Real-time Visualization of Sensor Data in Smart Manufacturing using Lambda Architecture

Nadeem Iftikhar[1], Bartosz Piotr Lachowicz[1], Akos Madarasz[1], Finn Ebertsen Nordbjerg[1], Thorkil Baattrup-Andersen[2] and Karsten Jeppesen[1]
[1]*University College of Northern Denmark, Aalborg 9200, Denmark*
[2]*Dolle A/S, Frøstrup 7741, Denmark*

Keywords: Industry 4.0, Real-time Visualization, Sensor Data, Smart Manufacturing, Lambda Architecture, Machine Learning.

Abstract: Smart manufacturing technologies (Industry 4.0) as solutions to enhance productivity and improve efficiency are a priority to manufacturing industries worldwide. Such solutions have the ability to extract, integrate, analyze and visualize sensor and data from other legacy systems in order to enhance the operational performance. This paper proposes a solution to the challenge of real-time analysis and visualization of sensor and ERP data. Dynamic visualization is achieved using a machine learning approach. The combination of real-time visualization and machine learning allows for early detection and prevention of undesirable situations or outcomes. The prototype system has so far been tested by a smart manufacturing company with promising results.

1 INTRODUCTION

Manufacturing industry nowadays is looking to increase productivity and enhance efficiency through integration of new digital industrial technology (Industry 4.0) within the production system. The production system includes but is not limited to: enterprise resource planning (ERP), manufacturing execution system (MES), control and hardware. Hence, to improve production performance it is critical to monitor the entire production system and predict any failures in order to avoid unplanned machine stoppages. This paper presents real-time production monitoring techniques capable of providing a live perspective on production performance using multiple data sources including sensor data. Additionally and importantly, the real-time sensor data is matched against predicted sensor data to alert in case of anomalies. Real-time production monitoring allows production managers and machine operators to comprehend and master incidents by taking timely actions.

To summarize, the main contributions in this paper are as follow: proposing a lambda architecture to analyze and visualize real-time and batch data, providing methodologies for real-time production monitoring, presenting a machine learning model to raise alerts at real-time, and demonstration of the proposed solution at a smart manufacturing company.

The paper is structured as follows. Section 2 describes the proposed approach to visualize sensor data. Section 3 provides the details about real-time production visualization methods. Section 4 introduces the machine learning model. Section 5 illustrates the data visualization dashboards. Section 6 presents the related work. Section 7 concludes the paper and points out the future research directions.

2 APPROACH

This section provides insight into the approach taken to build a complete solution of relevant tools and techniques to analyze and visualize real-time and historical data. In order to build this system, lambda architecture approach is chosen (Marz and Warren, 2015). The main idea behind lambda architecture is to build a system using a layered approach. Each layer has its own set of properties, techniques and technologies, and builds upon the functionality provided by the layers underneath it. The three layers of the lambda architecture are: speed, batch and serving. The speed layer processes recent data. The batch layer processes all the data present in the master data storage and the serving layer stores the real-time as well as batch views for visualization and later analysis.

Iftikhar, N., Lachowicz, B., Madarasz, A., Nordbjerg, F., Baattrup-Andersen, T. and Jeppesen, K.
Real-time Visualization of Sensor Data in Smart Manufacturing using Lambda Architecture.
DOI: 10.5220/0009826302150222
In *Proceedings of the 9th International Conference on Data Science, Technology and Applications (DATA 2020)*, pages 215-222
ISBN: 978-989-758-440-4

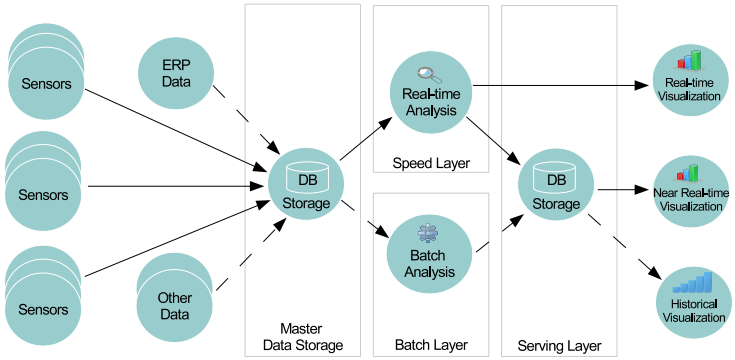

Figure 1: Conceptual lambda architecture.

Some of the benefits of the proposed solution are listed below: Industry 4.0 (fourth industrial revolution) connectivity by connecting production machines, sensors and ERP data in order to visualize the entire production system; enhancing the business process by minimizing downtime; boosting real-time visibility by using dashboards for prompt alerts; and providing end-users with unified views.

Most of the components of the proposed solution are successfully deployed and tested in a manufacturing company (*Dolle*). Dolle is recognised as the market leader in Europe for manufacturing timber loft ladders.

2.1 Lambda Architecture

The conceptual lambda architecture is presented in Fig. 1. The proposed architecture consists of six main modules: a sensor network; ERP data integration; integration of other external data, for example, indoor/outdoor weather data; master data storage layer for persistence; speed layer to process data streams in real time; batch layer to perform extended time analysis; and serving layer to store the output from the speed and batch layers. The solid arrows both in Fig. 1 and Fig. 2, represent streaming data, while dashed arrows represent batch data. Further, Fig. 2 presents the concrete lambda architecture. The sensor network consists of sensor nodes deployed throughout the machines to measure the status of the machines such as, machine on/off, pace in, pace out and fault/error. The ERP data contain details about products, job executions and work calendar. In addition, other data may include indoor temperature, humidity and so on. The data transmitted by the sensors is streamed into the master data storage (MongoDB) by *Phoenix controller*[1]. The Phoenix controller captures data (ranging from 1 to 10 times per second) from the sensor network and submits the data to MongoDB. The controller submits the data of a sensor in MongoDB, only if there is a change in the state of the sensor. If there is no change in the state of the sensor the value will not be stored. Further, the external data such as ERP data, is also submitted to the master data storage.

```
Sensor data snapshot:

[{'timestamp': '2020-02-03 08:45:30',
  'value': 'true',
  'port': '0103'},
 {..}, ..}]

ERP data snapshot:

[{'planned_start':'2020-02-03 08:00:00',
  'planned_stop':'2020-02-03 11:00.00',
  'actual_start':'2020-02-03 08:15:00',
  'actual_stop':'2020-02-03 11:45:00',
  'item_id':'550583',
  'machine_id':'1404',
  'name':'SW365/3section/12threads/..', ..}]
```

The sensor data snapshot provides a quick overview of the data. The sensor data (JSON format) contains three attributes. The data has varying granularity as explained above. The first three rows in the sensor data snapshot read as follows. Timestamp, represents the date and time of the sensor data acquisition. Value, corresponds to the state of the binary sensor (either TRUE or FALSE) and port, indicates the sensor number. For example, port = 103 represents pace out sensor (rolling out of a ladder). Similarly, the ERP data snapshot reads as follows. Planned start and stop, represent the planned job execution date and time. Actual start and stop, denote the actual job execution data and time. Item-id, identifies the product that is being produced. Machine-id, represents the machine and finally name, represents the type of product being produced. A custom-built web application programming interface (Dolle API) allows for access to the sensor and external data from the mas-

[1]https://hexdocs.pm/phoenix/Phoenix.Controller.html

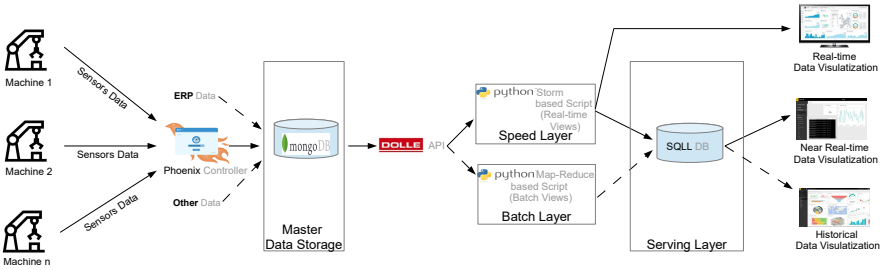

Figure 2: Concrete lambda architecture and the software platforms.

ter data storage by means of an HTTP GET request to provide the data to the speed layer for processing. The speed layer running on the Apache Storm platform provides real-time views, while the batch layer running on the Hadoop MapReduce platform delivers precomputed batch results. The real-time views are directly streamed to the dashboard(s) for real-time visualization using Dash, which is a Python framework for building web applications. The real-time views are also delivered to the serving layer with some latency for near real-time visualization.

3 METHODOLOGY

3.1 Overview

This section describes the proposed real-time visualization methods. Method 1 measures the productivity in terms of the number of ladders produced during a given period of time, whereas, method 2 calculates the length of time elapsed since last ladder produced. Both methods use a sliding window approach to get the data from the master data storage at predefined intervals (5 seconds in this case) and pass the data to the speed layer. In Fig. 3, the first window (blue) contains sensor data that arrives between the zeroth and nine-hundredth second. The second window (green) contains sensor data that arrives between the fifth and nine-hundred and fifth second. Note that data from fifth through nine-hundredth second is present in both windows. When the second window (green) is executed at time $t = 5$, data between the zeroth and fifth second is dropped from the data queue.

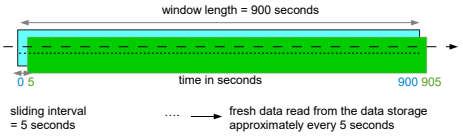

Figure 3: Sliding window.

3.2 Production Monitoring

The purpose of the production monitoring method is to measure the productivity with respect to number of ladders produced and display it as a live update every time the method is executed (approximately every 5 seconds). The main idea behind the method is to monitor the production process at real-time and to detect errors/faults at earliest possible time in order to take counter measures. Later in Section 5, this monitoring method is combined with prediction models, enhancing its ability to raise alerts at real-time.

The following outlines the four main steps of the productivity measurement method:

1. Read the sensor data from the master data storage every 5 seconds with a sliding window size of 900 seconds;
   - The sliding window's *start time* = *(current time - interval)* and *end time* = *current time*. Where, the *interval* is 900 seconds.
2. The data is then fed to the speed layer;
3. The speed layer performs the following functionality:
   - The data is filtered by selecting the values, where port equals to TRUE and ignoring the values, where port equals to FALSE;
   - The values of the port are aggregated in key-value pairs;
   - The aggregated results are displayed on dash board;
   - The aggregated data for real-time views are stored in the serving layer every 300 seconds for near real-time visualization;
   - The key-pairs are reset to zero.
4. GOTO step 1.

The following example will help in understanding how the sensor data is being processed by the speed layer. For this example, port/sensor 103 (pace out) is considered. The sliding window technique is used to visualize sensor data at real-time. The proposed

method divides the data in slices of constant length of 900 seconds. Further, the algorithm writes to the SQL database every 300 seconds for near real-time visualization. After executing the real-time processing method at 2020-02-24 14:15:00 (current date/time), it is of interest to see the snapshot of the data in the sensor data table (Table 1). As, the predefined window size is 900 seconds for that reason the start of window date/time is (2020-02-24 14:15:00 - 900 seconds) = 2020-02-24 14:00:00 and end of window date/time is 2020-02-24 14:15:00. In this table, the granularity of each rows differs. This is due to the fact that the sensor value for a specific port is only registered when it changes (different from the previous value). Each row in this table represents the state of the pace out sensor either TRUE or FALSE. TRUE means that the pace out sensor detects the rolling out ladder section and FALSE means vice-versa.

Table 1: Sensor data.

Timestamp	Port	Value
2020-02-24 14:00:00	103	TRUE
2020-02-24 14:00:01	103	FALSE
2020-02-24 14:00:17	103	TRUE
2020-02-24 14:00:19	103	FALSE
.	.	.
2020-02-24 14:01:19	103	FALSE
2020-02-24 14:01:42	103	TRUE
2020-02-24 14:01:43	103	FALSE
2020-02-24 14:02:01	103	TRUE
.	.	.
2020-02-24 14:14:29	103	TRUE
2020-02-24 14:14:31	103	FALSE
2020-02-24 14:14:46	103	TRUE
2020-02-24 14:14:47	103	FALSE

Table 2: Key-Value pair.

Port	Value
103	42

Further, the count of TRUE's is saved in a key-value pair of which port is the key and the count of TRUE's is the value (Table 2). This table shows that in the last 900 seconds 42 ladder sections were produced. Moreover, the data also posts live to the dashboard. The tachometers in Fig. 4 shows two readings at 15 minutes and hourly levels.

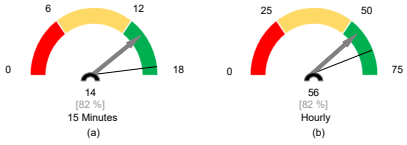

Figure 4: The pace of rolling out ladder.

The tachometer in Fig. 4(a) shows that 14 ladders of type *SW365/3section/12threads* were produced in the last 15 minutes and that the average number of ladders of this type are produced during this time is 17, so the performance is approximately 82%. The formula to calculate the pace out per 15 minutes is *number_of_ladder_sections / number_of_sections = 42 / 3 = 14*. The number of ladder sections are divided by three since the finished product (ladder) is a three section ladder (Fig. 5).

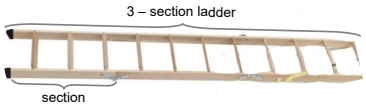

Figure 5: Three section ladder.

Similarly, the tachometer in Fig. 4(b) shows that 56 ladders are expected to be produced in an hour and the hourly capacity for the specific type of the ladder is 68, so the performance is approximately 82%. The formula to calculate the hourly production rate (the number of items produced per hour) is *(number_of_ladder_sections * 4) / number_of_sections = (42 * 4) / 3 = 168 / 3 = 56* ladders per hour. The number of ladder sections are multiplied by four is for the reason that there are four quarters in an hour and it is divided by three since the finished product (ladder) consists of three sections.

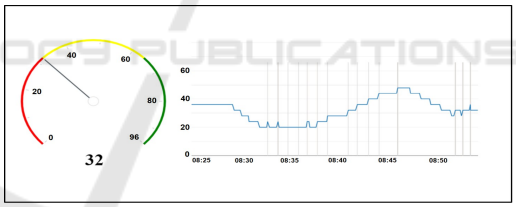

Figure 6: Hourly production rate (below average).

Further, a below average production situation is shown in Fig. 6. It shows an hourly production rate of 32 ladders against an expected average hourly production rate of 68 ladders. The grey lines represent the ladder output. In this case the rolling out ladder pace is 47%. A fault situation is shown in Fig. 7. The tachometer shows an hourly production rate of zero ladders. The reason is that current time is 09:12 and

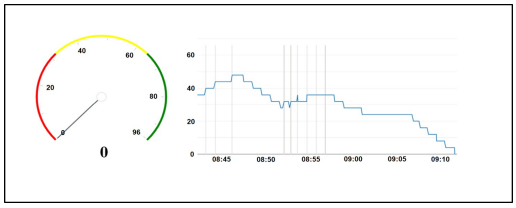

Figure 7: Hourly production rate (fault).

the last ladder was produced at 08:56. As a result no ladders were produced within the last 15 minutes dropping the hourly production rate of ladders to zero.

3.3 Time Elapsed

The purpose of the time elapsed method is to keep track of when the last time the sensor triggered and display it as a live update every 5 seconds. The idea is to be able to follow the sensor triggering pattern and see how many seconds have elapsed since the sensor was last triggered. This value is then compared to the expected range of the patterns. The method works as follows. First, the Dolle API gets the sensor data from the master data source in a sliding window size of 900 seconds. Following this, the acquired data is passed to the speed layer. The initial data format is JSON, which is then transformed into interpretable variables, such as integers and real numbers. Finally, calculations on the data are performed, only considering timestamps where the port value is TRUE and directly followed by a value of FALSE. Each time the method is executed it calculates the difference between the last timestamp of TRUE to FALSE transition and the current timestamp *(time_elapsed = current_time - last_output_time)*. For example, if the value of port 103 was TRUE at 2020-02-24 14:14:01 and at 2020-02-24 14:14:04 it became FALSE, then, if the method is executed at 2020-02-24 14:14:53 in that case it will show time elapsed since last ladder was produced = 2020-02-24 14:14:53 - 2020-02-24 14:14:01 = 52 seconds. In addition, the method resets the time elapsed to zero if the machine turns off. This ensures that the time elapsed value is displayed only if the machine is running and expected active in the production.

The following outlines the five main steps of the time elapsed method:

1. Read the sensor data in the sliding window size of 900 seconds from the master data storage approximately every 5 seconds;
2. Filter the data where the values are TRUE and transitioning to FALSE;
3. Get the time stamp for the last transitioning event for the specific port;
4. Calculate the time elapsed that is by subtracting the *last_timestamp* from the *current_timestamp*;
5. GOTO step 1.

Furthermore, the tachometers in Fig. 8 demonstrate the time elapsed. In Fig. 8(b), it can be seen that the time elapsed for port 103 (rolling out ladder) is 52 seconds, whereas, the average rolling out time of this

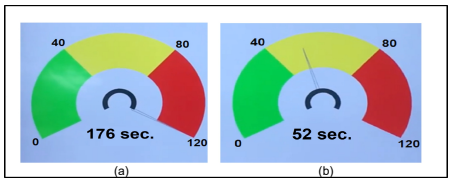

Figure 8: Time elapsed.

ladder type is 40 seconds. An alert situation (in red) is shown in Fig. 8(a). The tachometer shows an elapsed time of 176 seconds. That means that there were no ladders produced in the last 176 seconds. This indicates a need for immediate action as could be caused by a fault in the machine.

4 MACHINE LEARNING

This section introduces the real-time pattern detection machine learning model. To detect abnormal patterns is an important feature of smart manufacturing. For example, if the performance of the machine is diverging from the standard pattern then it could cause an unwanted stop to the whole production system. Hence, to achieve production efficiency it is vital to detect errors/faults or abnormal patterns in real-time in order to take early and appropriate actions to keep the production running without unwanted stops. The incoming real-time sensor data is compared to standard patterns by using machine leaning models and an alert could be triggered when some abnormal event is predicted to happen such as, the machine is going to stop. Further, Fig. 9 describes the pattern detection machine learning algorithm in detail. As mentioned in Fig. 2, the speed layer and batch layer based on Python Storm and MapReduce, respectively. The serving layer consists of a SQL Server. The proposed algorithm uses batch layer (MapReduce) to train the model to detect abnormal patterns. The MapReduce job runs at predefined intervals and updates the coefficients of the pattern detection models in the serving layer (SQL Server). Further, the speed layer reads the incoming streaming data from the MongoDB and detects abnormal patterns by applying the pattern detection algorithm. The pattern detection algorithm dynamically uses the latest calculated model coefficients obtained from the serving layer. In Fig. 9, the solid arrows represent streaming data, while dashed arrows represent batch data.

Several forecasting models are available, in this paper Auto-regressive model (AR) is chosen. An AR model is a widely used linear model that works on stationary time series. AR models use the dependent relationship between a value y and some number of

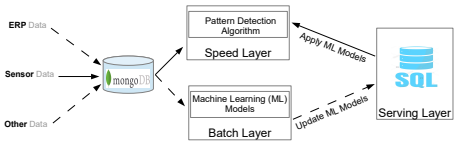

Figure 9: Pattern detection machine learning method.

lagged values y (Black, 2011). The regression coefficients in the standard AR models do not vary, while in the proposed model the regression coefficients may vary with the ladder type. For this reason, regression coefficients are calculated for each ladder type, separately. In other words, it means that for each ladder type a separate model will be trained. The value of t indicates the ladder type. As, there are approximately 30 different main ladder types, the time index parameter will be i_t, where $t = 1,..,n$ and n is the number of ladder types ($n = 30$). The proposed AR model of order P can be written as follows, Equation 1.

$$\hat{y}_{i_t} = c_t + \sum_{\lambda=1}^{P} \alpha_{t,\lambda} y_{i_t-\lambda} + \varepsilon_{i_t} \quad (1)$$

where c_t is the intercept with the $y-axis$ (a constant), $\alpha_{t,\lambda}$ is the regression coefficient, ε_{i_t} is the value of the white noise.

The following outlines the six main steps of the pattern detection algorithm:

1. Compute the AR model for the specific ladder type at the batch layer;
2. Update the coefficients of the model to the serving layer every 15 minutes;
3. Read the sensor data with a window size of 900 seconds from the master data storage every 5 seconds and predicts the reading for each sensor using the AR model;
4. Calculate the distance between the predicted and the actual sensor reading;
5. If the distance is above a certain threshold then raise an alert;
6. GOTO step 1.

5 DATA VISUALIZATION DASHBOARDS

The section describes the use case to deploy the proposed prototype system at Dolle (a smart manufacturing company in Frøstrup, Denmark). Dolle has deployed 14 sensors on two machines to collect production related data. The prototype system is demonstrated from the perspective of production managers and machine operators. The system uses real-time data collected from the deployed sensors from both the machines as well as ERP data. The sensor data consists of machine status, alarm, pace of side rails and wood rungs, pace of output products (ladders), faulty side rail, screwing machine error and so on. The ERP data consists of machine-id, job-id, ladder type, planned job start and stop date/time, actual job start and stop date/time, number of ladders produced, down time and so on. The sensor data is acquired with an interval of every five second, while the ERP data is acquired at the start of each new job. Wall mounted dashboards reporting the hourly production rate and elapsed time since the last ladder was produced are presented in Fig. 10.

Figure 10: Real-Time production monitoring dashboards.

Further, Fig. 11 shows the hourly production rate (the pace of number of items produced per hour) in real-time. It is important to monitor the hourly production rate in order to keep the production running. Fig. 11, displays the hourly production rate or pace that has dropped to zero (tachometer on the left) also visible on the graph (right) that no ladders were produced in the last 15 minutes. The last ladder was produced at 08:57. The grey lines in the graph represents the exit of an ladder from the machine. The dashboard also displays the ERP data such as, job-id, ladder type, machine number, planned start and stop, actual start and predicted stop. In addition, the dashboard displays the planned and actual number of ladder produced, time since last error as well as error type and machine downtime in the last hour. It is interesting to observe that the date/time for the predicted stop is automatically adjusted depending on the remaining number of ladders to produce and the machine downtime for the current job.

Moreover, machine learning models depending on the ladder type are applied to predict hourly production rates. The difference between the predicted and actual data is measured against thresholds and the presence of significant anomalies automatically trigger alerts, such as changing the color coding of the line in the graph (Fig. 11). The changing of color from blue to yellow and later to red triggers warning so that the situation can be actioned.

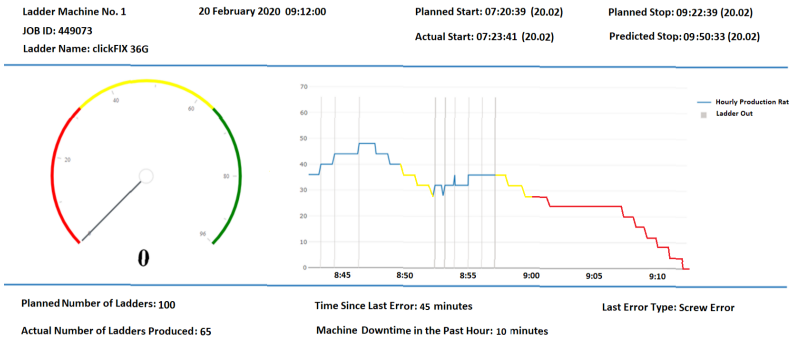

Figure 11: Real-Time visualization of hourly production rate with machine learning.

6 RELATED WORK

This section mainly concentrates on previous work done in relation to data analysis, data visualization and machine learning in smart manufacturing.

A near real-time and easy to use operational process dashboard designed for the machine operators to monitor the information about the products and the performance of the manufacturing process is presented by (Gröger et al., 2013). A state-of-the-art review by (Bordeleau et al., 2018) indicated that real-time monitoring and analysis are the major areas of research in the context of Industry 4.0. Similarly a comprehensive study by (Zheng et al., 2018) examined smart manufacturing systems for Industry 4.0 and presented some scenarios for production monitoring. In addition, (Tokola et al., 2016) suggested multiple dashboards for end-users in manufacturing industry based on different hierarchy levels to visualize operational, tactical and strategical data. These previous works focus on various conceptual aspects and recent advancements of data visualization and analysis in medium and large-sized manufacturing companies. The work presented in this paper builds upon the ideas presented in those previous works, however, the focus of this paper is to provide practical application of data analysis and visualization technologies in relation to data from operational and production systems in small and medium-sized (SME) enterprises.

Furthermore, a real-time monitoring system that utilizes IoT-based sensors, big data processing and various prediction models is proposed (Syafrudin et al., 2018). The main focus of this work is to visualizer indoor climate data at real-time and classify the values for fault prediction. In contrast to this work, the work presented in this paper emphases on real-time operational and production data analysis and visualization.

In connection with machine learning in smart manufacturing. A state-of-the-art review of deep learning techniques for machinery fault diagnosis, predictive analysis and defect prognosis is presented by (Wang et al., 2018). Predicting the bottlenecks in a production system by using ARIMA model is proposed by (Subramaniyan et al., 2018). Further, (Shin et al., 2017) presented a model for predicting energy consumption of manufacturing machinery. A generic solution in environment monitoring that is capable of detecting anomalies and displaying the recent situation is proposed by (Trilles et al., 2015). Finally, (Iftikhar et al., 2019) described the basic building blocks to construct a complete solution for data analysis and visualization.

To the best of our knowledge, this paper is one of very few to deal with the practical aspects of production monitoring and machine learning at real-time in order to enhance operational efficiency in manufacturing industry.

7 CONCLUSIONS AND FUTURE WORKS

This paper has presented the possibilities of real-time production monitoring based on a real world case study in the manufacturing industry. The proposed solution is quite flexible and salable to develop concrete prototype system for real-time, near real-time and historical data analysis and visualization. To enhance the operational efficiency real-time visualization dashboards were developed. Moreover, regression based machine learning technique was also used to detect anomalies. The effectiveness of the production monitoring dashboards and the performance of the prediction method were tested at the premises of a manufacturing company with promising results.

For the future work, a real-time interactive dashboard based on the idea of predictive maintenance will be developed. Further, it will be investigated how

the real-time dashboards help the smart manufacturing companies in general, in order to enhance their operational efficiency and productivity. Finally, the accuracy of the machine learning model will be tested to see how well it can predict the likelihood of developing anomalies in a machine.

REFERENCES

Black, K. (2011). *Business Statistics: For Contemporary Decision Making*. John Wiley & Sons, 9th edition.

Bordeleau, F. E., Mosconi, E., and Santa-Eulalia, L. A. (2018). Business intelligence in industry 4.0: State of the art and research opportunities. In *51st Hawaii International Conference on System Sciences*, pages 3944–3953. AIS e-Library.

Gröger, C., Hillmann, M., Hahn, F., Mitschang, B., and Westkämper, E. (2013). The operational process dashboard for manufacturing. *Procedia CIRP*, 7:205–210.

Iftikhar, N., Baattrup-Andersen, T., Nordbjerg, F. E., Bobolea, E., and Radu, P. B. (2019). Data analytics for smart manufacturing: A case study. In *8th International Conference on Data Science, Technology and Applications*, pages 392–399. SCITEPRESS Digital Library.

Marz, N. and Warren, J. (2015). *Big Data: Principles and Best Practices of Scalable Real-time Data Systems*. Manning Publications Co., New York.

Shin, S. J., Woo, J., and Rachuri, S. (2017). Predictive analytics model for power consumption in manufacturing. *Procedia CIRP*, 15:153–158.

Subramaniyan, M., Skoogh, A., Salomonsson, H., Bangalore, P., and Bokrantz, J. (2018). A data-driven algorithm to predict throughput bottlenecks in a production system based on active periods of the machines. *Computers & Industrial Engineering*, 125:533–544.

Syafrudin, M., Alfian, G., Fitriyani, N. L., and Rhee, J. (2018). Performance analysis of iot-based sensor, big data processing, and machine learning model for real-time monitoring system in automotive manufacturing. *Sensors*, 18(9):2946.

Tokola, H., Gröger, C., Järvenpää, E., and Niemi, E. (2016). Designing manufacturing dashboards on the basis of a key performance indicator survey. *Procedia CIRP*, 57:619–624.

Trilles, S., Schade, S., Belmonte, O., and Huerta, J. (2015). Real-time anomaly detection from environmental data streams. In *AGILE'15 Geographic Information Science as an Enabler of Smarter Cities and Communities*, pages 125–144. Springer, Cham.

Wang, J., Ma, Y., Zhang, L., Gao, R. X., and Wu, D. (2018). Deep learning for smart manufacturing. *Journal of Manufacturing Systems*, 48:144–156.

Zheng, P., Sang, Z., Zhong, R. Y., Liu, Y., Liu, C., Mubarok, K., Yu, S., and Xu, X. (2018). Smart manufacturing systems for industry 4.0: Conceptual framework, scenarios, and future perspectives. *Frontiers of Mechanical Engineering*, 13(2):137–150.