Vol-3 Issue-1 2025
Scientific Research Journal of Science, Engineering and Technology
ISSN: 2584-0584, Peer Reviewed Journal

Part Inspection Tracking and Forecasting in Machining and Assembly Plants

Raunak Ujawane[1], Balaji Mali[2], Aditya Agarwal[3]

[1]Central Quality Department, Dana Anand India Private Limited, Pune, India, (raunak.ujawane@dana.com)
[2]Axle Plant Head, Dana Anand India Private Limited, Pune, India, (Balaji.mali@dana.com)
[3]Mechanical Engineering Department, Thapar Institute of Engineering & Technology, Patiala, Punjab, India, (aagarwal11_be21@thapar.edu)
Correspondence:
Email id: aagarwal11_be21@thapar.edu

Abstract

Production facilities require regular part inspections to ensure that manufactured components meet design specifications and quality standards. These inspections maintain quality and generate valuable data about production patterns, such as peak and low periods, shift-wise performance, and potential bottlenecks. This paper presents a digital Part Inspection Tracking & Forecasting system integrated with a cloud database for real-time recording of part inspection statuses (categorized as OK, NOT OK, or Pending). In parallel, we develop a time-series forecasting module using a Recurrent Neural Network (RNN) model called the Long Short-Term Memory (LSTM) model. By leveraging the historical inspection data captured through the system, the LSTM model forecasts the number of parts expected for inspection over a future horizon. By facilitating data-driven decision-making, the integrated strategy provides a real-time dashboard for forecast visualization and data entry, supporting Industry 4.0 goals. With the aid of this useful technology, plants will be better equipped to predict and prepare for future demands. Many areas like warehousing, transportation, and dispatch will be eased by optimizing resource allocation based on predicted inspection volumes. In this paper, we have implemented all these relevant tools and some quality-of-life improvements to make our research and software invaluable for manufacturing and assembly plants. We describe the system architecture (including a Python Tkinter-based user interface and cloud backend), the data processing and model training approach, and experimental results on an automotive manufacturing dataset. Our findings show that the LSTM-based model outperforms conventional techniques in identifying both short-term trends and long-term seasonal patterns, producing accurate short-term projections of inspection load. This makes it possible for stakeholders to better allocate resources and predict workloads. We achieved RMSE values of less than 1 (error is the number of parts predicted vs. actual), indicating high correlation and accuracy.

Keywords: *Time Series Forecasting, Long Short-Term Memory, ARIMA, LSTM, Industry 4.0*

1 Introduction

Planning and system-level optimization may be greatly improved in any manufacturing facility by precisely anticipating production-related activities. For instance, predicting the volume of parts arriving for quality inspection in advance can help allocate inspectors, schedule maintenance, and smooth out workflows. This study is among the first to apply LSTM forecasting to part inspection workflows, enhancing operational efficiency in machining plants. To get it running, one of the biggest hurdles is that many plants have relied on manual entry (e.g., paper forms or spreadsheets) for part inspection records, which is slow and prone to human error. With the industry-wide transition to Industry 4.0, there is a strong push to digitize such processes and leverage real-time data for intelligence [1][2]. Quality 4.0 integrates traditional quality management with digital technologies, emphasizing real-time data collection and analysis to enable intelligent decision-making [3]. We can guarantee that information is immediately accessible to decision-makers and related systems by digitally collecting inspection data at the source and storing it on a cloud platform. This will serve as the foundation for more complex analytics, such as forecasting.

Advanced predictive models, such as LSTM, perform better than conventional techniques in addressing these issues. Time-series forecasting in manufacturing has numerous applications, including demand forecasting, predictive maintenance, and workload planning [4]. Recurrent Neural Networks (RNNs), especially LSTM networks, have gained popularity for these tasks due to their ability to learn temporal patterns and long-term dependencies in sequence data [4]. Unlike classical models such as ARIMA, which have been widely used in industrial time-series data, LSTMs can capture recent trends and periodic behaviors over longer horizons [4]. This makes them particularly effective in contexts where production volumes exhibit complex patterns (e.g., daily cycles, weekly seasonality, and irregular spikes due to shift changes or maintenance events).

This paper proposes a cloud-connected Part Inspection Tracker system with an integrated LSTM-based forecasting module. The tracking system provides a user-friendly GUI for shopfloor inspectors to log each part inspection (including part identifier, timestamp, and inspection result status). The data is transmitted to a central cloud server in real time, ensuring a unified, up-to-date dataset. We apply an LSTM model to this data to predict the total number of parts requiring inspection in upcoming time intervals (e.g., the next 24 hours). The GUI visualizes the predictions, giving plant personnel a forward-looking view of the expected inspection load. Such foresight is an asset for logistics and production planning, as it allows proactive adjustments to staffing or operations.

2 Methods & Resources

2.1 Literature Survey

To ground our approach in the context of existing research, we survey three key areas: (A) classical time-series forecasting methods versus modern deep learning in manufacturing contexts, (B) optimization and tuning of LSTM models for sequence prediction, and (C)

applications of LSTM in domains analogous to part inspection (such as production volume and energy output forecasting).

2.1.1 ARIMA vs. LSTM for Industrial Time-Series Forecasting

Autoregressive Integrated Moving Average (ARIMA) models are really popular and widely used for time-series forecasting, as they are simple, computationally less intensive, and statistically grounded. ARIMA models operate under the assumption that future values can be predicted from a linear combination of past values and past errors, with parameters to capture trends (integration) and seasonality [5]. ARIMA and its variants have been widely used in manufacturing and production for tasks like inventory demand forecasting or machine sensor readings, especially when data shows stable seasonal patterns. However, ARIMA models have limitations: they are essentially linear and struggle with complex nonlinear patterns or shifting regimes in the data [5].

Researchers have investigated whether LSTMs can outperform ARIMA in forecasting tasks. **Siami-Namini et al.** compared ARIMA and LSTM on various economic and financial time series and found that LSTM-based models achieved an **average 84–87% reduction in forecast error (RMSE)** compared to ARIMA [5] across various datasets. This remarkable performance gap underscores LSTM's ability to model complex patterns that ARIMA misses, even when only past values of the series are used. Similar trends have been observed in industrial datasets: Punia et al. report that a deep LSTM yielded significantly better demand forecasts in a multi-channel retail scenario than traditional statistical models [4]. These findings motivate our choice of LSTM for forecasting part inspection volumes.

It is worth noting that LSTMs outperform ARIMA in accuracy, but they come with significantly higher computational cost and complexity. Training a deep learning model usually requires more data and processing power. In our project, we have trained the model using Google Collab's robust hardware infrastructure and then exported it to be used locally. Once a model is trained, the processing is relatively less time-consuming, making it practically useable for shop floors.

2.1.2 Optimizing LSTM Models for Time-Series Prediction

While conceptually, it is easy to understand why we are using LSTM, for most use cases, it requires us to make several design choices for which there is no precise guideline to help decision-making. Some key things to configure include network architecture (number of layers and hidden units), training parameters (learning rate, epochs, batch size), and how the sequential input data is windowed and fed to the network. Even here, it is generally understood that things like higher epochs are better but come at huge computational expense. So, striking the right balance between accuracy and computational cost is critical. Tuning these hyperparameters is often crucial for good performance. Prior research provides some guidance. Yadav et al. (2020) optimized an LSTM for predicting Indian stock market trends by comparing **stateless vs. stateful** training modes and adjusting network depth [6]. Based on their research, we chose a stateless machine since our data is continuous and uses a sliding window of past time steps for simplicity (the network resets the state for each prediction cycle).

Hyperparameter selection for LSTMs often remains a process of trial and error, but some studies offer heuristics. For example, a small number of epochs can sometimes suffice if early stopping is used, as increasing epochs beyond a point may yield diminishing returns or overfitting [5]. We adopt a moderate epoch count and use validation performance to decide when to stop training (to avoid overfitting).

2.1.3 LSTM Applications in Production and Quality Prediction

Our use-case—forecasting part inspection counts in an automotive machining and assembly plant—shares characteristics with other industrial forecasting problems. One analogous domain is **predictive maintenance and anomaly detection** on production lines. For instance, Wahid et al. developed a hybrid CNN-LSTM model to predict machine failures in an Industry 4.0 setting, using LSTM layers to capture temporal patterns in sensor data [2]. Their system could forecast equipment failures by analyzing sequences of vibration and pressure readings, demonstrating how sequence models like LSTM can provide early warning of manufacturing quality or maintenance issues. Another related area is **product quality prediction**. Bai et al. (2021) proposed an ensemble model combining AdaBoost with LSTM to predict manufacturing quality metrics [7]. Training on historical process parameters and quality inspection results improved their hybrid model's prediction accuracy for whether a given process run would produce an acceptable part. LSTM networks have been employed in real-time quality control scenarios to predict the quality grade of products from sensor streams, allowing immediate interventions when the model predicts an out-of-tolerance condition [4].

Another interesting comparison comes from outside manufacturing: the energy sector. In **renewable energy production forecasting** (e.g., solar power output), the data can be highly volatile daily with larger seasonal patterns, much like how a factory's production might fluctuate daily and weekly. A solar photovoltaic power forecasting case study found that LSTM models outperformed conventional techniques because they could capture short-term weather-induced fluctuations and longer seasonal trends.[10] The study also stressed the need for data cleaning (removing outliers) and proper evaluation metrics like mean absolute error and R^2 score to assess model performance. We took inspiration from this when designing our data preprocessing. We chose **Root Mean Square Error (RMSE)** as our evaluation metric for the part inspection forecast, as discussed later in the Results section. While prior work has applied LSTMs to demand forecasting, their use in part inspection tracking remains underexplored.

2.2 Proposed Model

2.2.1 System Architecture

Our proposed approach consists of two main components: **(1) a Part Inspection Tracking UI** and **(2) a Forecasting Engine**. The part inspection stations in the plant are equipped with computers running a Python-based **Tkinter graphical user interface (GUI)** and a forecasting engine leveraging a pre-trained LSTM model. When a part is ready for inspection, the inspector uses the GUI to enter the details like its line, machine, part number, operation, and activity and mark its inspection status as OK, NOT OK, or Pending. Each entry is timestamped automatically. The GUI communicates with a cloud server directly on the plant's intranet. We implemented the backend as a centralized CSV database. When an entry is submitted, it is stored

in the cloud database in real-time. This ensures that data from all CMM rooms across the plants are aggregated into a single master sheet accessible by all stakeholders. This real-time data availability is critical for **Industry 4.0** integration and enables immediate analytics of the collected information.

Alongside the data entry interface, the GUI includes a **Forecasting Screen**. This screen allows a user (e.g., a supervisor) to select a time window or click "Forecast" for the system to retrieve the latest data from the cloud and generate a forecast for the upcoming period (for example, the next 24 hours divided into half-hour intervals). The forecasting engine is implemented in Python using the trained LSTM model. When triggered, it pulls recent historical data (e.g., the past 14 days of inspections) from the cloud database, feeds this data through the LSTM model to predict the next 48-time steps (which correspond to 24 hours if using 30-minute intervals), and then displays the forecast results on the GUI. The output is presented in numeric form (e.g., expected count of OK/NOT OK/Pending parts per interval) and as a line chart for easy visualization of trends. By having the forecast accessible in the same interface as the data entry, we provide a seamless experience where operators record current information and gain insights into future workloads. This integration exemplifies a cyber-physical system approach: the physical act of inspections immediately updates the cyber (digital) model, which influences physical-world decisions like resource allocation.

The model's training was done separately via Google Collab to speed up the training process and avoid unnecessary code in our user-faced application. At the time of testing, we experimented with various models despite having a good idea. This is done to ensure that all the popular models are tested, and we chose statistically the best model. As discussed in the next section, we have split the original data into industry-standard 70/30 train/test splits, and we will compare actual and forecasted values for the test data. Models tested include:

- **Prophet:** Developed by Facebook, Prophet is an additive model that automatically handles trends, seasonality (daily, weekly, yearly), and holiday effects. It is robust in detecting missing data and outliers and is especially useful for business time series that exhibit multiple seasonalities.
- **Holt-Winters Exponential Smoothing:** Holt-Winters is also known as Triple Exponential Smoothing, which captures level, trend, and seasonal components in time series data through smoothing. It's straightforward, interpretable, and practical for series with strong seasonal patterns, though it struggles to capture nonlinearities.
- **ARIMA:** A classical time series model that decomposes a series into autoregressive (AR) and moving average (MA) components after differencing to achieve stationarity. It works well for linear time series with consistent trends and noise but may struggle with complex seasonal patterns.
- **LSTM:** The industry standard LSTM model was used with Drop Out and Dense layers to create a robust network capable of understanding and forecasting the parts entering the plant.
- **Bidirectional LSTM:** A deep learning model that uses Long Short-Term Memory units in a bidirectional setup—processing forward and backward data—to capture long-term

dependencies in the sequence. This architecture is beneficial when context from both past and future (relative to a time step) improves predictions.

Table 1: Model Performance

Model	RMSE Error Values
LSTM	0.069
ARIMA	3.20
Prophet	2.80
Holt-Winters	2.75
LSTM Bidirectional	0.08

LSTM Forecasting Model

As seen from the last table, LSTM is giving incredible results on the validation dataset. Due to the good results, we move forward with our LSTM. Now, let's discuss the structure and overall architecture of LSTM.

General Architecture of LSTM

An LSTM network is built from units called cells, each of which processes a data sequence one step at a time. What distinguishes LSTM cells are their internal components: specifically, each LSTM cell contains three interactive gates - **input gate**, **forget gate** and **output gate**. The input gate governs the flow of new data to memory at the time step. The forget gate modulates how much of the past information in the cell state to retain or erase. This allows the cell to drop information that is no longer relevant to future predictions. The output gate determines how much of the cell's internal state to expose to the next layer or the next time step (as the cell's output). An LSTM can maintain a memory of important patterns over long sequences through this gating mechanism while discarding noise or less useful information. This capability is essential for our problem: for example, the LSTM can learn that daily patterns repeat every 48 half-hour intervals or that there is often a surge on Monday after a weekend (with little production). It can remember these longer-term patterns via the cell state, even as it processes individual time steps. Our LSTM model prediction step is fast (on the order of milliseconds per time step) since it runs on a pre-trained network with a relatively small size. We ensured that the model file was loaded into memory when the application started, avoiding any delay in loading during each forecast request.

Our LSTM model takes as input a sequence of past part counts (with features corresponding to each status category). We frame the forecasting as a time-series problem with overall parts entering each time interval. At each time step, these four values are fed into the LSTM. We use a **window size** of past T steps (e.g., T = 48 for the past 24 hours) as the input sequence, and the model is trained to predict the next value of the target series. For simplicity, our current model focuses on forecasting the Overall count of parts in the next step (since that is a direct indicator of workload). We found that the model can implicitly consider the composition of OK/NOT OK in the past when forecasting the total volume. In future iterations, we could extend the model to

predict each category separately, but the total count was the primary interest for our stakeholders as a proxy for production rate.

The network architecture consists of:

Table 2: LSTM Structure

Layer	Type	Units	Output Shape
LSTM	Recurrent	100 units	(INPUT_WINDOW, 100)
Dropout	Regularization	20%	(INPUT_WINDOW, 100)
LSTM	Recurrent	50 units	(50,)
Dropout	Regularization	20%	(50,)
Dense	Fully Connected	OUTPUT_WINDOW * length(features)	(OUTPUT_WINDOW * length features),)
Reshape	Reshape	-	(OUTPUT_WINDOW, length (features))

In our experiments, a single LSTM layer with an adequate number of hidden units could capture the necessary dynamics. Adding a second LSTM layer (stacking them) or combining it with a Convolutional Neural Network (CNN) layer did not significantly improve our data. However, CNN-LSTM hybrids have been noted in the literature to enhance feature extraction for very complex signals [2]. Therefore, we opted for a simpler model to reduce training time and avoid overfitting, given the dataset size. The output of the LSTM layer goes through a dense (fully connected) layer to produce the final numeric prediction for the next time step.

We trained the LSTM using the **Adam optimizer** (Adaptive Moment Estimation) with a mean squared error loss function corresponding to minimizing RMSE. The choice of Adam is motivated by its popularity and effectiveness in training RNNs, as it can adjust the learning rate adaptively during training. We initially experimented with a standard stochastic gradient descent optimizer, but Adam converged faster and yielded slightly better accuracy in validation. The training was run for 80 epochs on the training set. We used an early stopping criterion monitoring validation loss to prevent over-training. As mentioned earlier, normalization of inputs was applied, and the outputs were correspondingly scaled back to original counts for interpretation.

We employed a simple rolling prediction strategy to make the model forecasts more robust. Rather than training to predict many steps in one go (which can accumulate errors), our LSTM is trained to expect just one step ahead. For forecasting multiple steps (say 48 steps for a full day), we use the model recursively: predict one step, append that prediction to the input sequence (and remove the oldest step), then expect the next, and so on. This is sometimes known as iterative multi-step forecasting. Although errors can compound in this approach, it often performs well if the model is strong and if predictions are fed back cautiously. Some advanced strategies, such as the usage of separate models for different horizons or using encoder-decoder LSTMs with an

attention mechanism, could further improve multi-step forecasts [4], but those add complexity. Our chosen approach balanced simplicity and accuracy for the given forecasting horizon.

Training and Workbench Configuration: We trained the LSTM model using Python 3.8 with libraries such as TensorFlow/Keras for the neural network implementation. The training was conducted on a workstation with the following specifications:

- **System:** 64-bit OS, x64-based processor
- **CPU:** Intel(R) Xeon(R) CPU @ 2.00GHz
- **RAM:** 16 GB
- **GPU:** Tesla T4 GPU
- **Storage:** 112 GB

(The above hardware setup was used to expedite the training process. The trained model is lightweight enough to run on typical shop floor PCs for inference.)

On this workbench, training the model ran for 80 epochs, 64 batch sizes, and the largest model took around 2 hours. This training is generally a one-time or infrequent process; the model can be retrained periodically (for example, monthly or when significant new data has accumulated) to ensure the forecasts stay accurate as production patterns evolve. Because training can be time-consuming, one could offload it to a cloud server or a more powerful machine and then deploy only the trained model weights to the plant systems. Some of the key actions available in the GUI include:

1. **Data Entry:** Parts arriving at inspection are logged via the UI, which instantly sends the data to the cloud database.
2. **Data Storage:** The cloud database records each entry (timestamp, part ID, status, etc.). This builds the time-series dataset needed for forecasting.
3. **Trigger Forecast:** A user triggers the forecast on the UI. The system fetches recent historical data (e.g., last 2 weeks) from the database as input for the model.
4. **Forecasting:** The LSTM model (locally or on the server) processes the input sequence and generates the forecast for the specified horizon (e.g., the next 24 hours).
5. **Result Presentation:** The forecast results are returned to the UI and displayed as a graph and table. The user can interpret these results and take actions (e.g., schedule additional inspectors for expected high load periods or adjust production if a low output is forecasted).

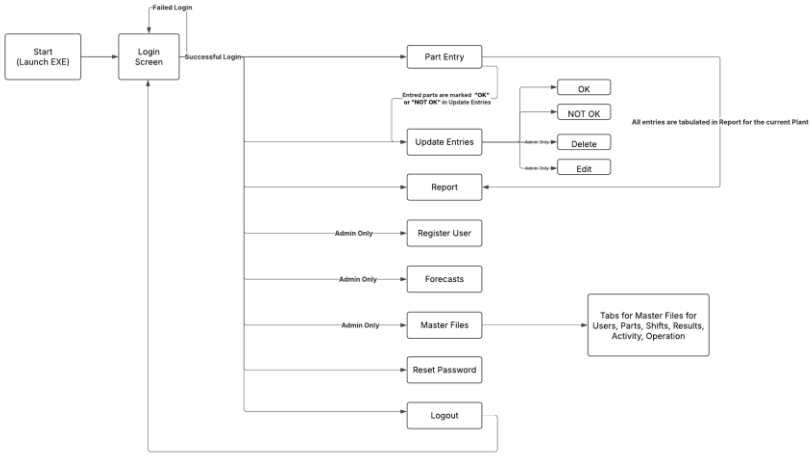

Figure 1: Flow chart of the GUI usage

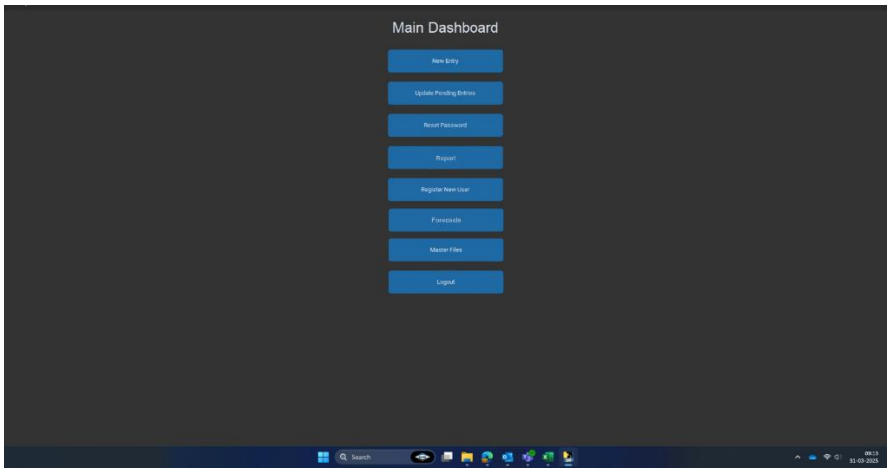

Figure 2: GUI

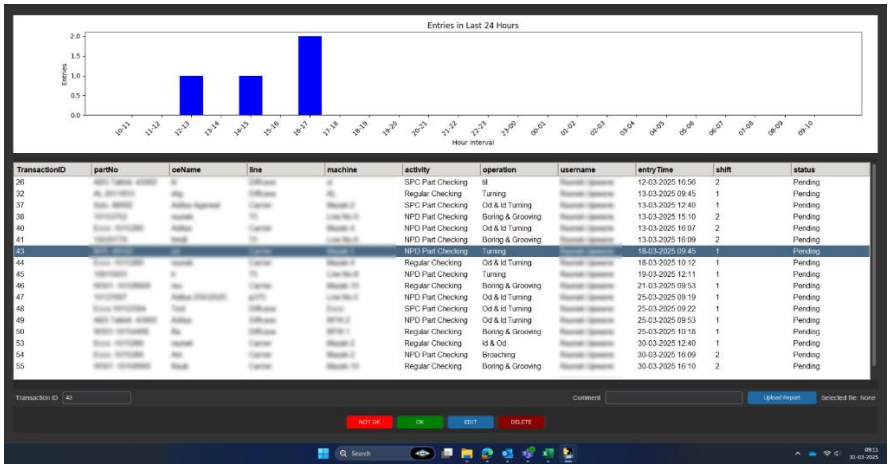

Figure 3: Data Update GUI

Each step operates in a feedback loop: These forecasts are refreshed frequently to align with the data being input. By utilizing a cloud server, consistency is maintained even when several UI instances are operating, as they all access and contribute to the same data source.

2.3 Dataset Description

Using a six-month dataset from a car OEM's machining and assembly facility, we created and evaluated our model. The raw data logged every part entering the inspection area, with each record containing a timestamp, part/batch ID, inspector/machine ID, and inspection status (OK, NOT OK, or Pending). To enable time-series forecasting, we aggregated this event log into 30-minute intervals, summing OK, NOT OK, Pending, and total (Overall) counts per interval based on domain knowledge aligning with shift schedules. This produced a continuous time-series dataset (~8700 intervals) with columns for Time, OK_count, NOT_OK_count, Pending_count, and Overall_count.

We split the data chronologically into 70% (~4.2 months, 6090 intervals) for training and 30% (~1.8 months, 2610 intervals) for testing—to preserve order and avoid lookahead bias. **Focusing on overall counts for this paper, we normalized all counts to a range of [0, 1]** using training set maxima. The data revealed daily cycles (peaks during production, troughs at breaks), weekly seasonality (lower night/weekend activity), and occasional surges from special orders or downtime recovery, which we retained to train the model on real-world anomalies.

For LSTM input, we preprocessed the data (removing duplicates, fixing incomplete records) and created sequences: each time step t used a window $[t-T, t-1]$ (T=48, or 24 hours) to predict the Overall count at t. The final dataset was stored as a CSV for 6 months, with 30-minute intervals, four count features, and ~8700 data points. In deployment, the model queries the database directly for real-time data.

Summary of Final Dataset:

- **Period:** 6 months of continuous operation
- **Interval:** 30-minute aggregated counts
- **Features per interval:** OK, NOT_OK, Pending, Overall counts
- **Total data points:** ~8700 intervals
- **Training set:** ~6090 intervals (first 4.2 months)
- **Test set:** ~2610 intervals (last 1.8 months)
- **Scaling:** Min-max normalized (0 to 1) on the training data range
- **Sequence length (LSTM input window):** 48
- **Forecast horizon:** up to 48 steps ahead (24 hours) for evaluation

The prepared dataset was stored in CSV format for convenience during development. In deployment, the model directly queries the database for the latest chunk of data rather than reading from a static CSV.

3 Results

We measure accuracy using **RMSE (Root Mean Square Error)** computed between the predicted and actual part counts in each time interval over the test period. As shown in Table 1, the RMSE value received for LSTM was 0.069. For context, we also trained various other models on our dataset and found LSTM to be the best still. ARIMA gives a RMSE of 3.2, making it substantially inferior to LSTM. Furthermore, ARIMA's graph shows a flatline since it cannot understand the pattern.

3.1 Figures

Below are figures with actual data for a period and the corresponding forecasted data. Most models were unable to understand the periodic patterns. For example, on shift change (for example, 2:30 pm daily), there is a regular upward trend in incoming parts, which results in a peak at around 3 pm. LSTM quickly understood and adjusted to the pattern, but ARIMA and other models couldn't match it closely.

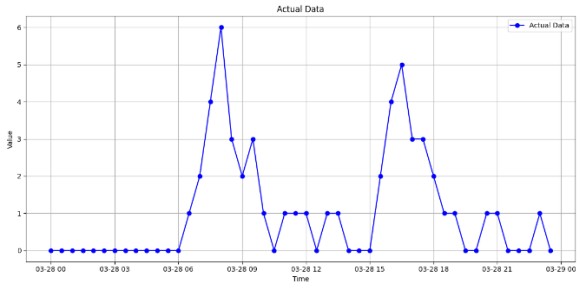

Figure 4: Actual Data

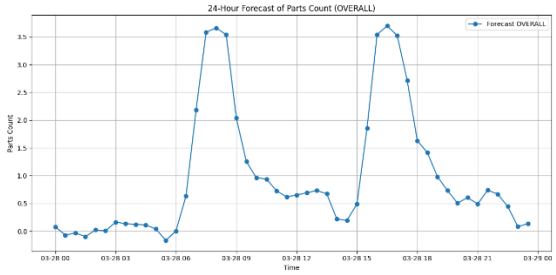

Figure 5: Forecasts from LTSM

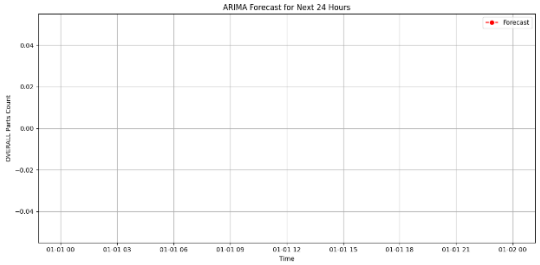

Figure 6: Forecasts from ARIMA (flat lined at 0)

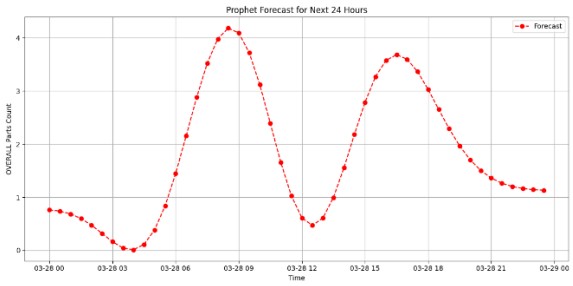

Figure 7: Forecasts from Prophet

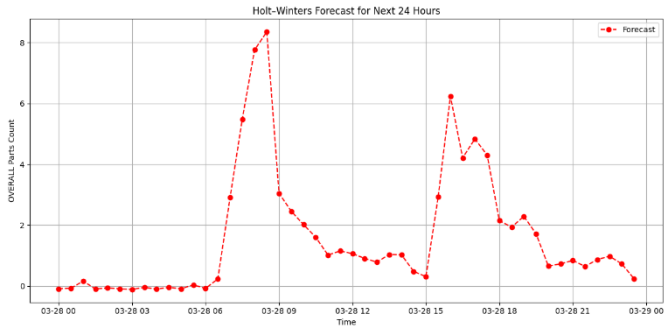

Figure 8: Forecasts from Holt-Winters

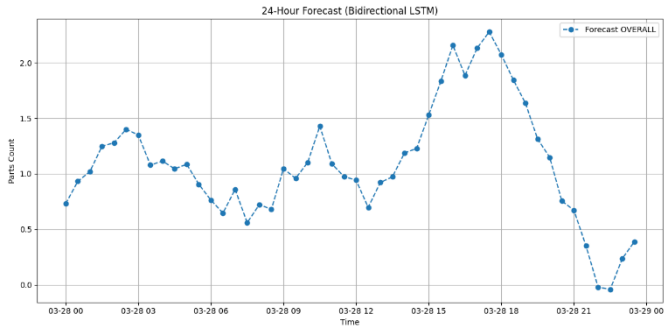

Figure 9: Bidirectional LSTM Forecast

We can see that the predicted curve follows the actual curve for both the LSTM models, Prophet and Holt-Winters. The prophet was a very smooth line, indicating that it had generalized the forecasting too much, resulting in it being unable to understand the noise. Holt-Winters is the closest, but it looks to be overcorrecting in nature. The peaks are higher than the actual data, while valleys are generally lower than the actual data.

Error Analysis: Inspecting the residuals (prediction minus actual), we did not find obvious autocorrelation remaining, which suggests the LSTM has extracted the patterns well. The residuals had a roughly zero mean (with a slight positive bias, meaning a tiny tendency to underpredict high values) and no strong periodic structure. We observed that most larger errors corresponded to intervals with unusual events like machine breakdowns or rush orders, which the

model had no direct way to foresee. This points to a limitation: our model is purely time-series based on past counts. In the future, integrating additional inputs (a common approach in Industry 4.0 implementations), such as machine status or planned production schedule, could help the model know about forthcoming changes and improve accuracy further [9].

Forecast Utility and Use Cases

Beyond numerical accuracy, a key question is whether the forecasts provided by our system are helpful for decision-making in the plant. We gathered informal feedback from a pilot trial where the system was deployed in a test environment with historical data replay. Supervisors noted that having a visual forecast of the next day's inspections helped them in a few ways:

- **Resource Allocation:** If a high volume of inspections is predicted for the upcoming shift, they can preemptively assign additional inspectors or extend the working hours of existing ones. Conversely, if a very low volume is expected (perhaps due to an anticipated lull or just after a maintenance shutdown), they might temporarily reassign inspection staff to other tasks. This adjustment is easier to do proactively rather than reacting at the moment. [1]
- **Production Planning:** In some cases, the forecasted inspection count can be a proxy for production output (since every produced part should pass through inspection). A significant dip in the expected inspections might flag an upstream issue (like a machine-scheduled maintenance or material shortage). By noticing this in the forecast, production planners can double-check if everything is in order or if any schedule changes haven't been communicated. It acts as a consistency check between production plans and likely outcomes.
- **Shift Handover Reports:** The team found it helpful to include the forecasted vs actual numbers in daily shift handover meetings. For instance, if the model consistently underestimates the night shift output by a little, that might indicate a systematic pattern change that the model hasn't learned yet (and it signals that perhaps retraining the model or updating parameters might be needed soon or simply that production is trending upward). Having this information facilitates discussions on factors causing deviations.

We also tested the forecasting scenario at different horizons. Although our primary focus was the 24-hour ahead forecast (with half-hour resolution), the system can be configured for shorter-term predictions (e.g., next 8 hours in finer 15-minute intervals) or longer-term (multi-day). With the current model, extending beyond 24 hours by iterative predictions leads to rapidly growing uncertainty. We found that a 24–48-hour window is a sweet spot where the model's predictions are still reasonably reliable; beyond that, the lack of any exogenous inputs means the model reverts to repeating average daily patterns.

Comparison with Literature Benchmarks: Our results align with what other researchers have documented in similar contexts. For example, an LSTM used for predicting the quality outcomes in a semiconductor manufacturing process achieved an MAE that was about 15% of the actual range of the quality metric [10], and our MAE for part counts is comparable to the typical range of variation. Also, the advantage of LSTM over ARIMA in our case reflects the ~85% error reduction reported by Siami-Namini et al. for financial data [5], lending credence to the notion

that LSTM's strengths also translate to industrial domains. Overall, the results validate that our Part Inspection Tracker with LSTM forecasting is effective.

4 Conclusion

In this paper, we have demonstrated a comprehensive approach to modernizing and optimizing the part inspection process in a manufacturing plant through digital tracking and AI-driven forecasting. The proposed system replaces manual record-keeping with a networked application that logs inspection results to a cloud database in real-time, aligning with Industry 4.0 paradigms of interconnectivity and data availability. Building upon this robust data pipeline, we trained a Long Short-Term Memory model to forecast future part inspection volumes. The LSTM model proved capable of learning the complex temporal patterns of production and inspection data, yielding highly accurate forecasts (with substantial error reductions compared to traditional ARIMA benchmarks) and operationally sound.

Our results show that even short-term forecasts (on the order of hours to a day ahead) can provide significant value in a machining and assembly context. Plant managers can take preemptive measures like redistributing the personnel, balancing production lines, or planning maintenance at the right times by understanding the anticipated inspection load ahead of time. Because resources may be matched to guarantee that every item receives the appropriate attention during inspection, this leads to increased productivity and perhaps superior quality. The ability to anticipate peaks in inspection also means critical quality issues (if they correlate with volume surges) could be identified and addressed more promptly.

The LSTM model's success in capturing daily and weekly seasonality and its adaptability to new data underscores the importance of bringing machine learning into manufacturing operations. It reinforces findings from broader Industry 4.0 research that data-driven methods, when applied to the rich streams of production data, can unlock efficiency gains and previously inaccessible insights [1][4]. Our work contributes explicitly to the relatively niche but essential problem of part inspection forecasting – a link between production output and quality control that historically hasn't seen much automation beyond basic SPC (Statistical Process Control) charts.

There are several avenues for future work and enhancements. First, we plan to incorporate **feedback mechanisms** where the model can be retrained or fine-tuned on the fly as new data comes in (online learning). This would help the system remain accurate over time, especially if there are gradual changes in production rates or shifts in working patterns (e.g., adding a new shift or changes in inspection procedures). Our second goal is to improve the input properties of the model. Only historical counts of examined components are being used. The projections might be strengthened, particularly for longer time horizons, by using inputs such as the projected production schedule, machine uptime statistics, or even environmental conditions, which can occasionally impact fault rates. This moves towards a **multivariate time-series forecasting** framework beyond just the series' history.

From a deployment perspective, one future enhancement is integrating the system with enterprise software (like MES – Manufacturing Execution Systems or quality management systems) so that the insights flow upstream and downstream automatically. For instance, if the forecast predicts a

potential overload, the MES could alert maintenance to ensure all inspection tools are functional or alert procurement if a particular part type seems to be failing more (if we extend forecasting to each part type's OK/NOT OK counts).

In conclusion, the Part Inspection Tracker and Forecasting system exemplifies how adopting digital tools and AI in a traditional manufacturing setting can yield **smart factory** benefits. We demonstrated that our in-house LSTM model, trained on real production data, can reliably forecast production-related metrics (inspection counts) and thus act as a surrogate for production forecasting and quality monitoring. This contributes to improved transparency and agility in operations. As manufacturing plants continue to embrace digital transformation, we expect that such approaches will become increasingly common, turning factories into environments where data from every process—machining, assembly, or inspection—flows into predictive models that drive informed, timely decisions. The ultimate vision is a brilliant production line where resources are optimally orchestrated through predictive analytics, minimizing downtime and ensuring quality in a manner that outperforms even the most experienced human planners [2].

5 Conflict of Interest

The authors affirm that the research was conducted without any financial or commercial relationships that could be interpreted as a potential conflict of interest.

6 Author Contributions

Mr. Aditya Agarwal conceptualized the approach, trained/tested the models, and wrote the code. Mr. Raunak Ujawane supervised the write-ups and helped in dataset building; Mr. Balaji Mali is the 2nd supervisor and contributed immensely to the methodology and problem statement definition.

7 Funding

No funding was received for the research work.

8 Acknowledgments

The authors thank Thapar Institute of Engineering & Technology (TIET) for granting access to Google Collab for training, and testing the machine learning models used in this study.

9 Data Availability Statement

The data used to study, train, and test our work is provided only upon reasonable request from the corresponding author. Feel free to contact the corresponding author for any guidance needed to reproduce the setup, implementation, or any other use case.

10 References

1. N. Kashpruk, C. Piskor-Ignatowicz, and J. Baranowski, "Time Series Prediction in Industry 4.0: A Comprehensive Review and Prospects for Future Advancements," Applied Sciences, vol. 13, no. 22, p. 12374, 2023
2. Wahid, J. G. Breslin, and M. A. Intizar, "Prediction of Machine Failure in Industry 4.0: A Hybrid CNN-LSTM Framework," Applied Sciences, vol. 12, no. 9, p. 4221, 2022
3. A. Escobar, M. E. McGovern, and R. Morales-Menendez, "Quality 4.0: a review of big data challenges in manufacturing," Journal of Intelligent Manufacturing, vol. 33, no. 8, pp. 2307–2322, 2022
4. S. S. Wishal and A. Rahimi, "A Review of Time-Series Forecasting Algorithms for Industrial Manufacturing Systems," Machines, vol. 12, no. 6, p. 380, 2022
5. S. Siami-Namini, N. Tavakoli, and A. Siami Namin, "A Comparison of ARIMA and LSTM in Forecasting Time Series," in Proc. 17th IEEE International Conference on Machine Learning and Applications (ICMLA), Orlando, FL, USA, 2018, pp. 1394–140.
6. S. Yadav et al., "Optimizing LSTM for time series prediction in Indian stock market," 2020.
7. Y. Bai, J. Xie, D. Wang, W. Zhang, and C. Li, "A manufacturing quality prediction model based on AdaBoost-LSTM with rough knowledge," Computers & Industrial Engineering, vol. 155, 2021, Art. 107227.
8. Yuchen Jiang, Pengwen Dai, Pengcheng Fang, Ray Y. Zhong, Xiaoli Zhao, Xiaochun Cao, "A2-LSTM for predictive maintenance of industrial equipment based on machine learning", Computers & Industrial Engineering, Volume 172, Part A, 2022, 108560, ISSN 0360-8352.
9. Hector I, Panjanathan R. Predictive maintenance in Industry 4.0: a survey of planning models and machine learning techniques. PeerJ Comput Sci. 2024 May 14;10:e2016. PMID: 38855197; PMCID: PMC11157603.
10. G. Z. Wei, X. Zhang, and Y. Chen, "Forecasting of Photovoltaic Power Production using LSTM Approach," in Advanced Statistical Modelling, Forecasting, and Fault Detection in Renewable Systems, 2021, ch. 1