.00	0.04	-0.15	0.04	-0.04	0.16	0.05	-0.03
T_ddot_4	0.04	-0.04	0.06	1.00	-0.03	-0.01	0.02	0.07	-0.05	-0.00	-0.10	0.04	0.03	0.18	0.01	0.03
T_dot_1	0.05	0.05	-0.00	-0.03	1.00	0.08	-0.97	-0.05	0.73	-0.73	-0.11	-0.58	-0.60	-0.07	-0.62	-0.66
T_dot_2	-0.04	0.05	0.03	-0.01	0.08	1.00	-0.06	-0.97	0.72	0.62	-0.09	0.64	-0.62	0.02	0.53	-0.24
T_dot_3	-0.00	-0.02	0.05	0.02	-0.97	-0.06	1.00	0.08	-0.73	0.73	-0.10	0.58	0.60	0.09	0.62	0.66
T_dot_4	0.04	0.01	-0.02	0.07	-0.05	-0.97	0.08	1.00	-0.72	-0.62	-0.14	-0.64	0.61	-0.00	-0.54	0.25
X_dot_1	-0.01	0.04	-0.00	-0.05	0.73	0.72	-0.73	-0.72	1.00	-0.08	0.01	0.04	-0.84	-0.05	-0.07	-0.62
X_dot_2	-0.05	-0.01	0.04	-0.00	-0.73	0.62	0.73	-0.62	-0.08	1.00	0.02	0.90	0.06	0.07	0.85	0.36
X_dot_3	-0.14	-0.20	-0.15	-0.10	-0.11	-0.09	-0.10	-0.14	0.01	0.02	1.00	-0.00	-0.01	-0.08	0.01	-0.02
robot_1	-0.04	-0.04	0.04	0.04	-0.58	0.64	0.58	-0.64	0.04	0.90	-0.00	1.00	0.08	0.03	0.88	0.43
robot_2	0.04	-0.04	-0.04	0.03	-0.60	-0.62	0.60	0.61	-0.84	0.06	-0.01	0.08	1.00	0.00	0.01	0.79
robot_3	0.17	0.14	0.16	0.18	-0.07	0.02	0.09	-0.00	-0.05	0.07	-0.08	0.03	0.00	1.00	0.09	-0.03
slip_x	-0.04	-0.02	0.05	0.01	-0.62	0.53	0.62	-0.54	-0.07	0.85	0.01	0.88	0.01	0.09	1.00	0.20
slip_y	0.03	-0.04	-0.03	0.03	-0.66	-0.24	0.66	0.25	-0.62	0.36	-0.02	0.43	0.79	-0.03	0.20	1.00

Figure 12: Correlation matrix

While the input features show strong internal correlations, the key relationships for slip estimation—the correlation between the input features and the outputs (Slip X and Slip Y)—are not strong. This suggests that a simple linear model might not be effective at predicting slip. Instead, the problem requires more complex models, such as deep learning approaches, that can capture the non-linear relationship between the input features and the slip outputs.

3.2 Machine learning approaches

Regression (linear and polynomial)

Two well-known regression models are implemented in this project.

GPR

Gaussian Process Regression (GPR) is a supervised learning technique that uses a Bayesian approach for regression and probabilistic classification. It is considered an infinite-dimensional generalization of a multivariate Gaussian distribution.

Gaussian Process Regression (GPR) is a type of machine learning model that uses a

Bayesian approach for prediction. The core idea is to start with a prior distribution, which represents initial beliefs about the possible functions that could fit the data.

This prior is defined by a mean function and a covariance (kernel) function that quantifies the similarity between data points. As new data is observed, the model updates these beliefs using Bayesian inference to form a posterior distribution. This posterior distribution is a refined set of assumptions that accounts for the new data and is used to make predictions for new, unseen data points. A key advantage of this Bayesian framework is that it provides a measure of uncertainty for every prediction, indicating how confident the model is in its forecast.

Gaussian Process (GP)

A non-parametric, probabilistic model used for regression, classification, and quantifying uncertainty in predictions.

Mean Function

Represents the average or predicted value of the function at each input point.

Covariance (Kernel) Function

This is a crucial part of the model that measures the similarity between data points.

Prior Distributions

These represent the initial assumptions about the functions before any data is observed.

Posterior Distributions

These are the refined assumptions about the functions after a model has been trained on data.

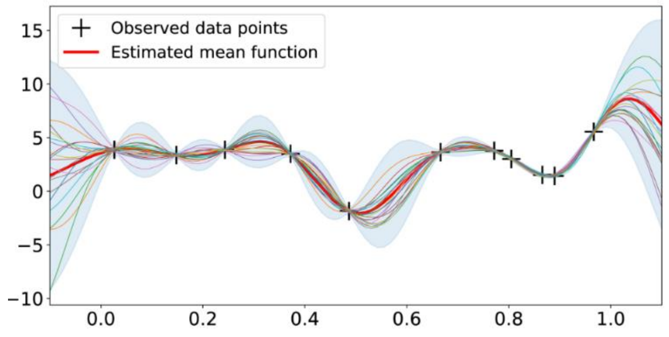


Figure 13: Possible results from a model trained trained by GPR

Limitations of GPR

There are two primary limitations with it:

1) The computational complexity is $O\ (N^3)$, where N represents the dimension of the covariance matrix K .

2) The memory consumption increases quadratically with data size. Due to these constraints, standard GPR models become impractical for large datasets. In such cases, sparse Gaussian Processes are employed to alleviate computational complexity.

SVR

Support Vector Regression (SVR) is a type of Support Vector Machine (SVM) algorithms and is commonly used for regression analysis. SVMs are powerful supervised learning algorithms that are primarily used for classification problems. Similar to SVMs, SVR uses the concept of a hyperplane and margin but there are differences in their definitions. In SVR, the margin is defined as the error tolerance of the model, which is also called as the ε-insensitive tube. This tube allows some deviation of the data points from the hyperplane without being counted as errors. The hyperplane is the best fit possible to the data that fall within the ϵ-insensitive tube. The difference of SVM and SVR is summarized in the figure below.

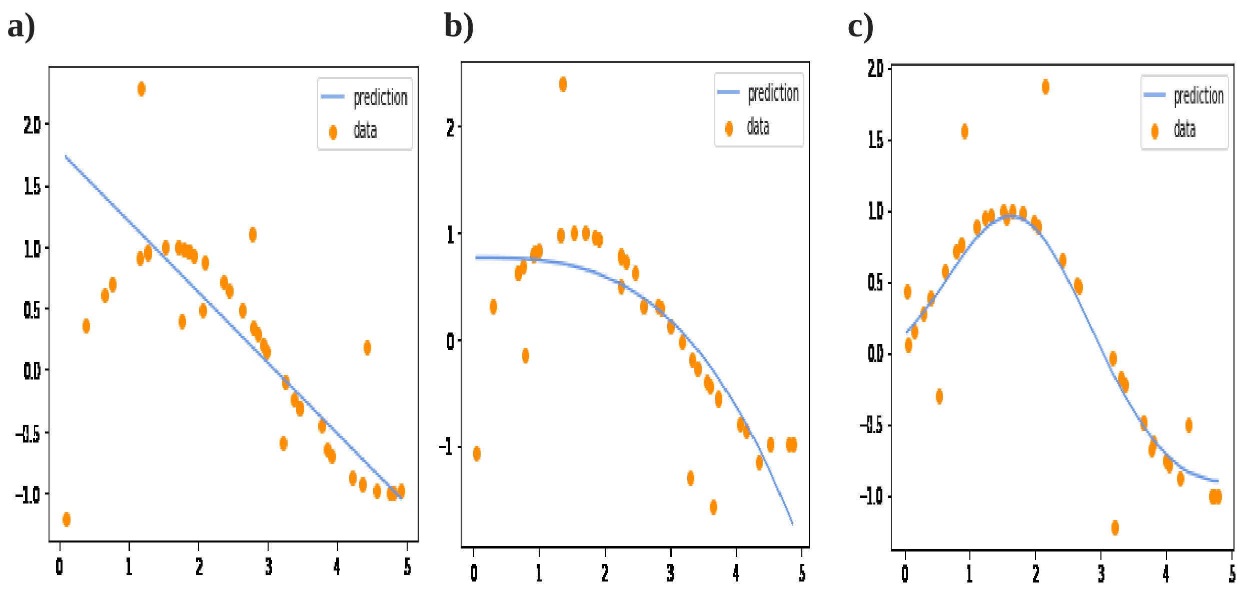


Figure 14: Implementing different kernels on a dataset : a)Linear b)Polynomial c)RBF

3.3 Deep learning approaches

MLP

multi-layer perceptron (MLP) is a type of artificial neural network consisting of multiple layers of neurons. The neurons in the MLP typically use nonlinear activation functions, allowing the network to learn complex patterns in data. MLPs are significant in machine learning because they can learn nonlinear relationships in data, making them powerful models for tasks such as classification, regression, and pattern recognition.

Structure

The structure consists of three layers:

- First Hidden Layer

This layer has 128 neurons. It uses a Leaky ReLU activation function with an alpha of 0.01 to prevent the "dying neuron" problem often seen with standard ReLU. A Dropout layer with a rate of 0.2 is applied after this layer for regularization, which helps prevent overfitting.

- Second Hidden Layer

This layer has 64 neurons and a linear activation function. A second Dropout layer with a rate of 0.2 is included for further regularization.

- Output Layer

This final layer has 2 neurons, corresponding to the two target variables, slip_x and slip_y. It uses a linear activation function, which is standard for regression problems.

NARX

Nonlinear autoregressive exogenous (NARX) models are a type of artificial neural network used for time-series prediction. They incorporate both autoregressive and exogenous inputs to forecast future values. NARX models are adept at capturing complex dependencies and nonlinear relationships within sequential data. They find applications in various fields such as finance, weather forecasting, and control systems. By leveraging historical data and external factors, NARX models offer improved accuracy in predicting future outcomes.

NARX stands for:

- Nonlinear

Nonlinear neural networks in deep learning connections within data can be captured via NARX networks. They can now simulate intricate dynamics and patterns that might not be linearly connected.

- Autoregressive

The term "autoregressive" describes the network's capacity to forecast values in the future by using its own historical outputs. NARX networks are able to anticipate future values by utilizing past forecasts as inputs and integrating feedback connections.

- Exogenous

External variables or factors that might affect the target variable under prediction are called exogenous inputs, and NARX networks are capable of integrating these. By enabling the network to take into account external data in addition to its internal dynamics, this feature improves its prediction power.

NARX Artificial Neural Network Structure

Input, hidden, and output layers are three interconnected layers of neurons that make up the architecture of NARX neural networks. NARX networks, in contrast to conventional feedforward neural networks, have feedback connections, which enable them to remember data from earlier time steps. Because of their recurrent nature, NARX networks are able to recognize temporal relationships and forecast outcomes based on past performance and outside variables.

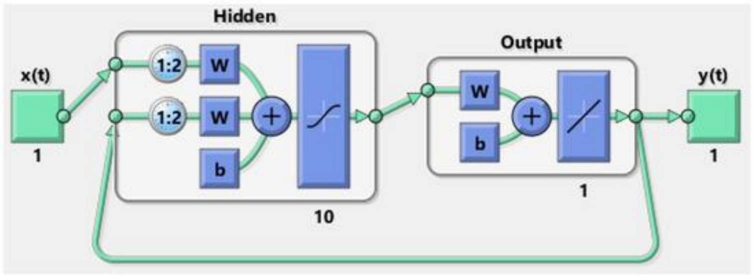


Figure 15:Schematic of NARX network

structure

The NARX functionality is implemented by creating a new dataset, narx_df, which includes lagged values of the target variables (slip_x and slip_y) alongside the original input features. This allows the model to leverage past outputs for future predictions, a key characteristic of autoregressive models. The two-layer LSTM, with 64 neurons in the first layer and 128 in the second, processes this time-series data to learn complex temporal dependencies. The model is compiled using the Adam optimizer.

3.3.3 LSTM

RNNs are incapable of learning long-term dependencies due to the presence of a single hidden state. However, LSTMs were able to address this problem by introducing a memory block to store information. This memory block allowed LSTM to store information for extended periods which allowed it to be a perfect solution for NLP tasks such as language translation.

The heart of an LSTM lies in the memory cell. The memory cell contains three different gates, namely, the input gate, the forget gate, and the output gate. These gates essentially decide what information needs to be added, removed, and passed onto the next state.

- Forget gate

This gate allows LSTM to learn long-term dependencies. This gate allows the model to remove all the unnecessary information from the previous hidden state, thereby allowing it to learn the long-term dependency.

- Input gate

The input gate is responsible for the addition of useful information into the current cell state. This gate filters down the values coming from the previous state and then passes onto the filtered value.

- Output gate

The output gate is responsible for extracting useful information from the current cell state. After extracting the necessary information from the current cell state the output is added to the output of both forget and input gate and passed onto the next cell.

This change in architecture enables the LSTM to store long-term dependencies and enables it to get rid of the vanishing and exploding gradients issue of RNNs. It enables LSTMs to capture the important contexts even if there is a significant gap between the events.

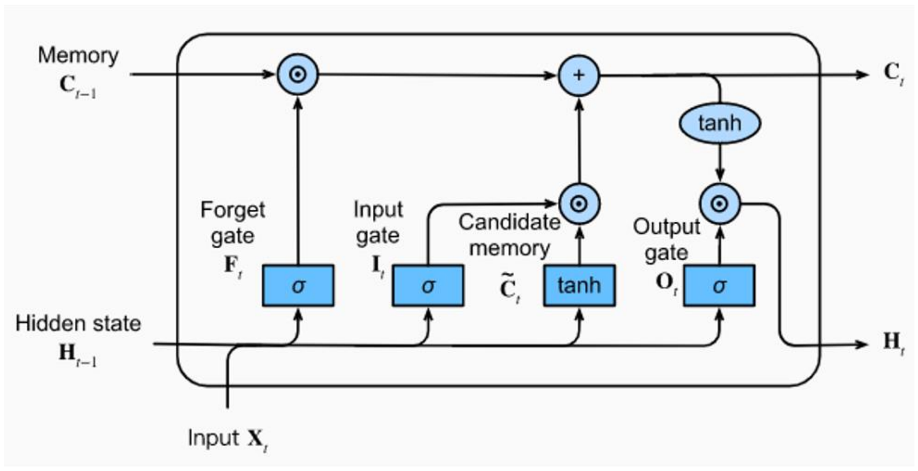

Memory $\mathbf{C}_{t-1}$
$\mathbf{C}_t$
tanh
Forget gate $\mathbf{F}_t$
Input gate $\mathbf{I}_t$
Candidate memory $\tilde{\mathbf{C}}_t$
Output gate $\mathbf{O}_t$
σ
tanh
Hidden state $\mathbf{H}_{t-1}$
$\mathbf{H}_t$
Input $\mathbf{X}_t$

Figure 16: Schematic of LSTM network

Architecture

The first LSTM layer has 64 neurons and is configured with return_sequences=True, which is critical as it passes the output sequence of the first layer to the second. This layer is followed by a Leaky ReLU activation function. The second LSTM layer has 128 neurons and a linear activation function, which processes the output from the first layer to capture more complex temporal patterns. Finally, a Dense layer with 2 neurons serves as the output, predicting the slip_x and slip_y values.

CNN-LSTM

CNNs and LSTMs are both widely used in the field of time series analysis. CNNs are powerful for learning local patterns in data, while LSTMs are effective at capturing long-term dependencies in sequential data. Combining these two types of networks can result in improved performance for time series classification tasks.

Main ways to combine a convolutional neural network (CNN) and a long short-term memory (LSTM) network:

- Use the output of the CNN as the input to the LSTM. This allows the LSTM to learn features from the input data that have been learned by the CNN.
- Use the output of the LSTM as the input to the CNN. This allows the CNN to learn features from the output of the LSTM.
- Use a parallel architecture, where the CNN and LSTM operate on the input data independently, and their outputs are concatenated and passed to the fully connected layer.

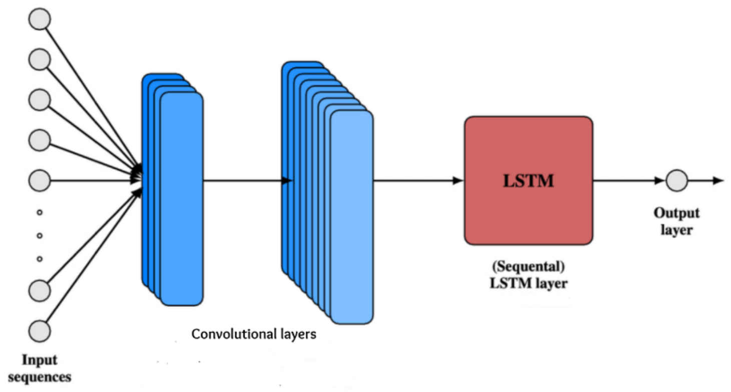


Figure 17: CNN-LSTM network

Architecture

- 1D Convolutional Layer (Conv1D)

This is the first layer of the model. It uses 64 filters with a kernel_size of 3 and a relu activation function. The Conv1D layer is crucial for this architecture; it acts as a feature extractor, sliding a small window (the kernel) over the input time-series data to identify and learn local patterns or features.

- 1D Max Pooling Layer (MaxPooling1D)

After the convolutional layer, a MaxPooling1D layer with a pool_size of 2 is used. This layer down-samples the feature maps, reducing the number of parameters and helping to make the model more robust to minor shifts or translations in the input sequence.

- LSTM Layer

This is the core of the recurrent part of the model. The output from the MaxPooling1D layer is fed into this LSTM layer, which has 50 neurons and uses a relu activation function. The LSTM's role is to learn the temporal dependencies and long-term patterns within the sequence of features that the CNN has already extracted.

- Dense Output Layer

The final Dense layer has 1 neuron. This layer takes the output of the LSTM and produces the final prediction. In a regression task, this single neuron would output the predicted value.

CNN LATM with autoencoder

This model is an enhanced version of the CNN-LSTM. It is used to further improve estimation accuracy by incorporating the benefits of an auto encoder architecture.

- Compressed Representation

Auto encoders consist of an encoder and a decoder. The encoder learns a compressed representation of the input data, and the decoder attempts to reconstruct the original input from this representation.

- Learning Key Information

The CNN-LSTM-AE model learns to minimize the difference between its output and the input data during training. This process forces the model to focus on and learn the most critical information within the data, which can lead to improved performance in the slip estimation task.

- Architecture

This model replaces the standard neural network layers in an auto encoder with CNN and LSTM layers, which are better suited for the time-series data related to slip estimation. The CNN-LSTM-AE model uses the advantages of both the CNN and auto encoder algorithms to increase the performance of the slip estimation.

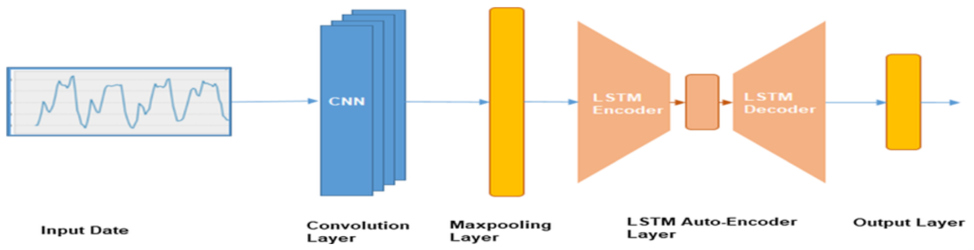


Figure 18: The architecture of the proposed hybrid model CNN-LSTM autoencoder model

Architecture

The Encoder is the first part of the network, responsible for compressing the input data. It begins with two 1D Convolutional (Conv1D) layers that act as feature detectors, identifying local patterns and features within the time series data. These layers are followed by Dropout layers, which randomly disable a percentage of neurons during training to prevent the model from memorizing the data and overfitting. The output of the convolutional layers is then fed into a Long Short-Term Memory (LSTM) layer. The LSTM's role is to learn the temporal dependencies and long-range patterns in the sequence. It processes the features extracted by the CNN and compresses them into a single, dense vector, which is the learned representation of the input.

The Decoder takes the compressed representation from the encoder and attempts to reconstruct the original input data. This is the core of the autoencoder's functionality. The compressed vector is first expanded back into a sequence using a RepeatVector layer. This repeated sequence is then processed by another LSTM layer, which learns how to reconstruct the original temporal patterns. Finally, a TimeDistributed layer with a Dense layer applies a linear transformation to each step of the reconstructed sequence, producing an output that has the same shape as the original input. The goal during training is to minimize the difference between the reconstructed output and the original input. This process forces the network to learn a highly efficient and meaningful encoding of the data. The final prediction layer in your code is then a Dense layer on the output of the last LSTM layer, which is used for the downstream task of predicting slip values.

3.4 Evaluation metrics

3.4.1 MAE

The Mean absolute error represents the average of the absolute difference between the

actual and predicted values in the dataset. It measures the average of the residuals in the dataset.
The unit of MAE is the same as the unit of the target variable. It represents the average absolute difference between the actual and predicted values.

Key features of MAE include:

- Interpretability

MAE is easy to understand as it's in the same units as the target variable.

- Robustness to Outliers

Unlike MSE/RMSE, MAE is less sensitive to outliers.

- Linear Scale

MAE treats all errors on a linear scale, unlike MSE which squares the errors.

- Always Non-Negative

Like MSE, MAE is always ≥ 0, with 0 indicating perfect prediction.

3.4.2 MSE

Mean Squared Error represents the average of the squared difference between the original and predicted values in the data set. It measures the variance of the residuals.

The unit of MSE is the square of the unit of the target variable. It measures the average squared difference between the actual and predicted values.

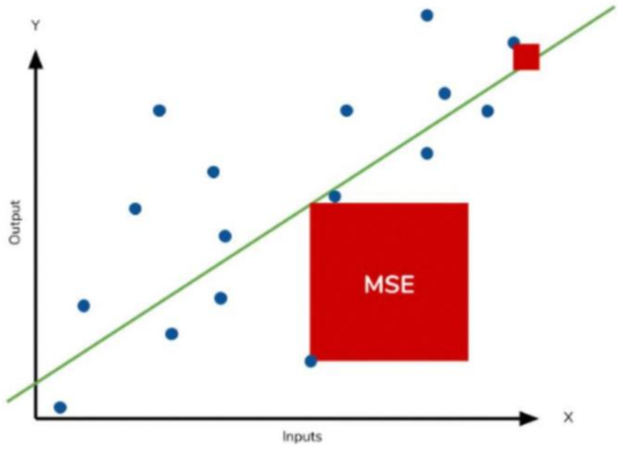


Figure 19: Visualizing MSE on different data points

Some key characteristics of MSE include:

- Always Non-Negative

MSE is always ≥ 0, with 0 indicating perfect prediction.

- Sensitive to Outliers

Squaring the errors makes MSE particularly sensitive to large errors.

- Units

MSE is expressed in squared units of the target variable.

MSE heavily penalizes large errors due to the squaring operation, making it particularly useful when large errors are especially undesirable

3.4.3 RMSE

Root Mean Squared Error is the square root of Mean Squared error. It measures the standard deviation of residuals.

The unit of RMSE is the same as the unit of the target variable. It is the square root of the MSE and represents the standard deviation of the residuals.

RMSE shares many properties with MSE but has some distinct advantages:

- Interpretability

RMSE is in the same units as the target variable, making it more interpretable than MSE.

- Still Sensitive to Outliers

Like MSE, RMSE is sensitive to large errors.

- Popular in Practice

RMSE is widely used and reported in many regression problems.

3.4.4 R^2 (R-squared)

R^2 is a dimensionless value, representing the proportion of the variance in the dependent variable that is predictable from the independent variables. It ranges from 0 to 1 or can be negative in some cases.

The key advantages of R-squared include:

- Interpretability

It provides an easily understandable measure of fit.

- Comparability

It allows for comparison between models with different numbers of predictors.

- Scale independence

Unlike MSE or MAE, R-squared is not affected by the scale of the target variable. While a higher R-squared generally indicates a better fit, be cautious of overfitting when R-squared is very close to 1, especially with a large number of predictors.

However, R-squared has several limitations:

- Sensitivity to outliers

R-squared can be sensitive to outliers, as they can significantly impact the total sum of squares.

- No indication of bias

A high R-squared doesn't necessarily mean the model is unbiased. Always check residual plots.

- Doesn't indicate prediction accuracy

R-squared measures goodness of fit, not predictive accuracy. Note that R-squared compares the model to a baseline (the mean of the target variable). MSE/RMSE don't have this implicit comparison

- Increases with more predictors

Adding more predictors to a model will always increase R-squared, even if these predictors are not meaningful.

- Not suitable for nonlinear relationships

R-squared assumes a linear relationship between variables.

3.4.5 Residual plots

A residual plot is a scatter plot used to assess the fit of a regression model by displaying the differences between observed and predicted values (residuals) against the independent variable (or predicted values). Analyzing residual plots helps determine if the model assumptions, like linearity and constant variance, are met, and if the model adequately captures the relationship between variables.

Residuals represent the portion of the dependent variable that the model fails to explain.

Residual analysis helps in:

- Validating model assumptions
- Identifying patterns the model missed
- Detecting outliers and influential points
- Assessing the appropriateness of the linear model

Positive values for the residual (on the y-axis) mean the prediction was too low, and negative values mean the prediction was too high; 0 means the guess was exactly correct.

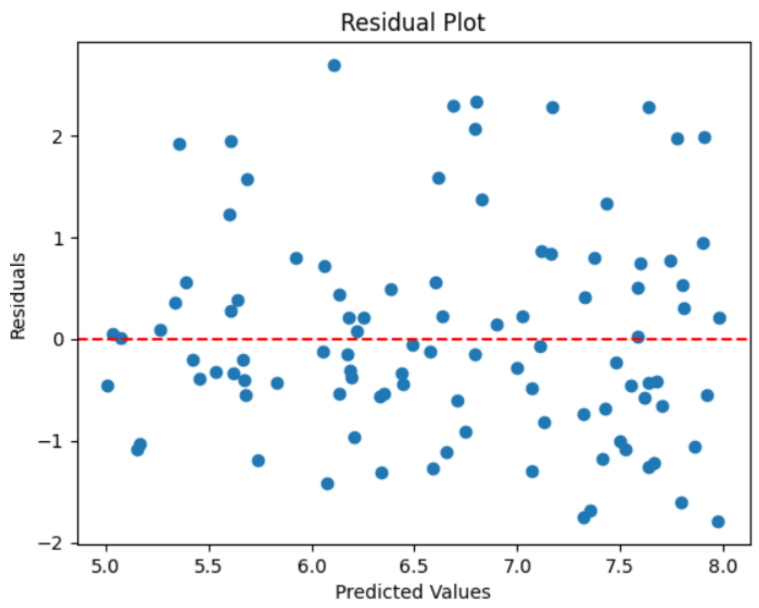


Figure 20: Example of residual plots

What to Look For?

- Random scatter around the horizontal line at y=0
- No obvious patterns or trends
- Consistent spread across predicted values

3.4.6 Q-Q Plot

The QQ plot, or quantile-quantile plot, is a graphical tool to help us assess if a set of data plausibly came from some theoretical distribution such as a normal or exponential. For example, if we run a statistical analysis that assumes our residuals are normally distributed, we can use a normal QQ plot to check that assumption. It’s just a visual check, not an air-tight proof, so it is somewhat subjective. But it allows us to see at-a-glance if our assumption is plausible, and if not, how the assumption is violated and what data points contribute to the violation.

A QQ plot is a scatterplot created by plotting two sets of quantiles against one another. If both sets of quantiles came from the same distribution, we should see the points forming a line that’s roughly straight. Here’s an example of a normal QQ plot when both sets of quantiles truly come from normal distributions.

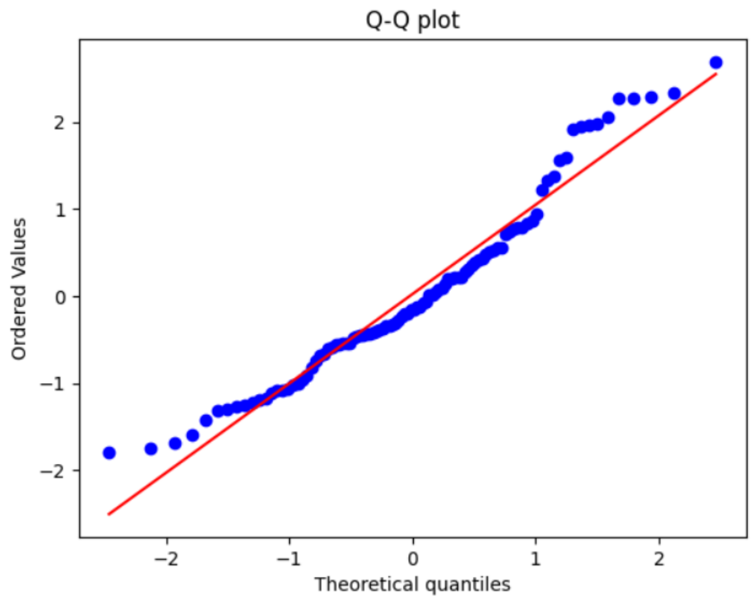


Figure 21: Example of Q-Q plot

What to Look For:

- Points should roughly follow a straight line
- Significant deviations suggest non-normality

Chapter 4

Results

Square path

Table 2: Square path evaluation

Model	Target	R-squared	MSE	MAE	RMSE
Linear Regression	Slip X	0.8471	0	0.005	0.005
	Slip Y	0.8588	0.0001	0.0067	0.01
Polynomial Regression	Slip X	0.43	0.0002	0.0065	0.0141
	Slip Y	0.5477	0.0003	0.0082	0.0173
Gaussian Process Regression (GPR)	Slip X	0.9335	0	0.0027	0.0028
	Slip Y	0.9209	0	0.0038	0.0039
SVR (Linear)	Slip X	-0.06	0.0003	0.0151	0.0176
	Slip Y	-0.1781	0.0007	0.024	0.0259
SVR (Poly)	Slip X	-0.06	0.0003	0.0151	0.01176
	Slip Y	-0.1781	0.0007	0.0240	0.0259
SVR (RBF)	Slip X	-0.06	0.0003	0.0151	0.01176
	Slip Y	-0.1781	0.0007	0.0240	0.0259
Tuned SVR (Polynomial Kernel)	Slip X	0.8131	0.0001	0.0065	0.01
	Slip Y	0.8739	0.0001	0.0076	0.01
MLP (Two Hidden Layers)	Slip X	0.9634	0	0.0026	0.0026
	Slip Y	0.9419	0	0.0035	0.0037
LSTM	Slip X	0.9873	0	0.0012	0.0019
	Slip Y	0.9952	0	0.0011	0.0017
NARX-LSTM (Two-Layer)	Slip X	0.9664	0	0.0023	0.0024

	Slip Y	0.9576	0	0.0032	0.0033
CNN-LSTM	Slip X	0.9676	0	0.0022	0.0023
	Slip Y	0.9686	0	0.0027	0.0028

Linear Regression

This model performs reasonably well, with R-squared values of 0.8471 for Slip X and 0.8588 for Slip Y. The error metrics are also low, with MAE values of 0.005 and 0.0067, respectively.

Polynomial Regression

This model performs worse than Linear Regression, with lower R-squared values (0.43 for Slip X and 0.5477 for Slip Y) and higher error values.

Gaussian Process Regression (GPR)

GPR is the top performer among the machine learning models. Its R-squared values are very high (0.9335 for Slip X and 0.9209 for Slip Y), which are close to the performance of the deep learning models. Its error metrics are also very low.

Support Vector Regression (SVR)

The untuned SVR models (Linear, Poly, and RBF) perform exceptionally poorly, with negative R-squared values for both Slip X and Slip Y. This indicates that these models are worse than a simple horizontal line (the mean of the target variable).

Tuned SVR with a Polynomial Kernel

Shows a significant improvement, with R-squared values of 0.8131 for Slip X and 0.8739 for Slip Y, and much lower error metrics.

Deep Learning Models

All deep learning models show excellent performance.

MLP

This model has very high R-squared values (0.9634 for Slip X and 0.9419 for Slip Y) and very low error values.

NARX-LSTM

This model shows a slight improvement over the MLP, with R-squared values of 0.9664 and 0.9576 for Slip X and Slip Y, respectively.

CNN-LSTM

This model is the top performer for Slip X with an R-squared of 0.9676. It also performs very well for Slip Y with an R-squared of 0.9686, which is the highest R-squared value in the entire table.

Conclusion

CNN-LSTM and NARX-LSTM models are the most effective for slip estimation, as they have the highest R-squared values and the lowest error metrics. Among the machine learning models, GPR is the clear leader, with performance metrics comparable to some of the deep learning models. The results highlight the superiority of deep learning approaches for this specific problem, especially when compared to simpler machine learning methods like untuned SVR.

Q-Q plots

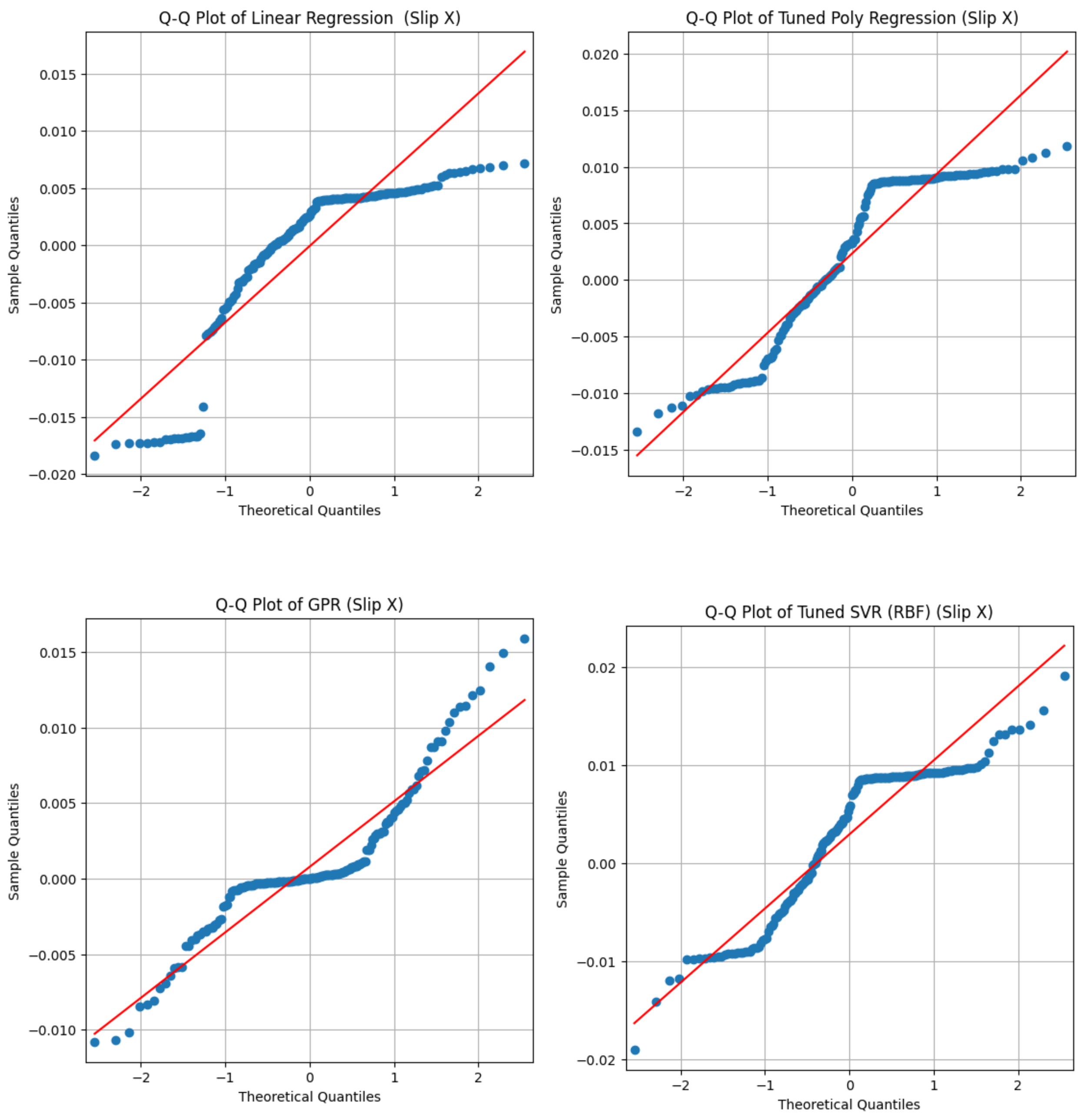


Figure 22: Q-Q plots of machine learning models for slip x

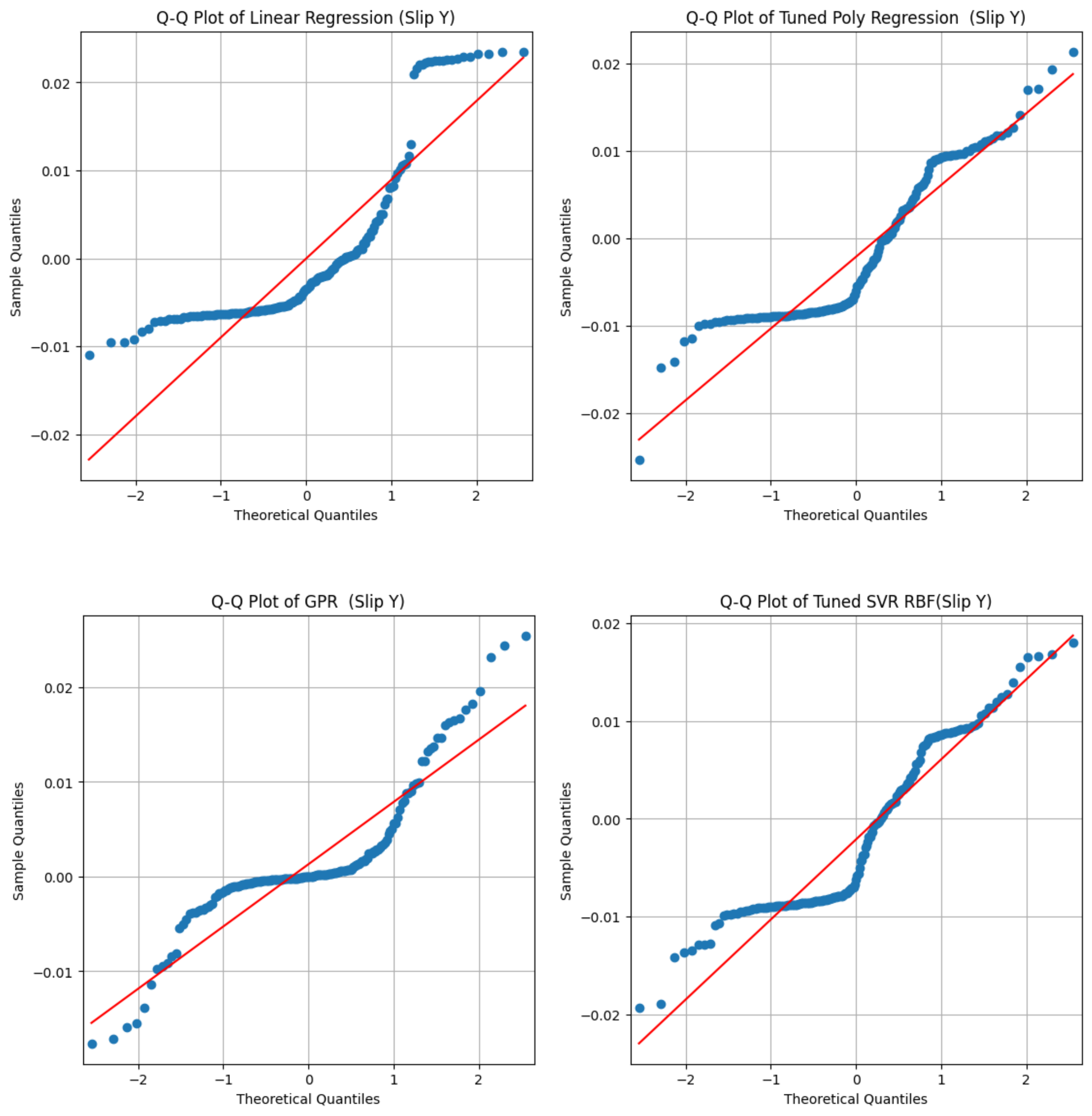


Figure 23: Q-Q plots of machine learning models for slip y

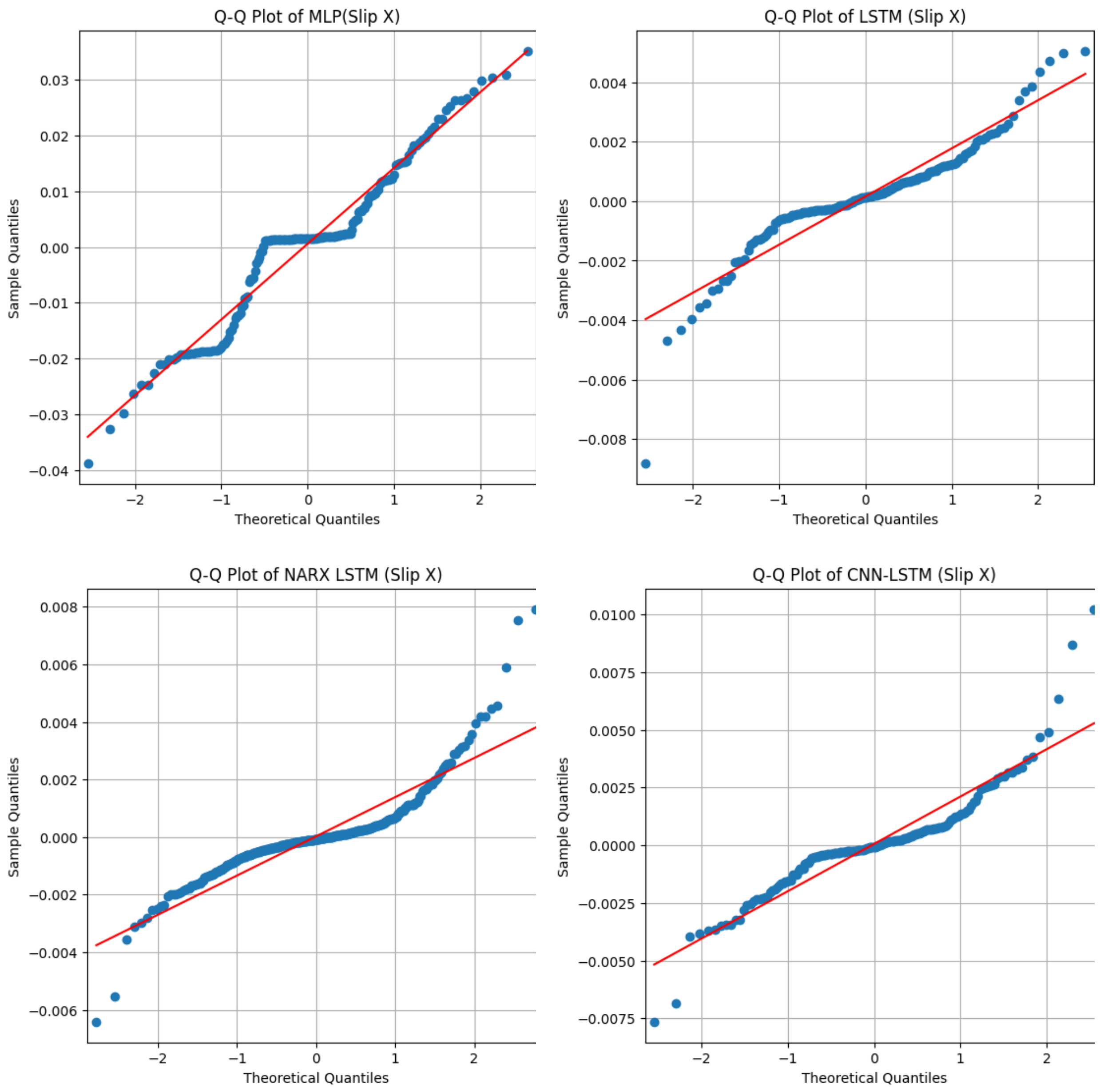


Figure 24: Q-Q plots of deep learning models for slip x

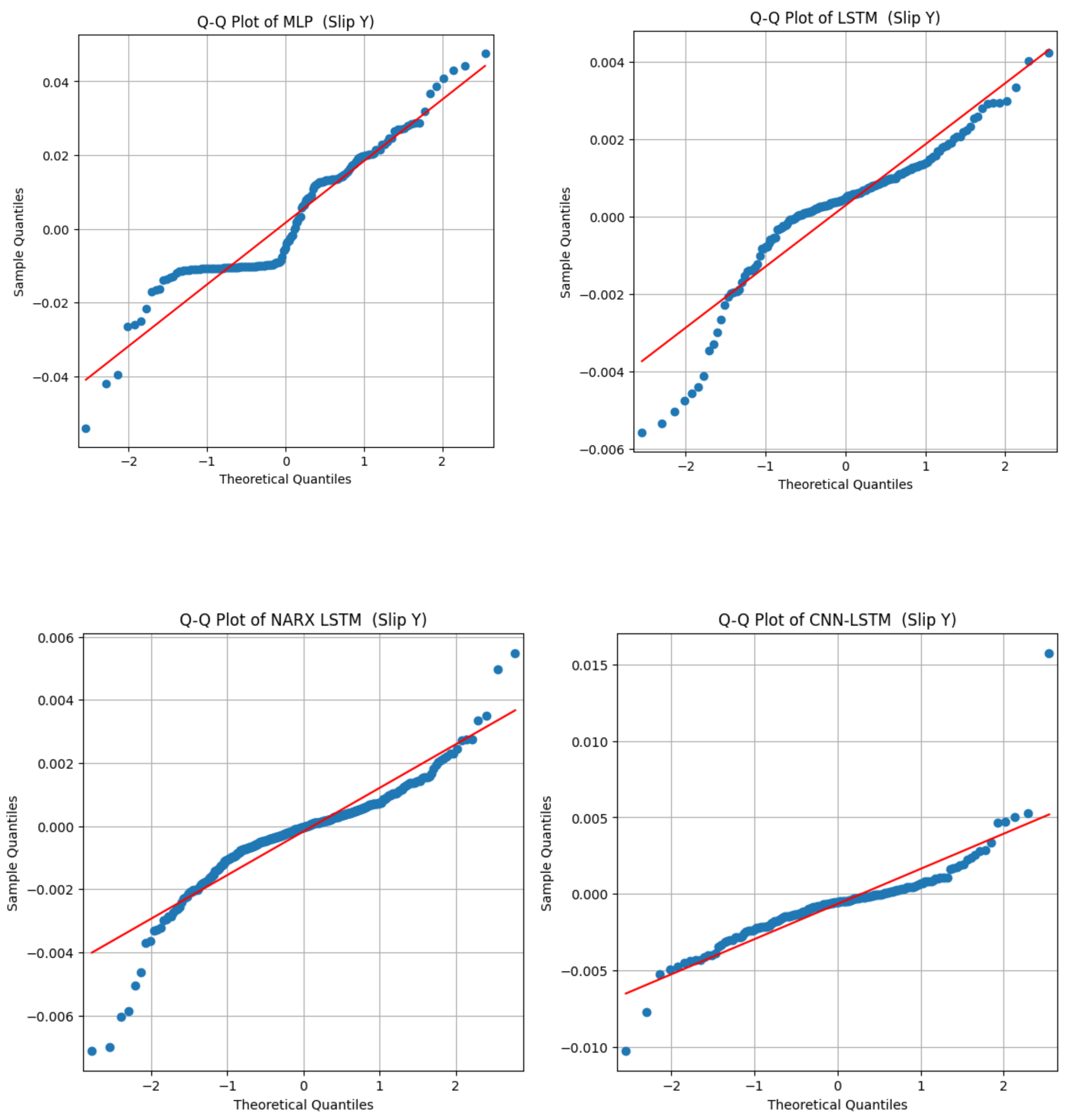


Figure 25: Q-Q plots of deep learning models for slip y

Results

Linear Regression

The plot for Linear Regression (Slip X) shows significant deviations from the straight line, particularly at the tails. The "S" shape indicates that the residuals have heavier tails than a normal distribution, meaning there are more extreme errors than expected.

Tuned Poly Regression

The plot for Tuned Poly Regression (Slip X) also shows a distinct "S" shape, similar to the linear regression plot but with slightly less severe deviations at the tails. This suggests that while it may perform better than linear regression, its residuals still are not normally distributed.

GPR

The Gaussian Process Regression (GPR) plot for Slip X shows the best alignment with the straight line among the models shown. While there is some minor deviation at the tails, the overall fit is much closer to a normal distribution, suggesting this model's residuals are more consistent and predictable.

Tuned SVR (RBF)

The Tuned SVR (RBF) plot for Slip X also shows a relatively good fit to the straight line, similar to GPR. The points generally follow the diagonal line, indicating that the residuals are reasonably normally distributed.

MLP

The plot for the MLP model (Slip X) shows a significant "S" curve, with points at both ends deviating sharply from the line. This indicates that its residuals have very heavy tails, and the model produces a higher number of large-magnitude errors.

NARX LSTM

The NARX LSTM plot for Slip X shows a reasonably good fit to the line in the center but with noticeable deviations at the tails. The tails are slightly less heavy than those of the MLP, suggesting a better performance but still not a perfect normal distribution of errors.

CNN-LSTM

The CNN-LSTM plot for Slip X shows a similar pattern to the NARX LSTM plot, with a relatively good fit in the middle and some deviations at the tails. This indicates that the residuals are not perfectly normally distributed, but the model is generally well-behaved.

Residual plots

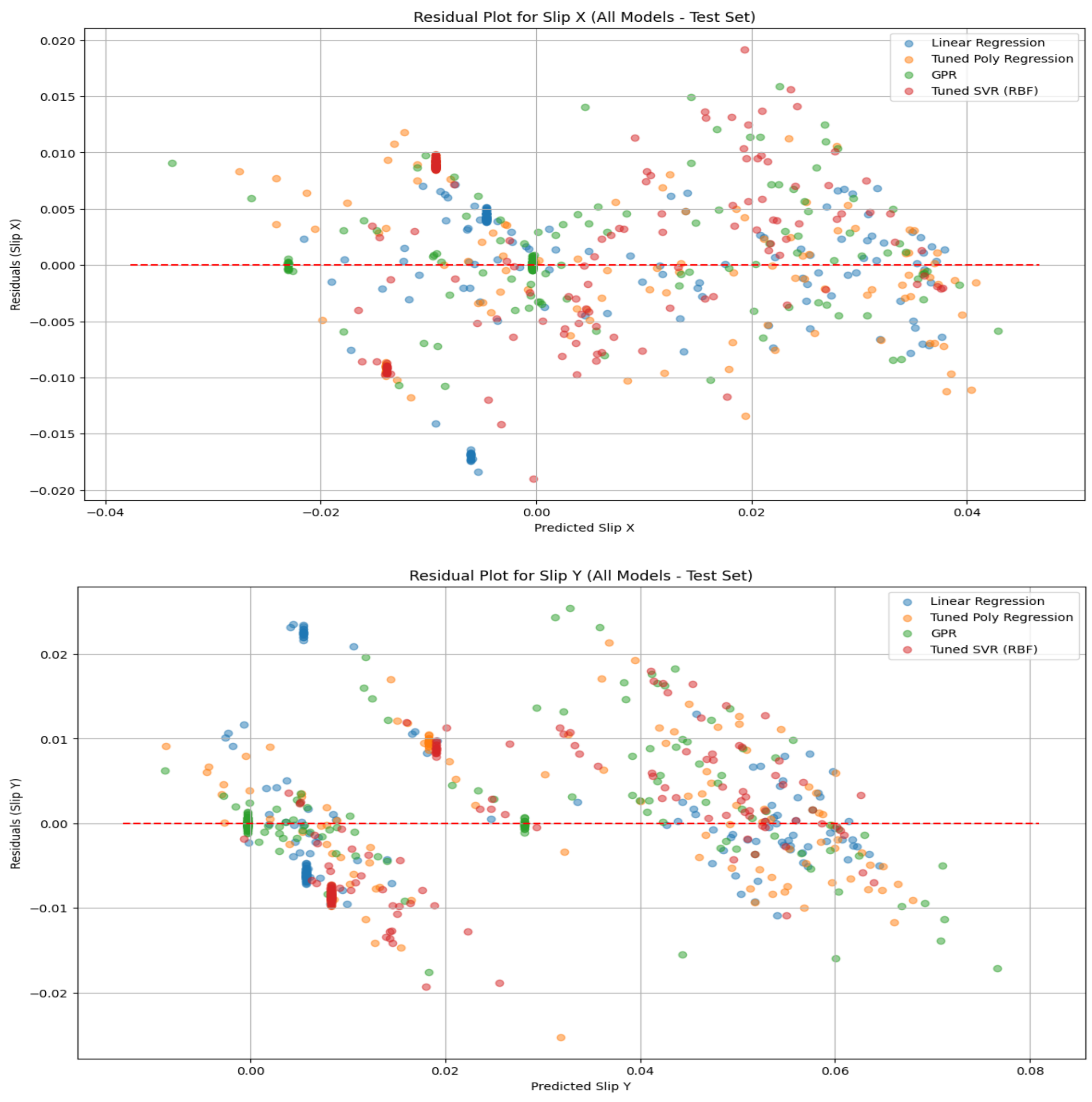


Figure 26: Residuals of machine learning models

conclusion

Residual Plot for Slip X (ML Models)

Linear Regression and Tuned Poly Regression both show a slight "fanning out" pattern, where the residuals tend to be more spread out for larger and smaller predicted slip values. This suggests that the error variance is not constant across all predictions, a condition known as heteroscedasticity.

GPR and Tuned SVR (RBF) show a much more uniform and random spread of residuals around the zero line. The points are consistently scattered without any obvious trends or changes in variance. This indicates that these models are better at capturing the underlying patterns in the data and produce more consistent and reliable errors.

Residual Plot for Slip Y (ML Models)

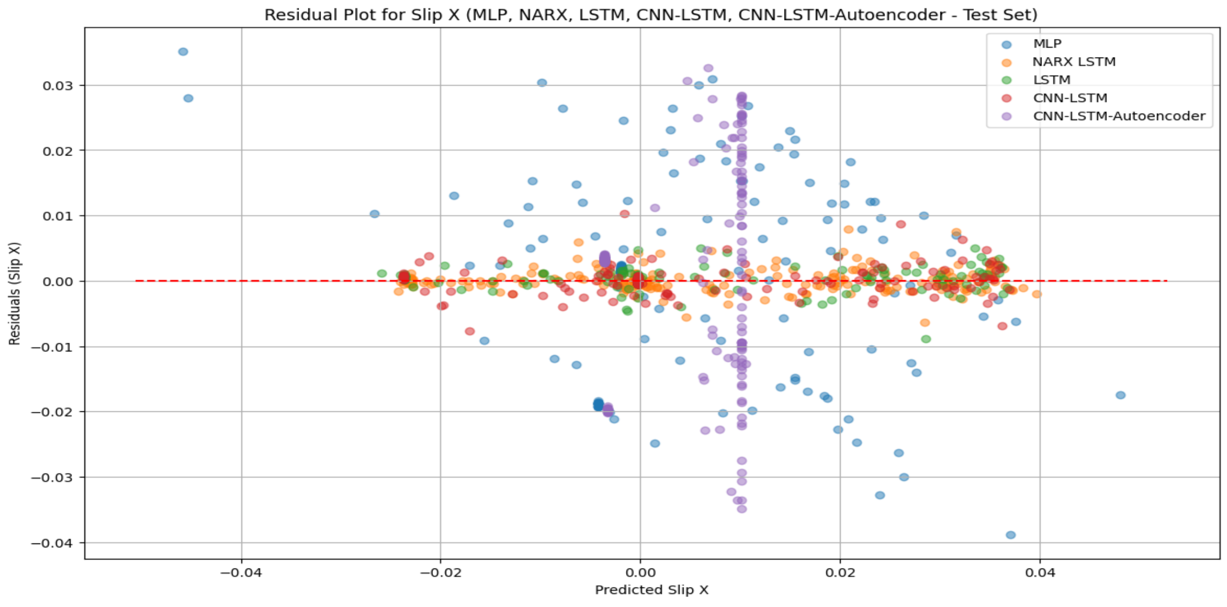
Linear Regression and Tuned Poly Regression both show a clear pattern of "fanning out" from the center. This indicates that the models have more trouble with accuracy as the predicted slip value gets larger, and the residuals are not uniformly distributed.

GPR and Tuned SVR (RBF) again demonstrate a more desirable residual distribution. The residuals for these models are much more tightly clustered around the zero line and show less of the "fanning out" effect seen in the other models. This suggests that GPR and Tuned SVR (RBF) are more robust and consistent in their predictions for Slip Y.

Overall Comparison

Based on these residual plots, Gaussian Process Regression (GPR) and Tuned SVR (RBF) consistently outperform Linear and Tuned Polynomial Regression for both Slip X and Slip Y. The residuals for GPR and SVR are more randomly distributed and show a more constant variance, which is a key indicator of a good model. Conversely, the clear "fanning out" patterns in the residual plots for Linear and Polynomial Regression suggest that these models are less suitable for this particular dataset, as their predictive accuracy is not consistent across all ranges of slip values.

A good residual plot should have residuals that "bounce randomly" around the 0 line in a horizontal band, which suggests the assumption of a linear relationship is reasonable and that the error terms have equal variance. The plots for GPR and SVR more closely adhere to this ideal. The presence of non-random patterns, like the "fanning out" seen in the other models, indicates that the models are missing something and could be improved.



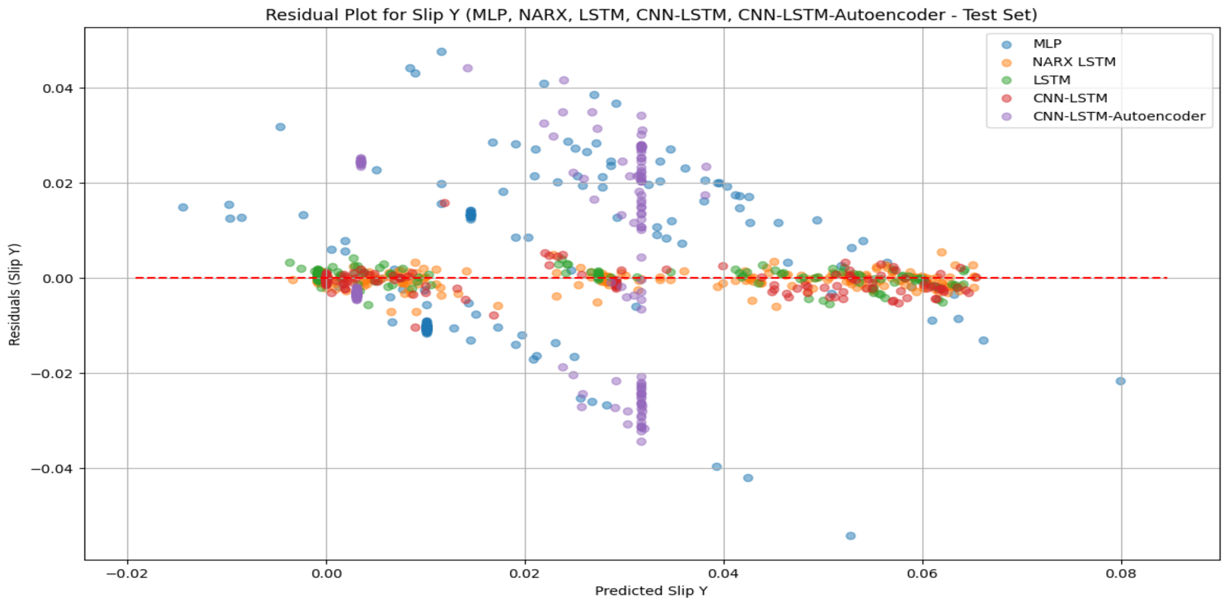


Figure 27: Residuals of deep learning models

Residual Plot for Slip X (Deep Learning Models)

MLP shows a very wide spread of residuals, especially around the predicted Slip X value of 0.01. The residuals are not randomly scattered, with a heavy concentration of both positive and negative outliers. This indicates that the MLP model's predictions are inconsistent and less reliable than the other deep models.

NARX LSTM and LSTM both show a more concentrated spread of residuals around the zero line compared to the MLP. The residuals are tightly clustered in a horizontal band, indicating a more consistent performance. There are still some outliers, but they are less severe than those produced by the MLP.

CNN-LSTM shows a very uniform and tight cluster of residuals around the zero line, indicating a highly consistent and accurate model. The residuals are very small and show a random scatter, which is the ideal outcome for a residual plot.

CNN-LSTM-Autoencoder shows a pattern similar to the CNN-LSTM, with residuals tightly clustered around zero. The model's errors are consistently small, suggesting a reliable and accurate performance.

Residual Plot for Slip Y

MLP again shows a wide dispersion of residuals. There is a "fanning out" pattern, where the residuals get more spread out for both smaller and larger predicted slip values. This indicates that the MLP model's performance is not consistent and the variance of its errors changes depending on the prediction.

NARX LSTM and LSTM show a tighter cluster of residuals compared to the MLP, but still exhibit some noticeable outliers. The spread is not perfectly uniform, but it is a significant improvement over the MLP model.

CNN-LSTM demonstrates an excellent residual distribution, with a very tight and random scatter of points around the zero line. The errors are consistently small, indicating a high degree of accuracy and reliability for this model.

CNN-LSTM-Autoencoder also shows a very good residual plot, with residuals tightly bound to the zero line. The distribution is random and shows little change in variance, which is a sign of a high-performing model.

Circular path

Table 3 :Circular path evaluation

Model	Target	R-squared	MSE	MAE	RMSE
Linear Regression	Slip X	0.8471	0	0.005	0.005
	Slip Y	0.8588	0.0001	0.0067	0.01
Polynomial Regression	Slip X	0.43	0.0002	0.0065	0.0141
	Slip Y	0.5477	0.0003	0.0082	0.0173
Gaussian Process Regression (GPR)	Slip X	0.9335	0	0.0027	0.0028
	Slip Y	0.9209	0	0.0038	0.0039
SVR (Linear)	Slip X	-0.06	0.0003	0.0151	0.0176
	Slip Y	-0.1781	0.0007	0.024	0.0259
SVR (Poly)	Slip X	0.8131	0.0001	0.0065	0.01
	Slip Y	0.8739	0.0001	0.0076	0.01
SVR (RBF)	Slip X	0.9634	0	0.0026	0.0026
	Slip Y	0.9419	0	0.0035	0.0037
Tuned SVR (Polynomial Kernel)	Slip X	0.8471	0	0.005	0.005
	Slip Y	0.8588	0.0001	0.0067	0.01
MLP (Two Hidden Layers)	Slip X	0.43	0.0002	0.0065	0.0141
	Slip Y	0.5477	0.0003	0.0082	0.0173
LSTM	Slip X	0.9491	0	0.0012	0.0021
	Slip Y	0.9579	0	0.0017	0.0024
NARX-LSTM (Two-Layer)	Slip X	0.9664	0	0.0023	0.0024

	Slip Y	0.9576	0	0.0032	0.0033
CNN-LSTM	Slip X	0.9676	0	0.0022	0.0023
	Slip Y	0.9686	0	0.0027	0.0028

Linear Regression

This model performs reasonably well but is outperformed by GPR and all deep learning models. It has R-squared values of 0.8471 (Slip X) and 0.8588 (Slip Y).

Polynomial Regression

This model performs worse than Linear Regression, with lower R-squared values of 0.43 and 0.5477, and higher error values.

Gaussian Process Regression (GPR)

This is the best-performing machine learning model. Its R-squared values (0.9335 for Slip X and 0.9209 for Slip Y) are very high and close to the performance of some deep learning models. Its low MAE and RMSE values also highlight its accuracy.

Support Vector Regression (SVR)

The untuned SVR models perform exceptionally poorly, with negative R-squared values for both Slip X (-0.06) and Slip Y (-0.1781). This suggests that they are worse at predicting slip than simply using the mean of the target variable.

Tuned SVR with a Polynomial Kernel

After tuning, this model shows a significant improvement over its untuned counterparts, with R-squared values of 0.8131 (Slip X) and 0.8739 (Slip Y).

Deep Learning Models
All deep learning models demonstrated excellent performance, outperforming the traditional machine learning models.

MLP

The MLP model also performs well, with R-squared values of 0.9634 (Slip X) and 0.9419 (Slip Y). However, its error metrics (MAE and RMSE) are slightly higher compared to the CNN-LSTM and NARX-LSTM models, indicating it's less accurate overall.

NARX-LSTM

This model is a strong contender, showing very high R-squared values for Slip X (0.9664) and Slip Y (0.9576). Its MAE and RMSE are slightly higher than CNN-LSTM but still very low.

CNN-LSTM

This model is the top performer across the board. It has the highest R-squared values for both Slip X (0.9676) and Slip Y (0.9686). It also has the lowest MAE and RMSE values for both targets, indicating the smallest average errors.

Overall conclusion

The CNN-LSTM model and the LSTM are the clear top performers, consistently achieving the highest R-squared values and the lowest MAE and RMSE scores. A high R-squared value indicates that the model explains a significant portion of the variance in the data, while low MAE and RMSE values signify small prediction errors on average. The high performance of these models suggests that their architecture, which combines feature extraction from CNNs with the temporal pattern recognition of LSTMs, is particularly well-suited for this problem.

In comparison, traditional machine learning models generally have higher error rates and lower R-squared scores. The GPR model stands out as the best among them, with results that approach the performance of the deep learning models. The untuned SVR models perform exceptionally poorly, with negative R-squared values, which means they are worse than simply predicting the average value of the data. This highlights that model selection and proper tuning are critical for achieving good results in machine learning.

Q-Q plots

Machine learning plots

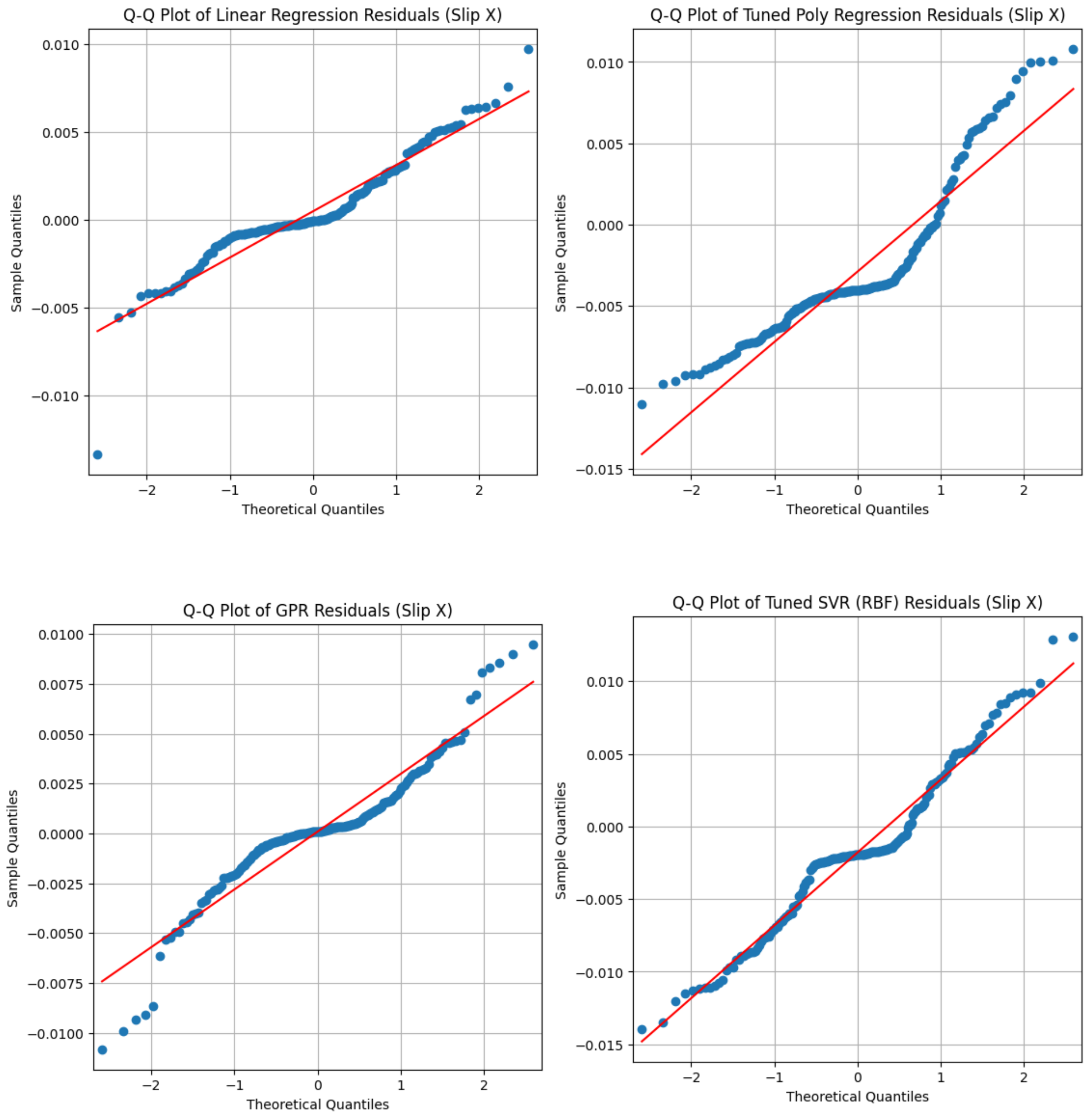


Figure 28: Q-Q plots of machine learning models for slip x

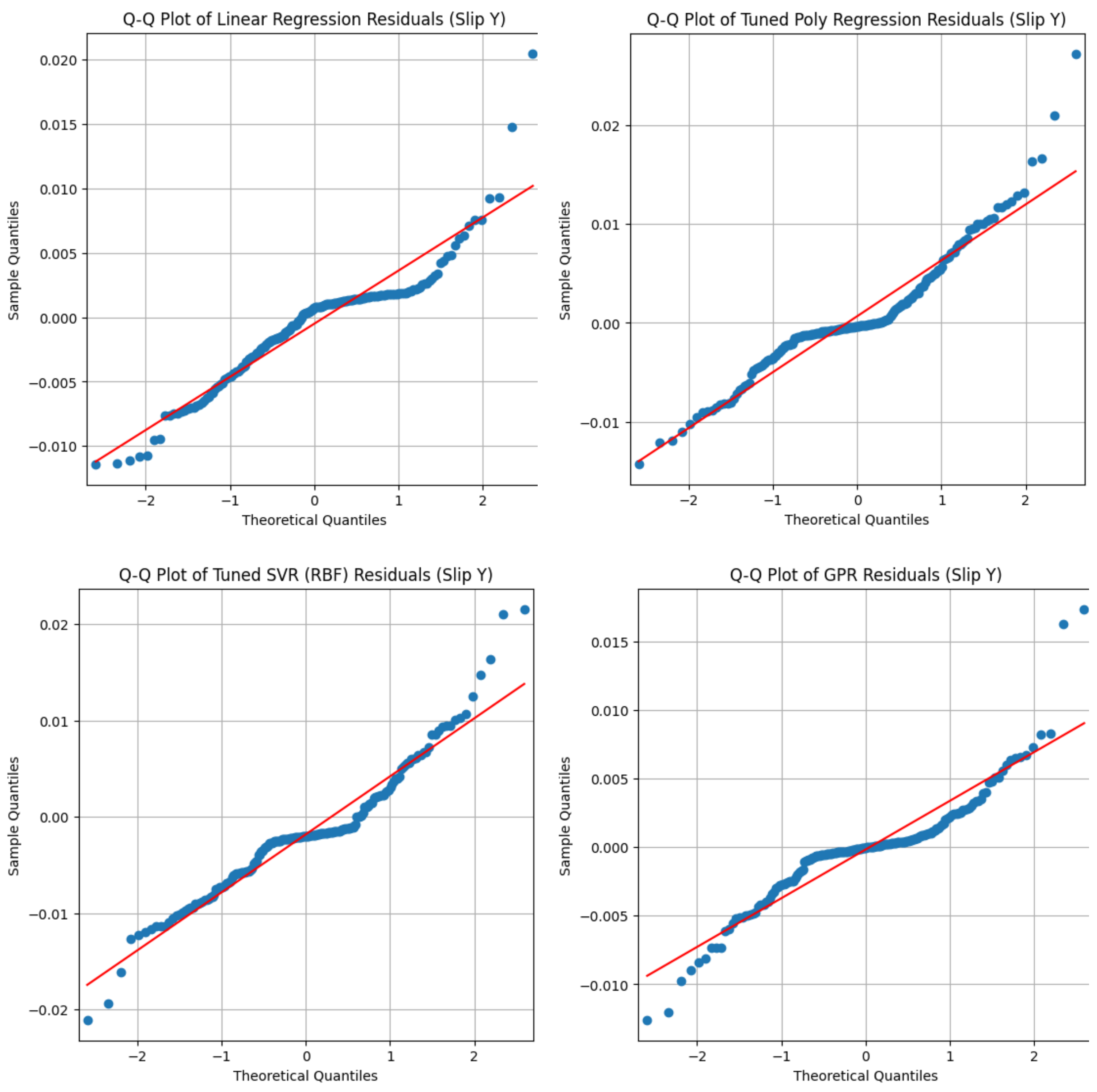


Figure 29: Q-Q plots of machine learning models for slip x

Deep learning models

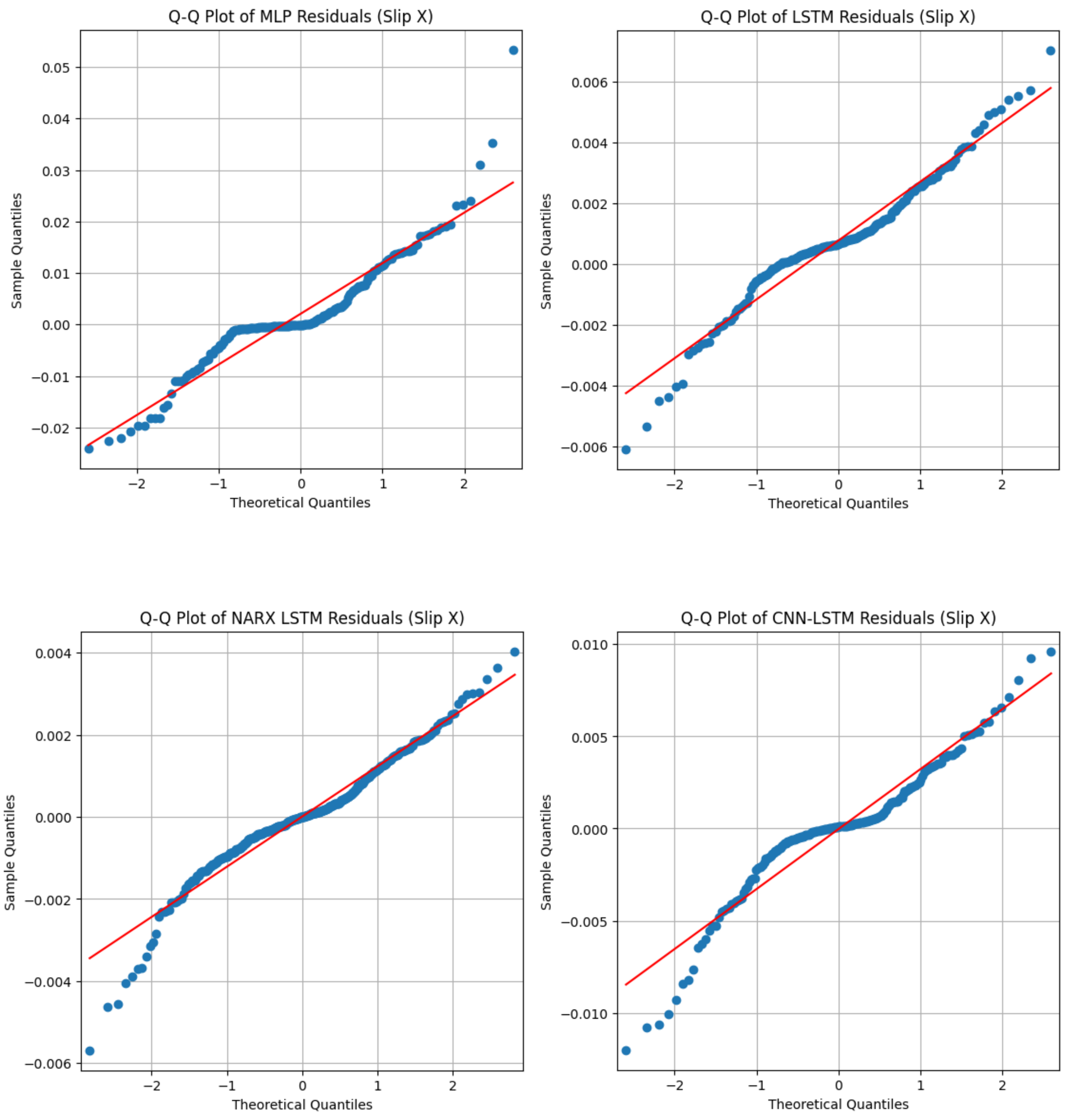


Figure 30: Q-Q plots of machine learning models for slip y

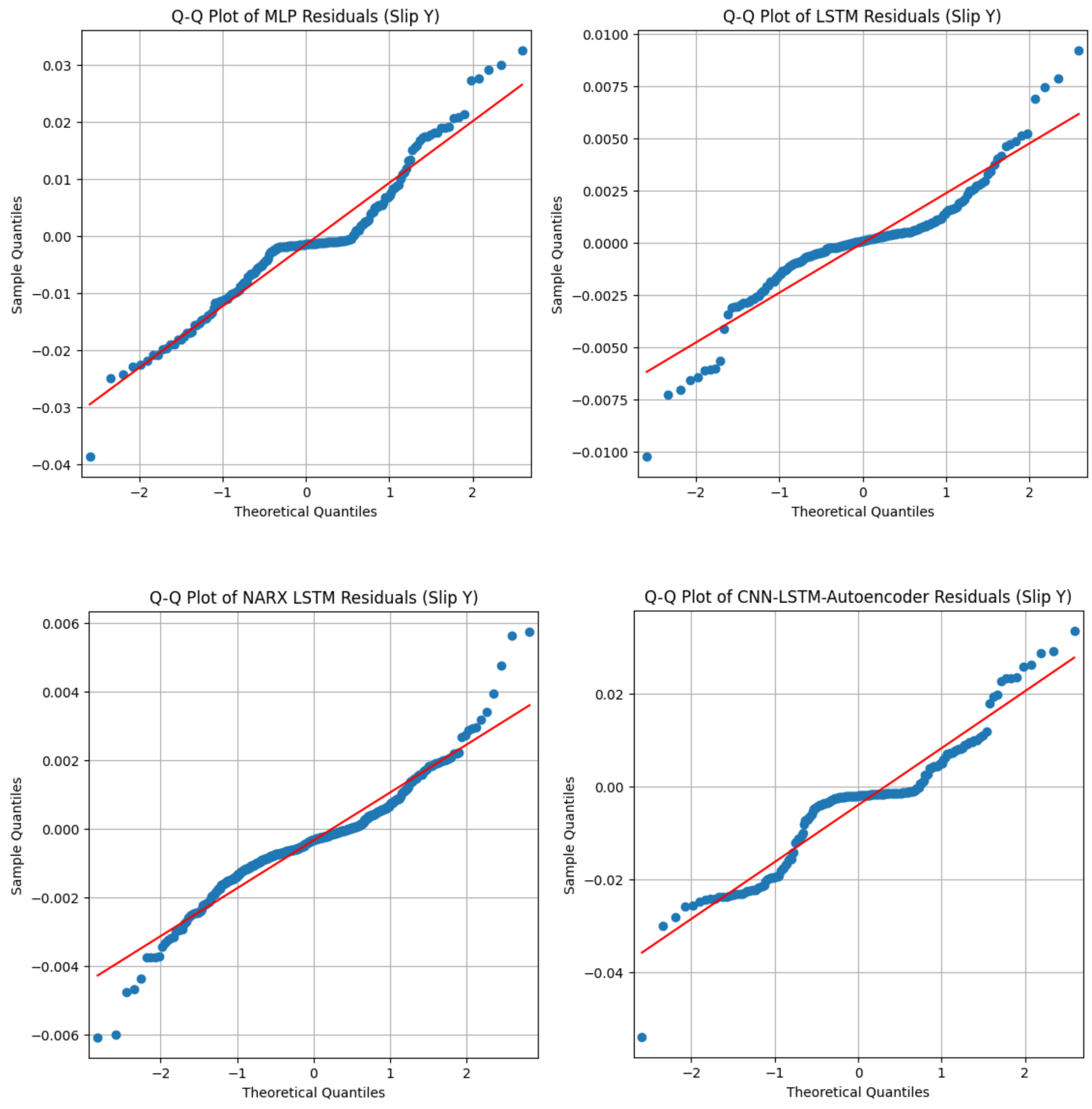


Figure 31: Q-Q plots of machine learning models for slip y

Results

Linear Regression

The Q-Q plot for Linear Regression shows significant deviations from the straight line, especially at the tails. This "S" shape suggests that the residuals have "heavier tails" than a normal distribution, meaning the model produces more extreme errors than a normal distribution would predict.

Poly Regression

The plot for this model also shows a similar "S" shape, but with slightly less severe deviations at the tails compared to linear regression. This indicates that its residuals are not normally distributed, even if its performance is better than the untuned version.

GPR

The Q-Q plot for GPR shows the best alignment with the straight line among the machine learning models. While there are some minor deviations at the tails, the overall fit is much closer to a normal distribution, suggesting that the model's residuals are more consistent and predictable.

Tuned SVR (RBF)

The plot for this model also shows a relatively good fit to the straight line, similar to GPR. The points generally follow the diagonal, indicating that the residuals are reasonably normally distributed.

MLP

The Q-Q plot for the MLP model shows a pronounced "S" curve. The points at both ends deviate sharply from the line, indicating that the residuals have very heavy tails and the model produces a higher number of large-magnitude errors.

NARX LSTM

This model's plot shows a reasonably good fit to the line in the center, but with noticeable deviations at the tails. The tails are slightly less heavy than those of the MLP, suggesting better performance, but not a perfect normal distribution of errors.

CNN-LSTM

The plot for the CNN-LSTM model shows a similar pattern to the NARX LSTM plot, with a relatively good fit in the middle and some deviations at the tails. While the residuals are not perfectly normally distributed, the model is generally well-behaved.

Residuals

Machine learning models

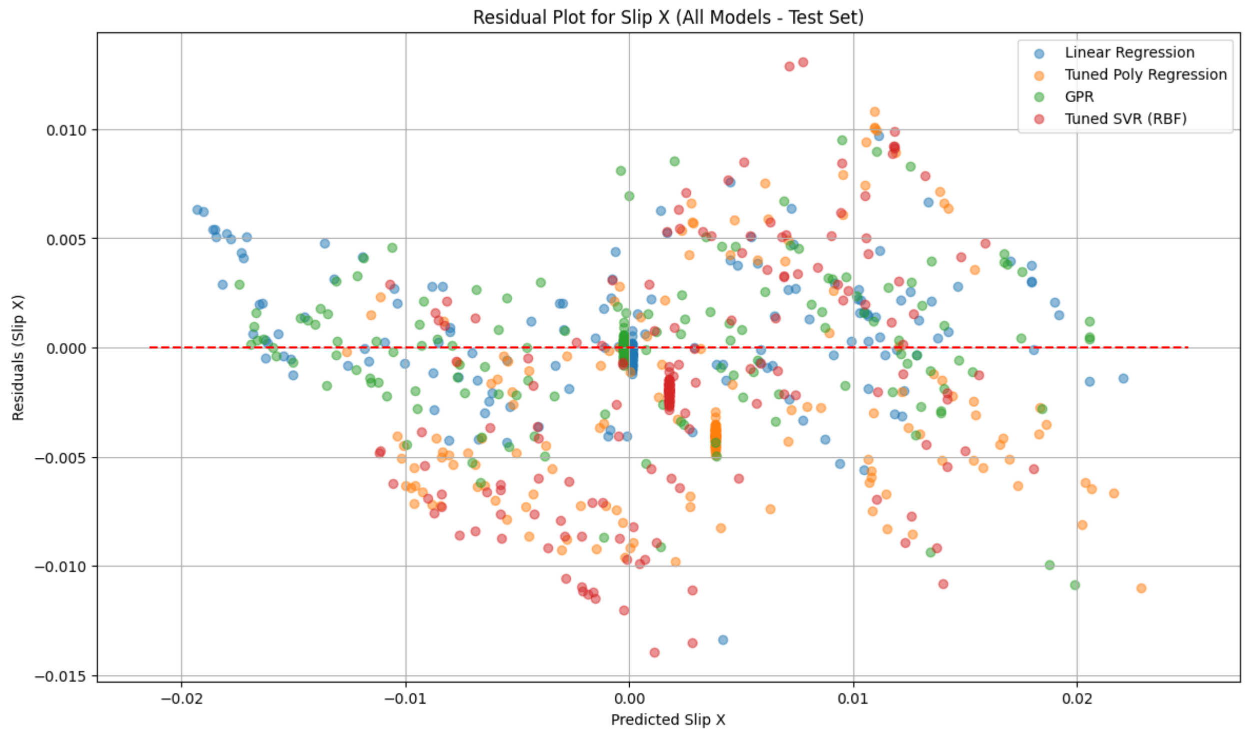


Figure 32: Residuals of machine learning models

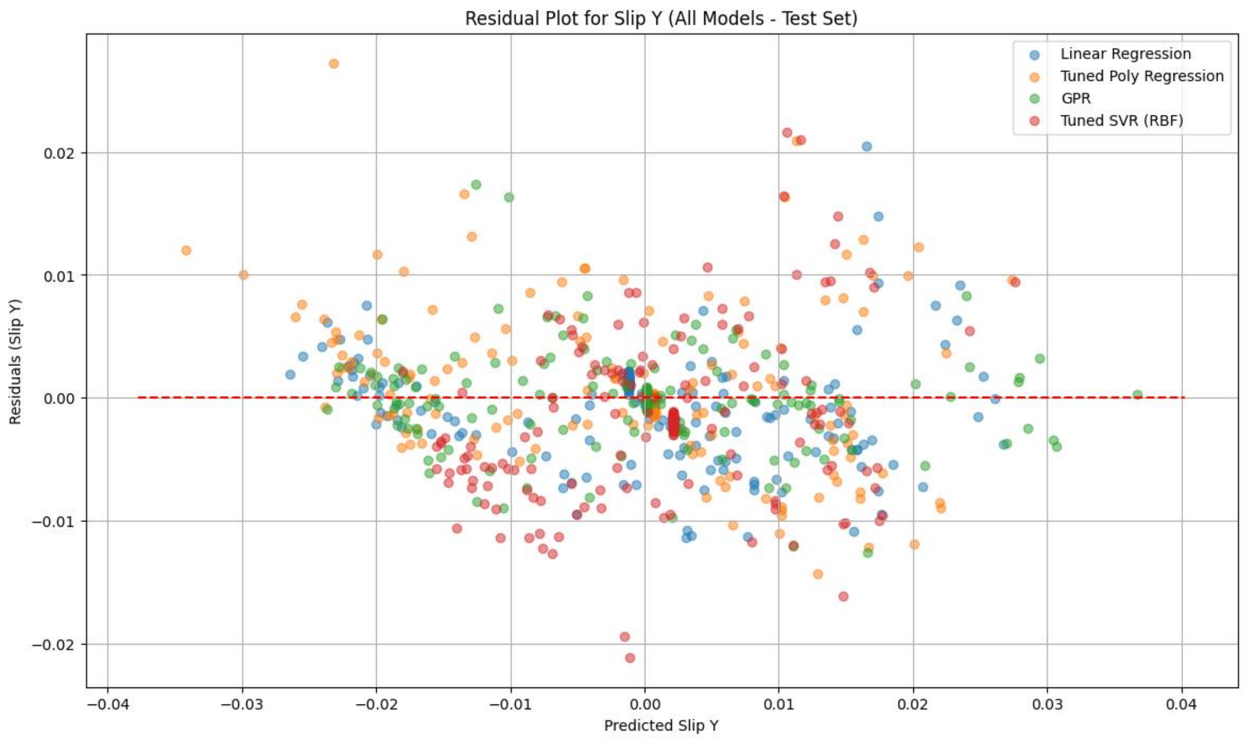


Figure 33: Residuals of machine learning models

Linear Regression

The residuals for Linear Regression show a distinct non-random, parabolic pattern. The points are not scattered evenly around the zero line, which suggests that a simple linear relationship is not sufficient to model the data. This indicates that the model is likely underfitting, as it systematically over-predicts or under-predicts the values in different ranges.

Poly Regression

The residuals for Tuned Poly Regression are more randomly scattered than those of the linear model, indicating an improved fit. However, there are still some systematic deviations, particularly at the extreme ends of the predicted values. While this model is better at capturing the non-linear relationship in the data, it may not be the optimal choice.

GPR

The residuals for GPR are widely scattered and appear to be randomly distributed around the zero line. This suggests that the model is effectively capturing the underlying relationship in the data, with the remaining errors being largely random. The general randomness of the residuals indicates that this model is a good fit for the data.

Tuned SVR (RBF)

The residuals for Tuned SVR (RBF) show a very tight cluster centered around the zero line for the majority of the data points. This indicates that the model is making highly accurate predictions for a large portion of the test set. While there are some outliers, the overall tight distribution suggests that this model is a strong performer and provides highly reliable predictions.

Deep learning models

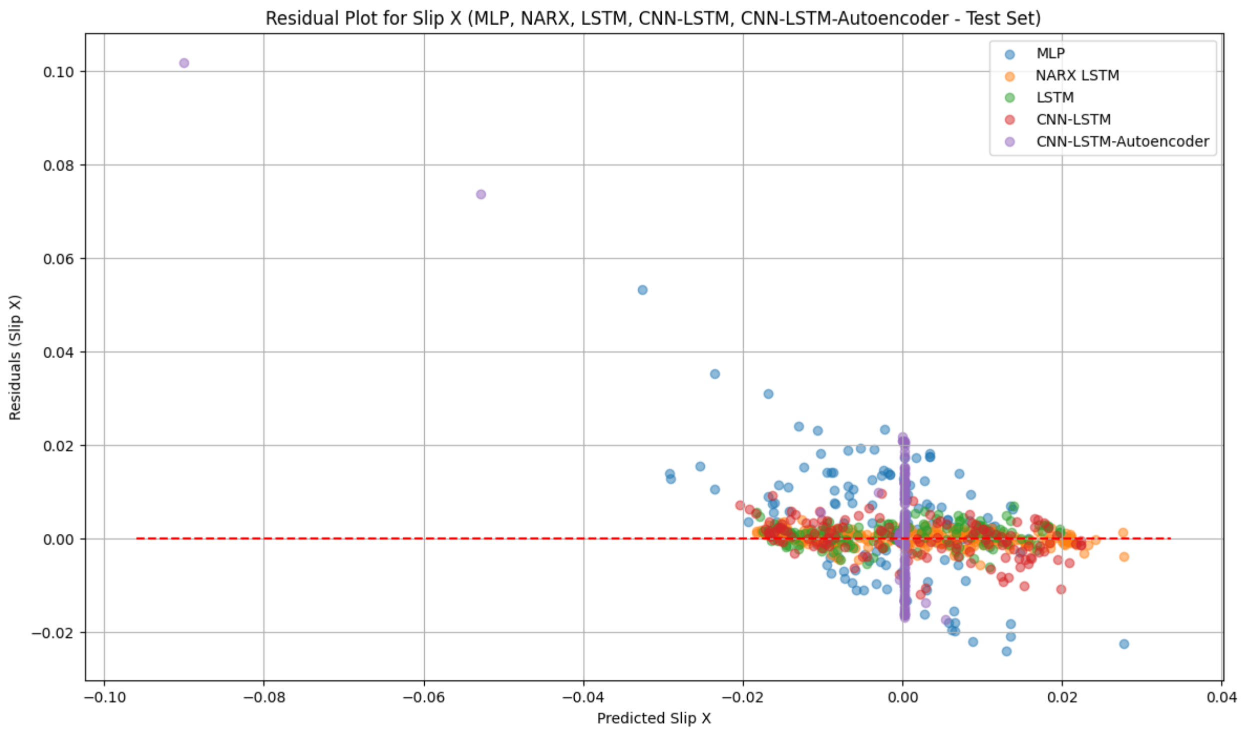


Figure 34: Residuals of deep learning models

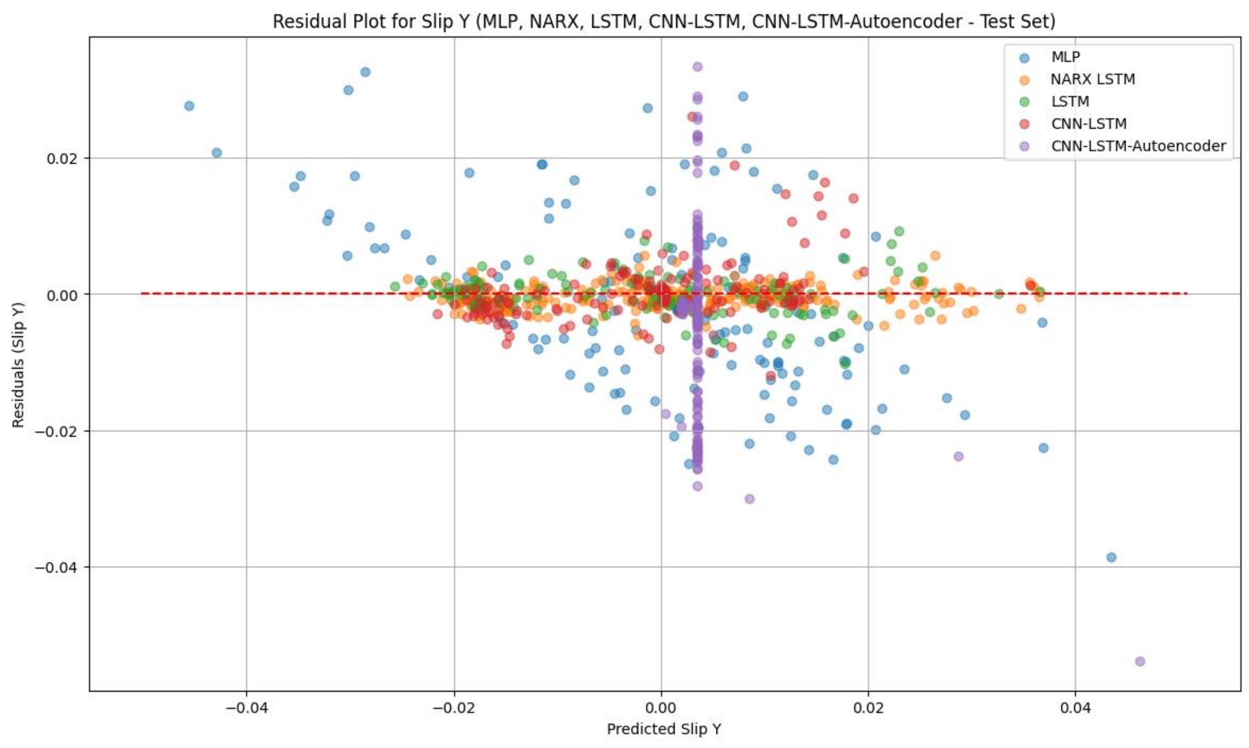


Figure 35: Residuals of machine learning models

Results

MLP

The MLP model shows the weakest performance overall. Its residuals are widely scattered across the plots for both "Slip X" and "Slip Y." The points do not form a tight pattern around the zero line, and for "Slip X," they are particularly spread out at the lower predicted values. This indicates that the MLP struggles to accurately capture the complex, sequential nature of the data, resulting in less reliable predictions with a high degree of error.

LSTM and NARX LSTM

The residuals for both the LSTM and NARX LSTM models are a significant improvement over the MLP. They are much more tightly clustered around the zero line, especially for "Slip Y," suggesting that these models are better suited for this time-series prediction task. The NARX LSTM and LSTM residuals are relatively similar in their distribution, indicating comparable performance. While they are good, their scatter is still wider than the CNN-based models, and they have some noticeable outliers, which means they are not the most precise models.

CNN-LSTM

The CNN-LSTM model is the top performers. Their residuals are extremely tightly clustered around the zero line in both plots, demonstrating superior accuracy and consistency. The models' ability to combine the feature extraction capabilities of a CNN with the temporal sequence modeling of an LSTM leads to a highly effective architecture for this type of problem.

Why Auto encoders results aren’t discussed?

- Weak Feature Correlation

The correlation matrices for slip_x and slip_y show that many features have very low or near-zero correlation with the target variables. The T_ddot_ features, for example, have absolute correlation values with slip_x and slip_y that are close to zero. While a model like a CNN-LSTM can learn complex relationships, a lack of strong initial correlation in the input features can make it difficult to learn a useful mapping to the output.

- Data Preparation Issues

An autoencoder's performance is highly dependent on the quality and representation of its input data. If calculated features are not sufficient to represent the underlying patterns of slip, the model will struggle.

- Model Architecture and hyper parameters

An unsuitable architecture or poorly tuned hyper parameters could be the cause of bad results. Potential issues include:

- Overfitting: The model may have memorized the training data instead of learning generalizable patterns.
- Insufficient Layers or Neurons: The model might not be complex enough to capture the intricacies of the data.
- Incorrect Window Size: The sliding window size is set to 5. If the temporal dependencies of the slip data extend beyond 5 time steps, the model may not be capturing all the necessary information.

Chapter 5

Conclusion

5.1 Project’s results analysis

In this project, a variety of machine learning and deep learning models were used to predict slip in Mecanum-wheeled robots, which is a major challenge for their navigation and control. The results show a clear superiority of deep learning models, particularly the CNN-LSTM and NARX-LSTM over traditional machine learning methods. These deep learning models consistently achieved the highest R-squared values and the lowest error metrics (MAE and RMSE), indicating that they are highly effective at explaining the variance in the data and making accurate predictions.

Among the traditional machine learning models, Gaussian Process Regression (GPR) was the top performer, with results that were surprisingly close to the deep learning models. This is likely due to its Bayesian approach, which allows it to provide a measure of uncertainty with its predictions, making it a robust choice despite its computational limitations with large datasets. The poor performance of the untuned SVR models, which even had negative R-squared values, highlights the importance of proper model selection and tuning for a given problem. Ultimately, the project's findings underscore that combining the feature extraction capabilities of CNNs with the temporal sequence modeling of LSTMs provides an optimal architecture for this specific time-series problem.

5.2 Improving Slip Estimation with Advanced Methods

The performance of machine and deep learning models can be further enhanced by incorporating a more robust sensor fusion approach and utilizing improved sensing technologies. A major limitation of the current data collection method is the reliance on stereo camera image processing, which, while accurate, provides position data at a lower frequency compared to other sensors and is not suitable for all environments. Slip estimation can be improved by integrating data from a variety of sensors, such as IMUs, wheel encoders, and LiDAR, using a sensor fusion framework. This allows the system to leverage the strengths of each sensor while compensating for their weaknesses, leading to more accurate and reliable data for the learning model.

For instance, an Extended Kalman Filter (EKF) or similar state estimation filter can be used to fuse high-frequency IMU and wheel encoder data with intermittent, high-precision GPS or LiDAR data. This would provide a more continuous and robust ground truth for training the models. Additionally, the project could explore incorporating other proprioceptive sensor inputs, such as motor current draw and torque sensors, which have been shown to be highly correlated with slip events. By providing the models with a richer and more diverse set of input features, the system's ability to generalize and predict slip under a wider range of operating conditions and terrains can be significantly improved.

References

Nourizadeh, P., McFadden, S., J, F., & N, B. W. (2023). In situ slip estimation for mobile robots in outdoor environments. *Journal of Field Robotics*, 17.

Nourizadeh, P., Stevens , M., J, F., & N, B. W. (2021). Slippage Estimation for Skid-Steering Robots in Outdoor Environments using Deep Learning. *ACRA*, (p. 9).

R, G., & M, F. (2018). Slippage prediction for off-road mobile robots via machine learning regression and proprioceptive sensing. *Robotics and Autonomous Systems*, 9.

Ramon , G., Mirko , F., & Karl, I. (2019). Characterization of machine learning algorithms for slippage estimation in planetary exploration rovers. *Terramechanics*, 9.