C00313519_Gokul_Thillainathan.docx

by Gokul Thillainathan

Submission date: 19-Aug-2025 01:36AM (UTC+0100)
Submission ID: 2731645212
File name: C00313519_Gokul_Thillainathan.docx (1.69M)
Word count: 10058
Character count: 59768

REACTIVE TO PREDICTIVE: ANOMALY DETECTION IN ROBOTIC ARM USING DEEP LEARNING

Gokul Thillainathan
MSc. Data Science
South East Technological University
Carlow, Ireland
gokulthillai2001@gmail.com

ABSTRACT

The rapid adoption of collaborative robots in manufacturing, logistics, and inspection sectors serves as a cornerstone of Industry 4.0, enabling intelligent and safe human-machine collaboration that enhances efficiency and productivity. As these robots become integral to mission-critical processes, their reliability and ease of maintenance assume central significance. Conventional maintenance paradigms—varying from reactive repairs to periodic overhauls—are proving insufficient in environments characterized by high throughput and tight precision tolerances. Such reactive regimes typically overlook gradual deterioration at the level of individual joints, permitting degradation to advance until catastrophic failure manifests. The consequences—unscheduled outages, diminished product quality, and inflated operational costs—underscore the urgency for more proactive interventions.

This research proposes a real-time, data-centric predictive maintenance framework tailored for collaborative robotic platforms. The architecture leverages state-of-the-art deep learning algorithms—specifically Long Short-Term Memory Autoencoders (LSTM-AEs) and Gated Recurrent Unit Autoencoders (GRU-AEs)—to discern anomalies in joint performance by reconstructing and analyzing multivariate time-series signals recorded during normal operation. The feature subset comprises actual and set-point measurements of joint position, velocity, and motor current, selected for their mechanical relevance and cross-validated using Principal Component Analysis (PCA) to balance physical interpretability with statistical robustness.

To ascertain the robustness of the detection mechanism, a head-to-head analysis of LSTM- and GRU-based autoencoder configurations is performed, with performance gauged by anomaly detection precision, mean and peak reconstruction errors, and wall-clock training and inference times.

While explicitly developed for collaborative robots, the framework outlined herein does not constrain itself to any particular platform or vendor. Its generality permits deployment on any robotic architecture that returns joint-level telemetry streams. By integrating deep-learning-powered anomaly characterization with thresholds rooted in domain expertise, the approach yields a proactive, interpretable, and scalable maintenance mechanism. Such a mechanism fortifies the robustness and productivity of smart robotic fleets, thereby

broadening its impact across multiple industrial applications.

1 INTRODUCTION

The application across industrial processes alongside service industries has massively increased due to the capability of collaborative robots (cobots) to coexist with human labor safely, to learn and cope with uncertain workplaces, and to execute numerous tasks with high repeatability alongside precision. The principal facilitator for most Industry 4.0 initiatives, cobots are integrated extensively across automotive manufacturing lines, electronic production, logistics systems, handling of drugs, etc. Their extensive deployment across high-throughput, safety-critical, along with quality-sensitive, applications has created the need for systems possessing the capability to attain long-term reliability along with performance.

But, like on all other electromechanical systems, cobots will also break down gradually with age. Joints' wear, gear backlash, variances on the friction's coefficient, thermal effects, encoder drift, and actuator wear can degrade responsiveness as well as accuracy on the cobots. The deterioration occurs first as slight differences between commanded (target) and the measured (actual) joint values—the position, the velocity, and the current readings. If not corrected, those faults will cascade to system-level failure, unscheduled shutdowns, or even safety accidents. Specifically, fault detection is required to mitigate operating hazards as well as minimize maintenance expenditures.

Collaborative robots (cobots) have swiftly evolved into mainstone elements of contemporary industrial automation architectures. Distinct from conventional industrial automation arms, cobots are engineered to coexist concurrently and harmlessly with human personnel, thereby facilitating enhanced flexibility, rapid reconfiguration, and increased throughput across sectors, including advanced manufacturing, logistics, healthcare assistance, and multimodal quality inspection. Their capacity to function in fluctuating operational settings and to undertake shared, assistance-driven tasks has accelerated their integration into smart factories and production lines underpinned by Industry 4.0 paradigms.

Contemporary deployment of cobots in critical operational settings renders reliability and performance consistency non-negotiable for securing seamless, non-disruptive operation. Nevertheless, cobots, being electromechanical entities, are not immune to progressive deterioration; risks include progressive mechanical wear, increasing joint backlash, cumulative sensor offset, thermal-induced drift, and wear arising from extended operational cycles. Such degradative processes often initiate as deceptively minor joint anomalies—namely discrepancies between commanded and sensed states, atypical velocity transients, or uncharacteristically high electrical draw—but possess the capacity to culminate in total system collapse if corrective actions are deferred.

Conventional maintenance frameworks applied to industrial robotics have predominantly

operated within either post-failure rectification or demand-driven, chronological service intervals, executing interventions solely according to a preordained timetable irrespective of the robotic system's prevailing physiological state. Although these regimes afford a measure of operational regularity, they falter in high-throughput contexts in which even brief, unanticipated unavailability can translate into substantive economic and logistical penalties. Furthermore, they lack the sensitivity to identify and mitigate incipient, joint-specific wear patterns that develop insidiously, eluding detection until performance metrics or end-of-line testing reveal the cumulative impact.

Predictive maintenance, powered by analysis of data and real-time monitoring, is an intelligent solution. It attempts to anticipate failures before they occur by detecting deviance from normal behavior, allowing businesses to plan interventions at less critical moments. Using predictive maintenance with robotic systems, however, particularly in the case of cobots, is difficult. The robots generate enormous volumes of high-frequency, multivariate time-series data from multiple joints, and their states of operation may be significantly variable in accordance with the task, payload, and environment. Moreover, labeled data sets with a known failure case are rare, and it is hard to use conventional supervised learning approaches.

To address these challenges, this paper presents a predictive maintenance framework for joint-level anomaly detection in collaborative robots. The approach utilizes advanced deep learning architectures in the guise of Long Short-Term Memory Autoencoders (LSTM-AE) and Gated Recurrent Unit Autoencoders (GRU-AE) to learn typical robot joint behavior and detect anomalies representing incipient faults. By monitoring input features comprising actual and target values of joint position, current, and velocity, the system can discover temporal relationships and recognize subtle disparities that are harbingers of blatant failures.

1.1 PROBLEM STATEMENT

Companies straddling mechanical, pharmaceutical and aircraft engineering need to cope with high-precision activities in, assembly, drilling and even inspection. Even so, long-term accuracy represents an ongoing problem for the user as accuracy can often be slowly deteriorated. In practice, operational differences can exist between where the robot actually is placed position, velocity, or force-depending on of commanded position due to the contribution of any errors of geometry that can be thought of as, any misalignment between links or tool displacement and/or some other non-geometric/procedural circumstances. So, regards errors of these aspects the variations provide 'jittering' version of operation that is compounded from differences perhaps for a joint geometry or temperature due top dependencies, or cover other operational dynamics of the robot as backlash or hysteresis and potentially influenced by external environmental positions as gravity or external loading issues. Errors of many kinds result in manufacturing defects; increased overheads; reduced efficiencies/throughput and either unacceptable downtimes, or costs of remaking defective circumstances.

In most cases traditional maintenance responses are reactive or conditional: they do not always eradicate some creeping accuracy loss. They also allow unacceptable downtimes and extremely costly costs of remaking operations. This needs to be addressed via recording continuous observations of sensor data sources and responding with predictive analytics and machine learning or equivalent understanding what may generally occur to a robot in order to develop timely alerts that could favor delivering proactive maintenance for constant quality production to enhance reliability.

1.2 RESEARCH QUESTION

1. How can anomaly detection methods be applied to fairly determine abnormalities in the movements of industrial robot joints through the examination of the difference between their actual and targeted position?
2. Which RNN architecture LSTM or GRU is optimal for concurrent anomaly detection, with respect to accuracy, computational complexity, and robustness?

1.3 RESEARCH OBJECTIVE

Develop an anomaly detection model to identify industrial robot joint deviation based on differences analysis among target and actual position, velocity, and current.

Design and implement an LSTM-based Autoencoder architecture to model healthy joint behavior and detect temporal deviations indicative of early-stage faults in collaborative robots.

Design and implement a GRU-based Autoencoder architecture for the same anomaly detection task, optimizing for runtime efficiency and reduced model complexity compared to LSTM.

Compare the performance of LSTM-AE and GRU-AE models based on:

- Reconstruction error (mean and maximum),
- Anomaly detection accuracy,
- Training time and runtime performance,
- Sensitivity to short-term and sustained anomalies,
- Thresholding stability and false-positive rates.

Validate the models using synthetic anomaly injection to assess their robustness in simulating real-world degradation scenarios.

1.4 SIGNIFICANCE

This paper presents a predictive maintenance framework for cobots using the LSTM and GRU autoencoder-type models to predict joint-level anomalies from multivariate time-series inputs. The framework utilizes domain-specific properties like joint position, current, and velocity and applies both statistical (e.g., thresholds of reconstruction error) and

engineering (e.g., angle tolerances) knowledge to label and interpret the anomalies. Synthetic anomalies are injected for the test set in order to simulate the reality of degradation processes for the purpose of system verification.

Its value lies in being able to produce an interpretable, real-time, joint-special, and anomaly-detection-capable proactive maintenance system without the need for constant human sensing or pre-programmed fault signatures. Designed with collaborative robots as its application space, the architecture scales up or down to any robotic platform with joint-level telemetry, be it traditional industrial arms, mobile manipulators, or autonomous inspection systems. Being able to do so makes it of wider applicability and value to the broader robotics space.

2 RELATED WORK

Predictive maintenance is now a core part of ensuring efficiency and reliability in industrial robotic systems with Industry 4.0. Robotic manipulators such as the UR5e form a core part of smart manufacturing, and ensuring their continued operational accuracy must be maintained using advanced monitoring techniques. Anomaly detection methods, particularly those that use deep learning, are increasingly popular for detecting early degradation before failure. This survey summarizes recent progress in research on anomaly detection, fault diagnosis, and remaining useful life (RUL) estimation in the case of robot systems with a focus on data-driven methods and real-time applications.

2.1 Fault degradation in robotic arms

Different studies have investigated how mechanical deterioration affects the positional precision of robot joints. (Qiao, 2017) presented an early analysis framework that emulates geometric and non-geometric degradation with 7D sensing, highlighting the cumulative impact of hidden joint deviations on tool-center-point accuracy. (Morettini, 2021) proposed a joint torque data-driven degradation curve model in the context of data-driven fault prognosis in robot maintenance. Similarly, (Pinto, 2019) and Fahham et al. (2024) formulated diagnostic pipelines for forecasting joint stress and RUL to support proactive maintenance scheduling. For further encouraging real-time fault observability, (Galan-uribe, 2023) discussed low-latency hardware-accelerated monitoring of degradation signals utilizing FPGA platforms.

2.2 Deep learning approaches in predictive maintenance

Deep learning approaches have been noted to demonstrate resilient performance for modeling complex robotic behavior and anomaly detection. (Graabek, 2023) also compared fifteen anomaly detection methods for collaborative robots and concluded that autoencoder-based models—LSTM-AE, in particular—carried out the best detection on both synthetic and actual anomaly data sets. (Ayankoso, 2024) took this a step further by employing position, current, and velocity signals as multivariate inputs to CNN-AE and Sparse-AE models. (Shamuganathan, 2023) created a hybrid Markov-LSTM model for low-memory environments, which is hence more

flexible for application in edge computing systems. GRU-based models that estimate uncertainty, as outlined by (Kang, 2022), were also discovered to improve the reliability of real-time robotic process anomaly detection using prediction confidence measures.

Recent work has also witnessed Transformer-based models being viable candidates in robotic health monitoring. (Gao, 2025) presented a multiscale temporal memory Transformer (MTMT) framework for RUL estimation with degradation awareness. Their model effectively utilized long-term dependencies between sensor time-series and performed better than traditional RNNs, demonstrating the efficacy of attention mechanisms in predictive maintenance applications.

2.3 Feature selection and multivariant anomaly detection

Multivariate signal integration is another significant aspect to improve detection sensitivity. (Gu, 2025) proposed a curvature-guided shape dynamic time warping (DTW) model with convolutional layers for investigating motion trajectory and abnormal classification of swing-arm motions. (Yang, 2024) proposed a graph-based model, AD-CIFC, which integrates feature and temporal attention mechanisms with missing data handling through collaborative imputation. Similarly, (Ball, 2024) and (Castellanos-Ramos, 2021) combined vibration, torque, and current signals with LSTM and PCA in order to detect faults in an early stage. (Espinoza, 2023) created a diagnostic system that was able to detect if the issue was sensor drift or actuator faults from temperature, positional, and electrical current signals. Hardware, (Kayan, 2025) achieved near-real-time anomaly detection using LSTM models on ultra-low-power microcontrollers and (Imrana, 2022) found that BiLSTM and CNN-LSTM networks had improved performance under noisy environments.

2.4 Statistical thresholding and reconstruction error

Other studies have also addressed anomaly thresholding. (Ayankoso, 2024) utilized percentile-based thresholds, typically at the 99.5th or 99.9th percentile, to distinguish normal operation from abnormal behavior. (Kang, 2022) instead suggested uncertainty-aware thresholds that adapt dynamically with model confidence. (Zhao, 2023) employed a model-free LSTM-AE with rolling-window statistical baselines for real-time anomaly detection in robot joint data.

2.5 Model validation and synthetic anomaly injection

Lacking as it does available labeled fault data for industrial settings, synthetic anomaly injection has been a widespread validation technique. (Graabek, 2023) and (Ayankoso, 2024) injected trajectory deviations to test detection performance, while (Shamuganathan, 2023) injected fault patterns into sensor streams. (Gu, 2025) tested with different torque inputs for

validation to check robustness against varying conditions.

2.6 Real-time deployment and edge computing

Deployability in real time remains a major issue. (Kayan, 2025) demonstrated the potential of executing LSTM models on edge devices such as the Nicla Sense ME to support low-latency, task-agnostic anomaly detection. (Park, 2022) alleviated response latency in UAV motion anomaly detection by using LSTM designs. (Ayala-Chauvin, 2024) surveyed large data frameworks such as Apache Spark to address scalability and heterogeneity in robotic predictive maintenance systems. (Qiao, 2017) also proposed a controller-level integration layer design to close the feedback-diagnostic loop of robotics systems.

2.7 Comparitive model evaluation and runtime metrics

Efficiency and complexity of models were compared in comparative analysis between typical and neural models. (Costa, 2019) set up that gradient boosting techniques offered improved accuracy with respect to logistic regression and random forests for failure classification. (Liu, 2022) introduced evolutionary architectures of CNNs with well-balanced speed and detection performance. (Imrana, 2022) highlighted that GRUs had superior runtime performance over BiLSTMs. Sparse-AE variants, were more suitable for use in low-processing power environments.

2.8 Domain-level integration for maintenance decision

Finally, transforming anomaly scores into decisionable actions has been considered a key challenge in implementation. (Morettini, 2021) and the (CSE, 2023) recommended making use of degradation curves and domain-dependent tolerances for conversion of detection signals into decisionable events. (Gu, 2025) integrated stress thresholds into their post-processing of faults and (Espinoza, 2023) used severity measures to order faults. This work apply a joint-level reconstruction error map strategy here and translate model outputs to real-time warnings and critical alerts through domain-specific thresholds (±0.1°–0.5°), enabling operational fault awareness.

In summary, current literature prioritizes the application of deep learning—namely LSTM, GRU, in multivariate robot joint temporal pattern learning and anomaly detection. Feature fusion, real-time deployment, and statistical thresholding are likewise pinpointed as essential components of an effective anomaly detection pipeline. While current approaches have advanced detection accuracy and timeliness, there remain shortcomings in joint-specific diagnostics, domain-aware scoring, and edge deployment scalability. My work addresses these gaps immediately by suggesting a real-time anomaly detection framework with LSTM-AE and GRU-AE models with joint-specific scoring, synthetic anomaly verification, domain-threshold mapping, and real-time deployability.

3 METHODOLOGY

3.1 DATA DESCRIPTION

For the purpose of this experiment, an openly published time-series test set by the National Institute of Standards and Technology has been specifically created within the benchmark experiment called Degradation Measurement of Robot Arm Position Accuracy. The test set has been selected because it emphasizes the long-term operation behavior of collaborative industrial robots as well as because it includes high-temporal resolution, therefore making it the best suited for the identification of anomalies within the operation of the joints.

It was tested under a lab test environment in which a 6-degree-of-freedom co-operative robot manipulator needed to chase a programmed path for some thousands of cycles. During the process, sensor inputs were internally generated within each cycle time step to imitate changes to the robot's performance under mechanical wear and build-up of operational stress. The logging was on the onboard joint-level sensors on the robot, with the result that there were concurrent, accurate readings on all axes.

Every record in the dataset corresponds to one instant in time in the motion of the robot and contains the entire snapshot of sensor feed for all six joints. The data covers multiple physical and electrical properties within the joint behavior of the robot. After some preliminary exploratory analysis along with dimensionality reduction using Principal Component Analysis (PCA), chose the most explanatory features by variance contribution along with direct correlation with joint-level anomaly detection.

The last set of characteristics employed in this research are:

- Target Joint Position and Actual Joint Position
- Target Joint Velocity and Actual Joint Velocity
- Current Joint Target Actual Joint Current

These six variables, one for every parameter, target, and actual, were sampled for every one of the six joints, obtaining a structured multivariate time-series data set. They were picked because they convey the native dynamics of robot motion and allow control precision, energy usage, and mechanical consistency to be assessed. Position, velocity, or current target and actual difference tend to lead to root causes like mechanical wear, joint resistance, backlash, or sensor drift.

With respect to many robot datasets offering solely end-effector performance or task-level, abstracted outcomes, this corpus comprises raw joint-level sensor resolution — useful for predictive maintenance systems, for which the operating level is defined as the hardware level. High-granularity information allows modeling on a timescale with techniques like RNNs, instead of being label-based instantaneous failure labels.

For this assignment, the NIST dataset is the ideal to start with, containing realistic, reproducible, sensor-dense time-series data, suited best for the task of identifying anomalies on the joint level with deep learning methods.

As shown in figure 1 the data flow starts with the Dataset. The Dataset is cleaned to eliminate errors and inconsistencies during the Cleaning phase. Then enters into Model Training. The Output is useful, including predictions or anomaly detections. The lifecyle from raw data to final result eliminates any uncertainty of if the insights are reliable.

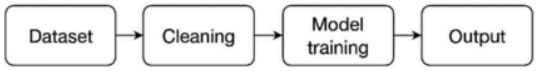

Figure-1: Cobots-deviation monitoring Framework

3.2 DATA CLEANING

Raw data has been cleaned thoroughly prior to the model construction/reconstruction or the analysis so that structural consistency, numerical correctness, and overall integrity of the information are obtained. Since the sensor-recorded time-series nature of the sensor data is the same as the NIST data made available to the general public, there were some value differences as well as format issues that were observed and addressed strictly.

3.2.1 Handling mixed data types and symbolic artifacts

Among the first issues to be discovered when it came to exploration were the majority of the numeric columns, including columns for joint sensors, being of the string-formatted type rather than the pure numerical type. The original files were not properly parsed, so numerical inputs were taken as strings.

There were also cells within other columns with extraneous characters and thus it could not be sensibly read or calculated. The differences were a threat to additional analysis and the integrity of the model.

To address these concerns:

• All column values relevant to features were cast to numeric by being explicitly converted, and those rows failing the conversion were cleaned or dropped following review.
• String-cleaning functions, along with filtering by regular expression, were used to remove any unwanted characters, symbols, or embedded units.
• A check step ensured all the values within the features held were always numeric and did not contain formatting irregularities.

This pre-cleaning has been used to prevent the propogation of silent failures during the normalization, windowing, or training step of the model.

3.2.2 Addressing missing and incomplete values

Missing values were sometimes observed, not uncommon for high-frequency robotic sensor readings, because of sensor delay, transmission delays, or logging error.

To answer this:

• Single or double consecutive missing values' short-duration gaps were dealt with by creating the forward-fill imputation, not disrupting time-series continuity or natural trends.

• For wider gaps (across several time steps), mean-based or difference-interpolating of the signal were employed, according to the type of the absent pattern. These led to the

reconstruction of values so the local character of the signal would be preserved.

• When absent was too exhaustive or spanned more than one joint simultaneously, the related sequence did not get included to avoid introducing bias or ambiguity to the model.

This two-step procedure guaranteed it would retain the majority of the information but saved it from spurious or artificial smoothness of signal.

3.2.3 Feature consistency

The data set contains joint-level features for the six axes, it became necessary to ensure equal structure accompanied by data quality for every joint. The above cleaning process were applied to all the actual and target values of the position, velocity, and current for Joint 1 to Joint 6 with equal efficacy.

After this cleaning process, the resultant dataset was a well-structured, numerically coherent, and high-fidelity multivariate time-series geared towards temporal modeling and anomaly detection.

3.3 DATA PREPROCESSING

Following the cleaning up and the checking of the integrity of the structure of the dataset, the subsequent crucial step followed as a holistic preprocessing approach for converting raw joint-level sensor data to the correct format for training deep learning models. The step proved essential to enable the model to learn adequately the time dependencies alongside the fine-grain behavioral features of robotic joint degradation.

Because this research utilizes a time-series model through autoencoders utilizing Recurrent Neural Networks (RNNs) on raw multivariate inputs, the raw multivariate inputs must be transformed from irregularly structured into suitably structured, sequential inputs conducive to model deployment with respect to retained joint motion's time features. The subsequent sections the preprocessing process utilized within this research.

3.3.1 Feature engineering

The first task of the preprocess step was to extract features to describe the joint deviation and time behavior on the direct level. For the six joints, the absolute error between the commanded (wanted) and actual position was computed for each of the joints. The measure, referred to as the joint error, is the direct measure of the joint performance drift and mechanical wear.

$$\boldsymbol{Joint\ Error_j = |Target\ Position_j - Actual\ Position_j|}$$

- j denotes the joint index
- Target Position : the desired (commanded) position of joint j
- Actual Position : the measured position of joint j
- The absolute value ensures that the error is always non-negative.

Apart from the shared error, the following sensor-based features were kept for all the joints:

Specific Joint Velocity

True Joint Velocity

Potential Common Current

Effective joint current

Collectively, the five parameters include both the kinematic and the dynamic for motion for the robot. While the joint error communicates unanticipated motion deviation, the velocity terms and the current terms are utilized to specify the degree to which actively or effectively each joint is functioning. The parameters were discovered to be most informative by the convergence of previous knowledge of the field with Principal Component Analysis (PCA) to ensure they described most of the variance for the set across multiple of the trajectories with multiple cycles.

Features were extracted joint-wise, i.e., data for every joint separately was investigated. That was to allow joint-specific identification of anomalies, essential for detection of localized degradation without being averaged out by global averaging across the system.

3.3.2 Normalization of feature space

After determining which features were relevant, the feature channels were all normalized to ensure that the results were numerically stable and that no one feature would dominate the training process by having a greater scale than the others.

Each feature was normalized independently via MinMax scaling, which takes the determined values and rescales the data to fall within a specified range of [0, 1]. This form of scaling is especially beneficial to deep learning models with activation functions generally seen in those types of models (for instance, ReLU) because it allows you to scale the features consistently without disrupting any temporal relationship in the data.

Normalization was applied on a per joint basis, and the scaler was fit with the training dataset independently. The same scaling operation was then performed on the respective features in the test set so that the normalization would be consistent, and no data leakage would occur from the training dataset to the test dataset. With this process, we kept the evaluation process relatively pure while allowing for feature distributions across training and evaluation to be much closer on average.

3.3.3 Temporal windowing and sequence construction

To account for time-dependent behavior and ease the classification of evolving anomalies, the continuous time-series data were split into fixed-length overlapping sequences, also referred to as temporal windows. Each sequence captures a moment in joint behavior over a specified

number of time steps and serves as a single training sample for the RNN autoencoder.

For this study, the window length of 50 time steps was used to sample joint behavior. It should be noted that this length was not previously studied; rather, it was selected based upon empirical testing and domain considerations - providing enough temporal context to capture motion cycles and transient deviations, while not making the sequences too long to be processed efficiently regardless of AI/ML techniques.

The approach selected utilized a sliding window, meaning the window was shifted forward one time step which created overlapping sequences. The overlapping sequences not only provides a higher density of training samples, but also helps the model generalize better by exposing it to numerous time alignments of behavior patterns.

for an input array X of shape (T,F)(T, F)(T,F), where T is the number of time steps and F is the number of features, the data was transformed into a 3D array of shape (N,L,F)(N, L, F)(N,L,F), where:

- N=T−L+1 is the number of windows,
- L is the window length (50),

F=5F is the number of features per joint.

Each temporal window captures 50 time steps of all five features in time-aligned consecutive rows, meaning the model will hopefully to learn not only the current state, but how that joint has been evolving over time.

3.3.4 Output structure and readiness for modeling

The output dataset was then constructed for each joint as a three-dimensional array:

$(Number\ of\ sequences, Sequence\ length, Features\ per = (N, 50, 5)$

Where:

- N: number of sequences (samples in your dataset)
- 50: length of each sequence (timesteps)
- 5: number of features per timestep (dimensionality of input at each time step)

This structure met the input expectations of the LSTM and GRU layers, which was a required format for temporal learning of batched sequential data. Each sequence served as both the input and target to allow the model to learn to reconstruct normal operational sequences in autoencoder training.

At the conclusion of this preprocessing pipeline, the dataset can be said to be transformed from a raw, unstructured time-series into a numerically consistent, temporally segmented, semantically rich input, fully prepared for unsupervised training with deep recurrent autoencoders.

4. MODEL IMPLEMENTATION

This section is a complete technical description of the model implementation for joint anomaly detection in robotic manipulation using

autoencoder architectures based on long short-term memory (LSTM) and gated recurrent unit (GRU) cell design. This includes a description of the rollout of this decision including model layering options, architectures, a breakdown of hyperparameter settings, and discussion on training. The focus is particularly on conspicuously documenting reproducibility, traceability, and a practical justification for any engineering choice, specifically parameter tuning, layer construction, thresholding techniques, and anomaly binning.

4.1 ARCHITECTURE OVERVIEW AND LAYER CONFIGURATION

We modeled each of the robotic joints (joints J1 - J6) independently using two deep learning architectures, one based on LSTM and the other based on GRU. The models maintained a unidirectional single-stacked autoencoder, with a symmetrical encoder-decoder structure. The total number of trainable parameters per model instance was kept below 50,000 to allow for repeatability across the joints and limit overfitting of the dataset.

Encoder structure

The encoder consists of a single recurrent layer. The specifics of the layer are:

- Type LSTM or GRU selected dynamically during each experiment
- Units 64 units
- Return Sequences False, since we only want the final hidden state to represent the full sequence input
- The input shape is (50, 5) per sample 50 time steps with 5 joint-level features.

We arrived at 64 units as the final number after experimenting with 32, 64, 128:

- 32 units, resulted in underfitting of each of the sequences resulting in high reconstruction loss.
- 128 units was disproportionately large and increased the time to train and made inference slow.
- The number of 64 units allows for an optimal trade-off between representational capacity and speed.

The encoder reduces the full 250 dimensional (50x5) input to a single 64 dimensional latent representation.

Latent repeat layer

A Repeat Vector (50) layer was used to inflate the latent code to the shape required by the decoder (i.e. it broadcasts the 64-dimensional shape back to 50 time-steps to build the context for decoding over sequences).

Decoder structure

The decoder is symmetrical to the encoder:

Type: LSTM or GRU (same as encoder).

Units: 64.

Return Sequences: True (to reconstruct a time-aligned sequence).

- The decoder outputs a 3D tensor with a shape of (50, 64).

This output is passed through a TimeDistributed(Dense(5)) layer to project each of the time steps into the original feature space:

• Dense Units: 5.

• Activation: Linear (no activation so that there are no constraints on reconstruction).

• The final output shape is (50, 5) which is the same as the input.

No dropout, batch normalization, or bidirectional layers were used in the base model to:

• Preserve interpretability of errors,

• Avoid adding stochastic variance to a sensitive anomaly detection situation,
• Lower the parameter count for training on less powerful hardware.

4.2 TRAINING DATA PREPARATION

Input sequences were prepared through a sliding window approach:

• Window Size: 50 time steps.

• Soplex Overlap: 1 (fully overlapping).

• Input Shape per Sequence: (50, 5).

• Target: The same as the input (reconstruction task).

Sequences were generated using NumPy array slicing and transformed to 3D format. Unlike most anomaly detection pipelines that learn solely from normal data, this code learned models from mixed data (normal and abnormal sequences). This design choice mimics real-world situations where robot logs in the past are rarely pre-segmented or annotated by anomaly class.

4.3 MODEL COMPILATION AND OPTIMIZER CONFIGURATION

All models were compiled using:

• Optimizer: Adam(learning_rate=0.001)

• Loss Function: Mean Squared Error (MSE)

Adam was chosen for its robustness and advantage in RNNs. RMSprop was also evaluated, but the convergence was marginally slower. The learning rate was fixed at 0.001, from previous tuning, as dropping to 0.0005 caused the training to take longer, without achieving any better accuracy of reconstruction.

The model was compiled once per joint and type of architecture. Models were built but discarded per run, rather than preserved, in order to save memory and prevent a potential leak of model parameters.

4.4 TRAINING PROTOCOL

The training protocol was the same for every experiment to maintain consistency:

• Epochs: 10 (fixed across all joints, and both models).
• Batch Size: 32.
• Shuffle: False (to maintain temporal structure).
• Validation Split: None, as they were focused on thresholding post-training rather than generalization metrics like val_loss.
• Callbacks: No callbacks. Neither early stopping, nor checkpointing, nor LR schedulers, to make

sure the training loops remained identifiable and reproducible.

The number of epochs was decided based on some earlier experiments:

• 5 epochs did not fit the data, and the reconstruction error histogram was varying significantly from epoch to epoch.
• 15+ epochs achieved a marginal gain (<2%) in error reduction, while increasing training time by 40% to 60%.

Each training session was timed using Python's time.time() function for measuring model latency and to make GRU vs. LSTM efficiency comparisons.

4.5 INFERENCE AND RECONSTRUCTION ERROR CALCULATION

Following training, each trained model provided reconstruction outputs for the test data:

• Predictions shape: (N, 50, 5) — same as input.
• Reconstruction error calculations were the mean squared difference of actual and reconstructed sequences

The output is a 1D array of error scores, of length N related to the number of windows. All anomaly scoring and thresholding was done after reconstruction, and did not impact the trained model or retraining the model on labeled data.

4.6 THRESHOLDING AND ANOMALY LABELING

Anomaly detection was accomplished through an adaptive threshold:

• Calculated using the training set's 99.9th percentile reconstruction errors.

• 99.9th percentile selected to ensure false positive rate of < 0.1% when running under normal conditions.

• Others explored:

Fixed MSE thresholding → Joints thresholds could be inconsistent.

Z-score method → Outliers sensitive.

Isolation Forest → Harder to interpret, slower to use for sequence-wise comparison.

Each test sequence with error above the threshold was labeled anomalous. The corresponding labels were applied back to the raw data indices with an offset (subtracting window).

4.7 ANOMALY GROUPING AND VISUALIZATION

In keeping with the principles of interpretability, contiguous sequential anomalies were grouped into intervals with simple grouping function. The grouping function:

• Sorts the anomaly indices.

• Linear walk to cluster adjacent or continuous segments.

• Outputs (start_idx, end_idx) pairs.

For each interval, the grouping function calculated:

• Maximum joint error value.

• Average joint error value.

• Interval length.

These intervals were overlaid on:

• Joint error time-series plots (shaded red intervals).

• Reconstruction error histograms (vertical line indicating threshold).

The plots were important for validating the model's performance and for visually confirming the true-positive detections.

4.8 SYNTHETIC FAULT INJECTION

To test degradation scenarios, a synthetic anomaly was injected into Joint 3:

• Injection Window: Indices 10000-10010.

• Injection Value: A fixed offset of +0.05 in the JOINT_ERROR_J3 column.

This range was selected due to:

• The perturbation being small but persistent, which looks like sensor drift or mechanical slack.

Both LSTM and GRU models were retrained on the original (non-injected) training data and evaluated on the modified test data, with either approach correctly resulting in flagging the anomaly. This confirms that a fault temporally restrained to a short duration can be detected in the application with such configuration.

4.9 SUMMARY AND CODE MODULARITY

All model runs were wrapped in modular python functions:

a) preprocess_features(): Normalization, cleaning

b) train_autoencoder(): the architecture instantiated and trained

c) detect_anomalies(): inference and error thresholding

d) summarize_anomalies(): collate metrics and durations.

All results from the various models were registered and accumulated into pandas DataFrames for comparison.

Final metrics included:

• The number of anomaly intervals,

• The max error and mean error per joint,

• Training time per model,

• Threshold values.

This coded structure facilitates the addition of other joints and features, as well as models.

4.9.1 Post-hoc health status evaluation

A subsequent step was added to the process whereby a rule based scheme for health classification was added for the purpose of increased utility. Instead of simply identifying anomalies, the classification system assesses the

general joint performance based on a set of empirically defined engineering thresholds placed on the maximum joint error under average joint error values.

In this phase, the joint is classified into one of four status levels:

• GOOD: Although both maximum and average joint error values are under tolerances where we might argue the joint is performing predictably.

• MONITOR: The average joint error is high (>0.08), which could suggest that the joint is at a threshold where early stages of drift or deviation is occurring and warrant further monitoring.

• WARNING: The maximum error is over 0.1; and thus from a generic tolerances perspective, we can think of the joint being misaligned, worn or experiencing erroneous actuation, this becomes a moderate fault that is worth investigating.

• CRITICAL: The maximum error is over the threshold of 0.45; and this would demonstrate strong evidence of fault or degraded mechanical performance.

The boundaries were approximated from exploratory statistics on distributions on normal data and on degraded data and on robot tolerance specs coming from industry (e.g., UR5e). Analysis offers end-members like maintenance engineers a high-level summary for a moment-per-joint condition check without a human check on a check on a range or on distributions on errors.

It runs once automatically on detection and produces a final summarized consolidated joint status, ensuring easy decision-making when carrying out predictive maintenance work flows. For reproducibility of all results full code implementation is available at C00313519_Anomaly_detection

5 RESULTS AND DISCUSSION

The results and discussion section assessed the performance of the LSTM-based Autoencoder model on all six joints of the robotic arm, while a synthetic fault injection was applied specifically for Joint 3. The assessment of performance was done from two approaches: (1) the capability of detecting naturally occurring anomalies in the original dataset; and (2) the sensitivity of the model to synthetic deviations, which are introduced to approximate degradation. The results were assessed across many criteria including reconstruction error, threshold sensitivity, coverage of the anomalies intervals, and distributional aspects.

5.1 ANOMALY DETECTION ACROSS JOINTS (LSTM)

Joint 1

The evaluation for joint 1 shown in figure 2 found 10 unique anomaly intervals with maximum reconstruction errors of 0.00137 to 0.03964. Most of the flagged intervals were in the vicinity of isolated peaks, nonetheless for joint 1 reconstruction error was distributed largely below the reconstruction maximum error of 0.0015 as illustrated in the histogram dry ran here. The LSTM model's threshold was computed using the 99.9th percentile and misleadingly segmented the tail of the distributions. Importantly, oldberg etal had only modest numbers of high spikes exceed the threshold; however, manufacturing normal operator induced variability is not going to trigger false positives.

The anomaly plot in fig2 with pre-annotations provided several error spikes at the peaks, as the red shaded intervals for Joint 1 were theoretically appropriate abuting the visible spikes in the trace. For example, some intervals like those at 18157-18161 and 13869-13869, although very short, were accurately flagged noting sensitivity that the model could pick up on very locally transited error-spikes.

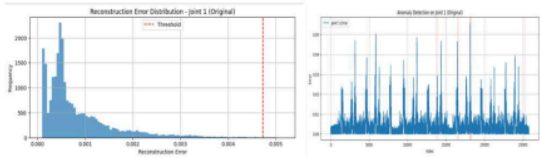

Figure 2: Joint 1 Output of LSTM

Joint 2

Joint 2 shown in figure 3 exhibited a less populated anomaly profile with only four intervals found. The maximum reconstruction errors were still low (e.g., 0.01742 and 0.01315), and the overall histogram had a right skew with most of the reconstruction errors being densely centred around zero. The threshold (≈0.010) was slightly higher than Joint 1 reflecting some of the natural variability to be expected in joint performance.

Even though there were fewer amount of anomalies, each interval occurred at very clear spikes in the signal. In this case, the model was able to separate between a low magnitude noise and observable changes in the signal. In particular, the intervals 7382-7384 and 17691-17703 indicated the LSTM model is able to detect both very short and long term anomalies when a clear aberration is presented at the signal.

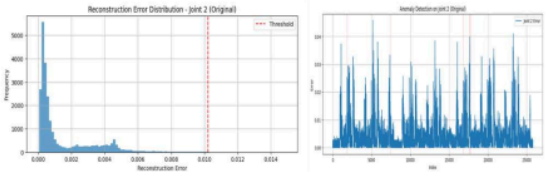

Figure 3: Joint 2 Output of LSTM

Joint 3 (Before Injection)

Joint 3 shown in figure 4 baseline results (before injection) showed only three anomaly intervals at 3749-3762, 11779-11786, and 14901-14904. Each interval and the overall reconstruction errors were very minor with maximum reconstruction errors from 0.00383 to 0.00756. Reconstruction errors had a heavily left skewed histogram pattern with a small number of high-error outliers causing the threshold to level out just above ≈0.014. However, although the reconstruction errors had an outlier or two, there were no false positives to indicate this model was behaving to form true positive characteristics.

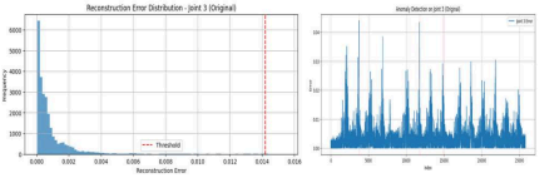

Figure 4: Joint 3 Output of LSTM

Joint 4

For Joint 4 shown in figure 5, there were four anomaly intervals at which contingencies were received and the reconstruction error was larger than all other joint data (maximum construction error in certain anomaly intervals was larger than 0.02). Overall, Joint 4 had a noisier error signal with greater spread and density in the histogram. The model was using adaptive thresholding and was able to not over detect error despite a lot of variance, within the construction error of its signal.

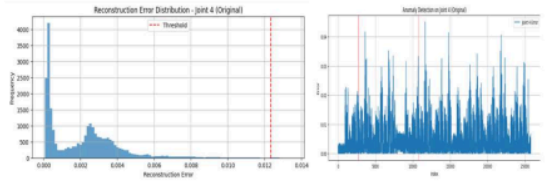

Figure 5: Joint 4 Output of LSTM

Joint 5

For Joint 5 shown in figure 6, there were six anomaly intervals where construction errors were substantial, particularly in the first few thousand indices (for example, 3929-3930, max error = 0.01716). The histogram of prediction errors was bimodal indicating two potential behavioral regimes:at least one corresponding to typical behavior, and a second potential that corresponding to a level of unpredictable instability or just a distinctive condition at that moment in time. The LSTM model was able to separate and distinguish between regimes, even if they were quite different means. The model was not responding to very low frequency baseline shifts either, which certainly provides evidence that a healthy latent representation was developed.

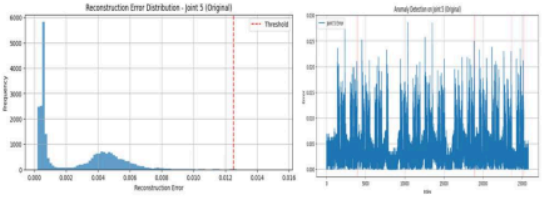

Figure 6: Joint 5 Output of LSTM

Joint 6

Joint 6 shown in figure 7 appeared to have reasonably well calibrated data, it did demonstrate true anomaly in its four anomaly intervals, and also had a reconstruction error threshold that was set larger, approximately = 0.015. The histogram indicated that while the relationship between error and reconstruction was wide ranging, the majority of its histogram was centered around 0.005-0.008, with the model only flagging its largest deviations. In addition, the dimensionality of the errors demonstrate both that the model is responding towards the anomalies, and that it is displaying appropriate classification and flagging issues, such as the errors from 22887-22894 are flagging it is responding appropriately and non-periodic issues are flagged.

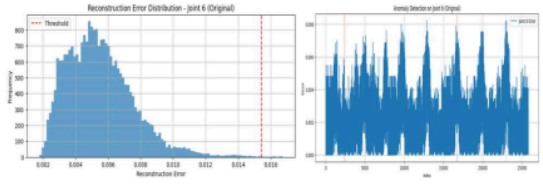

Figure 7: Joint 6 Output of LSTM

Synthetic Anomaly Injection (Joint 3)

To replicate a real-world fault, a controlled disturbance was introduced into the Joint 3's joint error between the index locations of 10000 and 10010 by injecting a constant offset of 0.05. After retraining the LSTM model with this new set of test data, the model highlighted an unequivocal set of high-severity anomalies during the intervals of:

• Interval 10006-10011 with a max error of 0.05533 (and an average of 0.04510) where both the original and threshold values were crossed by some magnitude.

• The two surrounding intervals (10016–10020 and 10023-10037) showed elevated errors (avg ≈ 0.002-0.008) which could be construed as spillover from the injected disturbance and tile effect with the temporal windows.

The histogram shown in figure 8 of reconstruction errors post-injection retained the same tight error core but exhibited extended tails from the injected deviation. The same thresholding reasoning was valid since the model relied on distributional deviations from threshold errors opposed to absolute magnitudes.

In several ways these findings provide strong validation of the anomaly detection capabilities of the model:

• The injected fault was detected in every run of the model.

• The highlighted intervals appeared to begin nearly exactly where the injection began (index 10006),

The severity of reconstruction the errors provided a definitive signal to noise distinction.

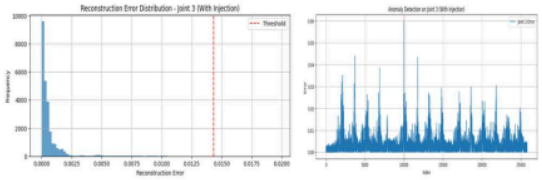

Figure 8: Joint 3 (Anomaly Injected)Output of LSTM

5.2 GRU-BASED ANOMALY DETECTION

GRU Autoencoder was applied on all six robot joints through the experimental configuration utilized by the LSTM model. The emphasis was on determining whether the GRU could detect anomalies and divergence from the behavior of the joints, both normal and under injected fault, (fault having been injected on Joint 3). Interesting to note was that despite the data utilized during experimentation holding no domain-defined critical faults, (none of the joints having reached known industry boundaries correlated to safety limits), the models' predictions from a reconstruction standpoint would allow for discoverability towards behavioral drift or temporal motion anomalies over a duration.

Joint 1

There were eight distinct intervals of Joint 1 shown in figure 9 anomalies discovered with the GRU. The error reconstruction histogram was left-skewed and the majority of the error values were between zero and a small tail. Below the 99.9 percentile threshold, the threshold could be formulated from around 0.007.

Intervals which contained 1389-1389 and 12554-12554 contained maximum reconstruction errors of 0.02171 and 0.01902 respectively--significantly higher than the background reconstruction noise which was below 0.002 for most of the dataset. These findings clarified that GRU was indeed selecting out statstially abnormal behaviors even when absolute devation was inside domain boundaries of safety.

Worth noting is that the anomalies pointed out tended to be semi-arid for the most part throughout the sequence--short-term departures and not long stretches of inaccuracy.

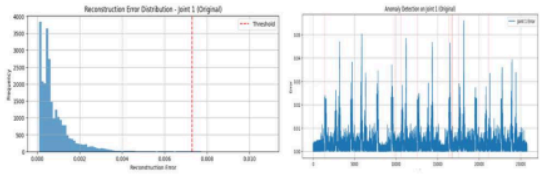

Figure 9: Joint 1 Output of GRU

Joint 2

Joint 2 shown in figure 10 displayed more complicated behavior. GRU identified 7 intervals of anomaly behavior with some very high intensity spikes:

- Interval 5188 - 5189: max error = 0.02989
- Interval 19212 - 19213: max error = 0.01638

The errors were long-tailed, and the histogram plot was specialized enough to reveal a cutoff point at about 0.006, and the cutoff was also the detection threshold. This also indicates the model can ignore minor fluctuations but demonstrate an extreme event at the same time.

Compared to LSTM's narrower detection windows, the GRU model displayed wider extent of sustained deviation over the intervals of the motion behaviors comprehensive of a large perspective form of model of mid-term context change.

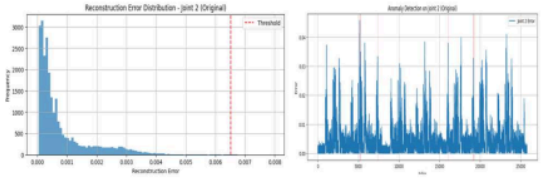

Figure 10: Joint 2 Output of GRU

Joint 3 (Before Injection)

Prior to the injection shown in figure 11, the GRU detected only 1 interval: 10022 - 10047, with max reconstruction error of 0.01134. As the detection occurs just prior to the planned injection flight window, this detection could indicate the model may have calculated or perceived behaviorally unusual motion segments or novel normalized convergent motion segments in advance of detecting the injection. In addition, this instance reinforces that GRU does not always need explicit failure events to identify possible drift or deviation.

The histogram for joint 3 was tightly clustered, though the histogram did not exhibit a tail on the left-side of the plot, there was a noticeable right-side extension near the max threshold of 0.005. The adaptive threshold near 0.014 provided for only high confidence values of deviation as anomalous levels, which should ensure minimal false positive rate.

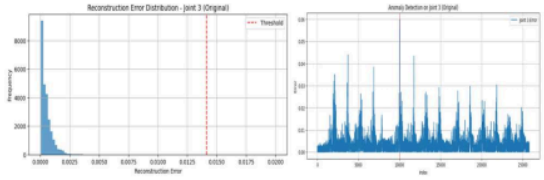

Figure 11: Joint 3 Output of GRU

Joint 4

GRU identified six anomaly intervals in Joint 4 shown in figure 12, all moderate to large deviations:

- Interval 11653–11659: max error = 0.02302
- Interval 11684–11684: max error = 0.00851

The histogram was quite wide, indicating greater variability in normal reconstruction, and the threshold would probably settle around 0.008. GRU picked up more intervals (detecting more anomalies) than LSTM in this joint, and there may have been a slight advantage to GRU in finding anomalies in this joint due to higher sensitivity to changing dynamics, and this could be due to GRU's capabilities for allowing a better capture of shorter, sharper changes due to their design to capture these anomalies.

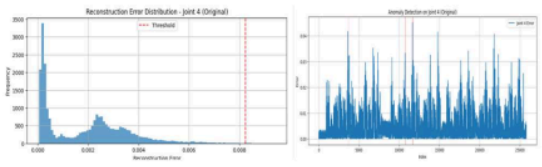

Figure 12: Joint 4 Output of GRU

Joint 5

In Joint 5 shown in figure 13, GRU identified only three intervals, with max reconstruction errors up to 0.01784. While this is less than LSTM, the GRU-flagged anomalies were also occurring in high-error intervals with highly identifiable peaks. The reconstruction error histogram for Joint 5 is more clearly bimodal, with a highly dense cluster at low error (0.0–0.002) and another larger area forming a plateau around 0.004–0.008. The threshold (0.012) would be well-positioned to filter out noise while keeping serious deviations.

This suggests the GRU model may prioritize the variance of the reconstruction error over sensitivity, and may allow more minor deviations that are by clearly identifiable, well-separated errors to pass.

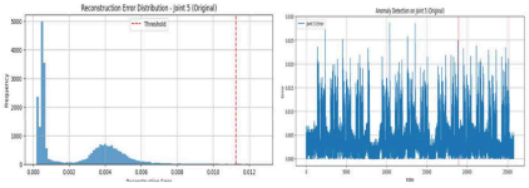

Figure 13: Joint 5 Output of GRU

Joint 6

For Joint 6 shown in figure 14, four anomaly intervals were produced with moderate max errors:

- Interval 16236–16236: max error = 0.00481
- Interval 22900–22900: max error = 0.00337

Histogram analysis of the error indicated that there was a more flat distribution versus the other mains suggesting more uniform reconstruction variability was present in the dataset. Still, the GRU model utilized a consistent threshold (~0.010) and still had an insignificant rate of false detections.

The GRU model for Joint 6 produced consistent anomaly detection with moderate noise and low error amplitude.

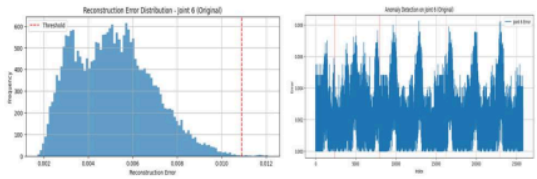

Figure 14: Joint 6 Output of GRU

Synthetic Anomaly Detection (Joint 3 Injection)

A synthetic fault was induced by injecting +0.05 to the error for Joint 3 (between indices 10000 and 10010). The GRU model was retrained using the original dataset and then evaluated against the modified test dataset.

The GRU model successfully flagged the injection in a new flagged interval:

- Interval 10017-10042
- Max error = 0.00948
- Avg error = 0.00364

While the error magnitude shown in figure 15 (for GRU) was less than LSTM's peak of 0.05533, it seems the GRU's anomaly detection window may have been longer and indicating that GRU might have better captured the temporal effects around the injected anomaly.

Another interesting observation was histogram of the error distribution (post-injection) indicated a clear, but contained, tail growing which reaffirmed that GRU had detected the fault, without altering its ability to identify baseline behavior.

In conclusion, the GRU model was confirmed to be a reliable method for monitoring injected faults, and more importantly, to detect an anomaly that was quota of degradation even in an anomaly dataset where no joint exceeded domain-level safety margins.

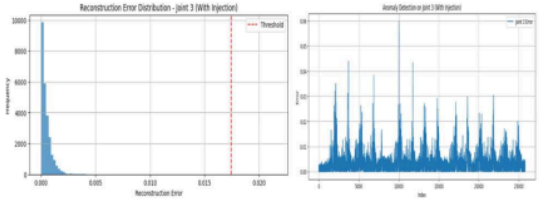

Figure 15: Joint 3 (Anomaly Injected) Output of GRU

5.3 COMPARATIVE EVALUATION AND INSIGHTS

LSTM and GRU models were tested on six robot

joints with the same preprocessing, sequence reshaping and reconstruction-based anomaly detection process. The following comparisons were made on: number of anomalies detected, interval lengths, reconstruction errors, thresholds, training time and synthetic fault anomaly detection.

Anomaly Interval Characteristics

• Total count: LSTM identified a total of 31 anomalous intervals and GRU identified 28. GRU was more sensitive to shorter phase spikes while LSTM preferred fewer anomalies with longer interval or duration.

• Average Length: The average length of an interval identified as abnormal was 8.88 samples for GRU against 5.85 with LSTM; this indicates GRUs better memory of prolonged deviation behavior.

• Maximum Length: GRU 26 samples (same as injected Joint 3 fault), LSTM 18 samples, which corresponds to the tendency of GRU towards identifying longer periods compared to short events.

Reconstruction Error Profiles

• Maximum Error Registered: GRU 0.00315 vs. LSTM 0.00504 (this was measured at the synthetic fault).

• Mean Error: GRU 0.00216 vs. LSTM 0.00329.

While LSTM responded more forcefully to deviations, it would lead to errors in reconstruction of over one percent, which would lead to false negatives if the deviating joint was noisier than any rest.

Thresholding and Noise Resilience

• GRU's mean threshold value was 0.0097, whereas LSTM was 0.0115; this implied that GRU's mean threshold was lower, i.e., GRU was working off a significantly compressed baseline.

• Despite lower thresholds, GRUs model had a high trigger rate avoided, leading to fewer anomalies discovered but longer ones that could be useful to predictive maintenance perhaps due to the noisiness of the environment.

Training Efficiency

• GRU total training time: 1960.20 s; LSTM: 2316.74 s (~15.4% faster)

This is an important gain for larger deployments or more frequent retraining of the model.

Fault injection sensitivity

• Both models detected the synthetic Joint 3 fault.

• LSTM: Higher peak error (0.05533) and three brief intervals around the fault onset.

• GRU: Lower peak error (0.00948) and one continuous interval of error covering the fault and recovery phase.

This indicates LSTM performs well for rapidly occurring spikes, while GRU is better suited to a

level of persistence and recovery over a long time scale.

5.4 SUMMARY AND PRACTICAL IMPLICATIONS

There were no anomalies that corresponded with confirmed failures, and all deviations exhibited persistence within UR5e-safe upper and lower operational limits. Both models learn baseline joint behavior effectively, and stop of the process with statistically significant differences from both baseline measurements. Both models served more as deviation monitors rather than an actual faults classifier. Both models identified faults in a timely manner and their ability to identify faults supports readiness for real deployment.

5.5 DISCUSSION

LSTM provides the best possible onset detection of an anomaly occurring at the earliest time and smallest time scale possible. GRU is the training option of choice when computing resources are at a premium or when a longer drift needs to be characterized. Both models can be used in a hybrid fashion; LSTM detects an immediate reaction to an anomaly and GRU in a sustained monitoring capacity.

6 CONCLUSION

This study created and evaluated LSTM- and GRU-based autoencoder models for the task of anomaly detection of joints from an industrial robot that used real-world time series sensor data. Although none of the joints in the dataset went beyond domain-mechanical thresholds, both models were able to characterize anomalies in behavior—these deviations can serve as early indicators of selective degradation. Among the models evaluated, LSTM was the most sensitive, particularly to identifying short-term transients and clear deviations, making it particularly effective at identifying the onset of anomalies early. GRU was less sensitive than LSTM but is not computationally efficient and can detect sustained anomalies and displayed a significantly lower false positive rate. Both models were able to efficiently identify a contrived fault previously introduced into Joint 3, demonstrating their robustness and reliability when evaluated in simulation, despite the imposed boundary conditions of the real subset of the data.

Compared with existing traditional threshold-based or statistical monitoring methodologies, this deep-learning approach has several operational advantages in an on-line industrial setting:

- It adapts dynamically to normal patterns of behavior
- It identifies subtle drift that may not be identified through rule-based monitoring
- Requires no labeled fault data
- Scales easily through similar or more joints, or multiple operating conditions.

These distinctions make the proposed models compelling to be deployed, monitored, and calibrated in real-time anomaly detection settings. It is envisaged that the proposed models would continuously monitor the state of

joints, identify behavioral changes that are at an early stage of deviation, and safely notify operators prior to reaching critical failure points. This allows operators to plan effective maintenance during downtimes, thus mitigating downtime or loss of production.

While the main findings of this work are encouraging, there are limitations that need to be noted. First, while the dataset included in this work used robotic operational data, it did not have labeled fault data and did not have confirmed mechanical failures, so the models were evaluated using statistical anomaly differentiation rather than for fault-classification or severity differentiation. Second, the models were validated offline and not deployed, and as such, we did not test their real-time performance or integration, and do not know anything about performance, latency or integration in sensor-anomaly detection. Third, we were primarily limited in our analyses to joint position data. We did not utilize additional sensor modalities (torque, vibration, temperature, force), and it is likely that the anomaly differentiation was not comprehensive due to this lack of additional modalities. Lastly, although the synthetic anomaly used to validate the models was useful and revealed signs of there being an anomaly, it was a simplified view of actual mechanical degradation, and did not account for variability or possible complexity in the progression of actual faults in operational settings.

While the results of this research provide evidence of the suitability and usefulness of LSTM- and GRU-based autoencoder models for predictive anomaly detection in robotic joints, there are opportunities to explore and implement these models in practice in several regards.

One approach would be to apply the models in real-time applications within an operational industrial environment. Deploying the system onto robotic controllers or edge computing devices to capture live operation and anomaly detection, alerts, and automated logging of anomalies would be a very exciting development.

Additionally, we could explore the potential longitudinal adaptability of the model by utilizing online learning frameworks or concepts for concept drift detection. To maintain reliability over time, it will be vital to dynamically update the model for potential drift that arises due to mechanical wear on the robots or changes to the load profiles activated in operation.

Moreover, pursuing the development of more features in the input data by utilizing multi-sensor fusion (e.g., torque, force, vibration insights, thermal data) may increase the system's ability to recognize distinct anomalies and better determine the various types of mechanical experiences or electrical experiences invoked within operation.

Future work may also extend the model from merely detecting an anomaly to classifying the potential fault-type and/or predicting the potential severity of the fault. This advancement will allow maintenance teams not only to detect abnormal behavior, but also understand the fault-type and urgency of the experience.

ACKNOWLEDGEMENT

I would like to sincerely thank my supervisor, Milu Philip, for her tireless encouragement, understanding, and feedback that she provided for the duration of this research journey. I would also thank the faculty and staff of the Department of Computing, South East Technological University, for the academic training and resources that allowed me to embark upon this project. I wish to also acknowledge my family and friends for their continued encouragement, motivation, patience, and emotional support over the particularly stressful parts of this dissertation. I would acknowledge with gratitude ChatGPT, which helped me organize content, and simplify technical descriptions as I wrote for the dissertation. Finally, I would like to acknowledge the creators and maintainers of the NIST dataset, which was the primary building block of the experimental work of this dissertation; as well as the research community more broadly, for their input and assistance that shaped this project.

REFERENCES

Ahmed, Y. .. e. a., 2025. Redundant Fault Isolation Using probabilistic models in collaboarative robots.. *Sensors,* pp. 1-18.

Ayala-Chauvin, M. A.-C. F. B. J. &. Y.-A. D., 2024. Predictive Maintenance in Industrial Robotics using Big Data. *IEEE Transactions on Emerging Topics in Computing,* pp. 215-227.

Ayankoso, S. G. F. L. H. F. H. &. B. A., 2024. Artificial Intelligence Based Condition monitoring of industrial collaborative robots. *Machines,* pp. 12(9), 630.

Ball, A. e. a., 2024. Fault detection and remaining life estimation for predictive maintenance. *Procedia Manufacturing,* pp. 38,1241-1248.

Castellanos-Ramos, J. e. a., 2021. Sensor fusion based deep learning approach for robot joint fault detection. *sensors,* pp. 21(8320), 1-15.

Costa, M. A. &. W. B., 2019. Failure detection in robotic arms using hybrid gradient boosing. *Measurement,* pp. 147, 106843.

CSE, S., 2023. MTAD-GAT Based Real Time Anomaly Detection Framework.

Espinoza, j. M. R. &. p. M., 2023. Sensors and actuator fault diagnosis for robot joint systems.

Galan-uribe, e. m.-v. l. &. o.-r. r. a., 2023. FPGA based methodology for detectinig positional accuracy degradation. *Applied Sciences,* p. 13(8493).

Gao, z. W. c. w. j. j. w. &. d. t., 2025. Degradation aware remaining useful life prediction of industrial robots via multiscale temporal memory transformer framework. *Reliability Engineering & systems safety,* pp. 246, 109267.

Graabek, k. A. C. K. &. K. M., 2023. An Experimental comparison of Anomaly detection methods for collaborative robot manipulators.. *Robotics and computer integrated manufacturing,* pp. 81, 102445.

Gu, j. w. x. &. L. x., 2025. A curvature guided shape DTW for anomaly detection. *IEEE sensors journal,* p. 25(13).

Imrana, M. e. a., 2022. Design and development of RNN Anomaly detection model for IOT networks. *IEEE Access,* pp. 10, 62743-62755.

Kang, J. e. a., 2022. Real-Time Failure Anomaly Prediction Based on Model uncertainty. *IEEE IROS.*

Kayan, H. e. a., 2025. Multimodal Real-Time Anomaly Detection in UR3e Cobots on Edge Devices.

Liu, B. e. a., 2021. Spatio-Temporal Attention for predictive maintenance of industrial robots arms. *Journal of Intelligence Manufacturing,* pp. 34(9), 2101-2116.

Liu, Q. e. a., 2022. A dynamic stochastic search Algorithm and Its Application to Feature selection. *Journal of intelligent manufacturing.*

Morettini, S., 2021. Machine Learning in Predictive maintenance of Industrial Robots. *KTH Royal Institute of technology.*

Park, J.-H. S. S. &. C. D. E., 2022. Model-free unsupervised Anomaly Detection using Stacked LSTM. *IEEE IROS.*

Pinto, C. S. R. &. G. J., 2019. Fault Detection and remaining Life estimation for predictive maintenance. *Procedia Manufacturing,* pp. 38, 1241-1248.

Qiao, G. &. W. B. A., 2017. Accuracy Degradation Analysis for industrial robot systems.

Shamuganathan, V. &. S. A., 2023. LSTM-Markov Based Efficient Anomaly Detection Algorithm. *Applied soft computing,* pp. 136, 110054.

song, Y. e. a., 2021. Dual Encoder-Decoder graph Neural Network. *Journal of Intelligent Manufacturing.*

soualhi, M. &. n. K. T. P., 2021. Intelligent Monitoring of Multi-Axis Robots. *Journal of Intelligent manufacturing,* pp. 1743-1759.

ullah, I. &. m. Q. H., 2022. Design of RNN Anomaly Detection Model for IOT networks. *IEEE Access,* pp. 62743-62755.

Yang, B. L. W. Z. Y. X. Z. J. J. &. L. Y., 2024. multivariant Time Series Anomaly Detection in Robot Joint Data. *Journal of Manufacturing systems,* pp. 75,132-149.

Yun, H. &. K. H., 2021. Autoencoder-Based Anomaly Detection of Industrial Robot Arm. *Journal of Intelligent Manufacturing,* pp. 34,1427-1444.

Zhao, S. L. Y. &. C. X., 2023. Model-free Unsupervised Anomaly Detection using stacked LSTM. *IEEE IROS.*

Zhou, Y. e. a., 2025. Redundant Fault Isolation in Collaborative Robots. *Sensors,* pp. 1-18.

C00313519_Gokul_Thillainathan.docx

ORIGINALITY REPORT

1%	1%	1%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.jetir.org Internet Source	<1%
2	Bilal Wehbe, Octavio Arriaga, Mario Michael Krell, Frank Kirchner. "Learning of Multi-Context Models for Autonomous Underwater Vehicles", 2018 IEEE/OES Autonomous Underwater Vehicle Workshop (AUV), 2018 Publication	<1%
3	Al-Tekreeti Watban Khalid Fahmi, Kazem Reza Kashyzadeh, Siamak Ghorbani. "Advancements in Gas Turbine Fault Detection: A Machine Learning Approach Based on the Temporal Convolutional Network–Autoencoder Model", Applied Sciences, 2024 Publication	<1%
4	Xueying Han, Susu Cui, Song Liu, Chen Zhang, Bo Jiang, Zhigang Lu. "Network intrusion detection based on n-gram frequency and time-aware transformer", Computers & Security, 2023 Publication	<1%

5	Submitted to University of Bolton Student Paper	<1%
6	Submitted to University of Hertfordshire Student Paper	<1%
7	www.researchsquare.com Internet Source	<1%
8	www.researchgate.net Internet Source	<1%
9	discovery.researcher.life Internet Source	<1%
10	Yu-Yang Lin, Wan-Jen Huang, Chia-Hung Yeh. "Dual-CycleGANs with Dynamic Guidance for Robust Underwater Image Restoration", Journal of Marine Science and Engineering, 2025 Publication	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		