ISSN 1330-3651 (Print), ISSN 1848-6339 (Online) https://doi.org/10.17559/TV-20200906191853
Original scientific paper

Deep Learning-Guided Production Quality Estimation for Virtual Environment-Based Applications

Akm ASHIQUZZAMAN, Hyunmin LEE*, Tai-Won UM, Kwangki KIM, Hye-Young KIM, Jinsul KIM*

Abstract: In modern smart factories, quality estimation is vital for maximum productivity. However, quality estimation by definition relies on an imbalanced dataset, as most smart factories are highly efficient. In this research, we propose a guided quality estimation system that can recognize faulty data among a highly imbalanced production dataset. We also propose a customized LSTM model that is trained to ensure high accuracy in the quality estimation system. This is achieved by our proposed batch-wise balanced training method. Moreover, traditional means of evaluation for this type of method are not suitable, again due to the highly imbalanced nature of the dataset. Thus, a proper evaluation metric is also discussed. The proposed customized LSTM model with custom batch-wise SMOTE + ENN achieved 99.9% accuracy with an f_1 score of 95%. This new proposed method for the imbalanced smart factory quality estimation will improve drastically and give pathway to more improved quality. Finally, we discuss practical implementation for the edge server consisting of the proposed guided production estimation system and real-time visualization. Feasibility analysis of this virtual environment-based application of the proposed framework ensured low computational overhead and faster processing.

Keywords: data rebalancing; deep learning; ensemble Learning; industrial control; information management

1 INTRODUCTION

Smart factories represent a new, technology-driven approach that utilizes internet-connected machinery to monitor and estimate production-based forecasting [1]. Rapid network development and the growth of modern edge-based solutions for faster computing have given rise to a new generation of smart factories. Most of the newest factories or production houses, also known as smart factories, have gradually shifted into fully cloud-based solutions to achieve full automation in production. This type of automation is done by incorporating IoT (Internet of Things)-based devices into the internal production itself. IoT devices grant the ability to expand smart factory production related computation to transfer over the edge and in open remote environments. Mohammadi et al. [2] estimated that the total market share of smart factory related applications will surpass $2.7 trillion by 2021. This rise of smart factories will eventually require a great deal of modern prediction and monitoring-related support to ensure optimal production values.

A smart factory can analyse patterns in the data and verify steps in those processes. This is commonly referred to as a smart factory-based Quality Estimation System (QES). Human quality checking is prone to error and biases, and can sometimes fail due to simple negligence. In a high-precision production line, such as that used to produce semiconductors or automobile/airplane parts, these IoT-based smart factory QESs can go far in maximizing production. Also, this machine-driven system usually works thousands to millions of times faster than their human counterparts, resulting in a faster production timeline. Quality estimation or production quality estimation is not new. Essentially, QES involves measurement of various production-related and environmental conditions to produce statistical predictions. Historically, this was accomplished with a statistical quality control process [3], in which process control data is used to determine a statistical range based on the probability distribution of the data and the samples collected. However, the idea of using deep learning to estimate this process has not been extensively researched or explored.

There is, however, much research on data load distribution and QoS (Quality of Service) prediction [4]. The usage of cloud-based deep learning prediction for intelligent prediction and monitoring in modern smart factories has not been properly researched [5]. The main drawback to neural network-based monitoring and anomaly prediction is the huge computational overhead, which is not always possible to provide in a remote factory setting. This shortcoming of traditional approaches encourages the development of newer algorithms and frameworks, in particular, cloud-based platforms that enable deep learning [6]. Such systems process data in real time and give predictions based on the results [7]. A standard cloud-based virtual platform with the support of edge-based server-driven computational load balancing can be used to create a neural network for real-time prediction of production data, which can be monumental for smart factory-based applications. Smart factory-based production has seen very little improvement in the current research landscape. The main drawback of this approach is the lack of a framework that combines a virtual platform with edge-based servers and deploys this system in the field as a fast, high-quality prediction system.

To address the limitations and problems that currently exist in smart factory-based production quality estimation and monitoring systems, we proposed a new ensemble model that combines multiple deep learning models and training strategies to create a new kind of production quality estimation system. Our proposed model both collects sensor data from the factory in real-time and processes this data. Later, we propose various deep learning models that enable quality estimation for the final factory products. We also introduce an edge server-based application and micro-services on a virtual platform in the mobile edge for faster response.

2 RELATED WORKS

Use of the QES program for factories is not a new concept. Numerous systems have been used for factory production quality control. Some legacy programs rely heavily on traditional statistical methods. In the Fourth Industrial Revolution, the widespread use of computing

power enables data processing on a much larger scale than before. Also, real-time visualization and neural network-based prediction is a very recent convergent technology. This topic has not seen much in-depth research [8]. However, there has been some recent research on neural network-based production monitoring and intelligent prediction [1]. In the 1950s, Nathanial Rochester of IBM Research Labs led the first attempt to model a neural network [9]. But the research algorithms that lacked multi-layered perceptrons, which were the precursors of modern neural networks, could not achieve the necessary training and accuracy. A back-propagation algorithm developed in 1989 was used to train neural networks and gave rise to the new generation of neural networks [10]. On the other side, cloud computing and virtual platform-based edge computing technology has been quite thoroughly researched [11]. IoT-based applications equipped with deep learning capabilities have been researched as means of improving service quality and resource optimization [12].

Guided quality estimation systems (QESs) have not yet been thoroughly studied for development of highly effective frameworks. Relevant data processing techniques and methods of tackling imbalanced datasets to ensure proper training have not been addressed in any current research [13]. By utilizing research on malicious detection with a custom data-rebalancing scheme, a new type of guided production system can be derived that addresses these problems in smart factory-based production quality estimation and monitoring systems. We propose a new ensemble model that combines multiple deep learning models and training strategies into a new kind of production quality estimation system. Our proposed model is a combination system that collects sensor data from the factory in real-time and processes this data to obtain pre-processed data. Then we propose various deep learning models to obtain the final production quality estimation. We also introduce an edge server-based application and micro-services that rely on a virtual platform for faster response times.

3 PROPOSED METHOD

The proposed system can be divided into three main parts. The first two steps are data collection and processing. As data collected in a factory mostly consists of categorical quality reports, most of the time the data needs to be processed into a distinctive categorical dataset. So, the final part of the model involves detection of the given data inputs in real time. Fig. 1 shows the whole system in a basic block diagram. The deep learning system reads the data and predicts the output; the data is then reviewed to detect potential quality control problems in real time. The data is then pre-processed for both the balancing technique and the final online detection scheme. The second part of the system entails customized data balancing via machine learning. The model is then tweaked according to the performance evaluation. Finally, real-time data is fed into the online-based application and the program detects faulty data in real time. The basics premise of the common QES algorithm is rather simple. The previous estimation methods can be expressed with some simplified equations.

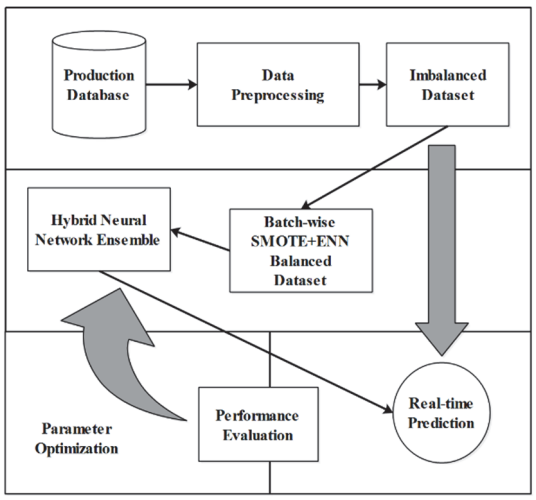

Figure 1 Proposed System Block Diagram

If there are $x_1, x_2, x_3, \ldots, x_n$ observed production-related elements and the final quality estimation is Q_{es}, we can express the relationship between these entities as:

$$Q_{es} = w_1 \cdot x_1 + w_2 \cdot x_2 +, ..., + w_n \cdot x_n \quad (1)$$

here, $w_1, w_2, w_3, \ldots, w_n$ are the weight factors or coefficients for each judge/quality checker value. Now, if we assume that Q_{es} is a non-linear estimate which can be calculated based on a threshold value of any non-linear function, we can rewrite the whole equation as follows:

$$Q_{es} = f\left(w_1 \cdot x_1 + w_2 \cdot x_2 +, ..., + w_n \cdot x_n\right) \quad (2)$$

where $y = f(x)$ is the activation function. This linear function can be used to predict the proper estimation base on the error minimization of the data fitting. Previous works on QES relied heavily on the statistical estimation process for each input and gave the weights based on calculations that had previously been done by sampling. However, in our proposed system, this output is given by the combined hybrid DNN and LSTM together with the weighted average of their perspective training time accuracy and, finally, the softmax function for the final result. DNN and LSTM each have particular advantages over different kinds of algorithms; many researchers have reported improved results with this kind of hybrid ensemble [14]. The combination of RNN and LSTM has shown superior performance compared to benchmark neural network ensembles [15]. Given our research and previous literature reviews, a hybrid model that combines both LSTM with DNN is proposed for the hybrid neural network. A recurrent neural network (RNN) is a sequence based learning algorithm. Repetitive structures or recurrent input or data flows classify a network as a recurrent structure. RNNs are designed to process information in sequence. In a typical neural network, all inputs and outputs are considered to be independent of each other, but that assumption does not always hold true. RNN gives special advantages in processing data with chronological patterns. An RNN has a "memory" which stores details about what has been measured up to the present point in time. In theory, RNNs can underline series data with indefinitely long sequences. But in practice, they are

limited to looking back over only a few steps [16]. Long short-term memory (LSTM) is a special kind of RNN [17] that was first proposed by Hochreiter et al. [18]. LSTM is a specialized type of RNN with a memory cell, making it capable of learning from very long sequences [19]. The equations below describe how a layer of memory cells is updated at every time step *t,* where x_t is the input to the memory cell layer at time *t*; W_i, W_f, W_c, W_o, U_i, U_f, U_c, U_o and V_o are weight matrices; b_i, b_f, b_c and b_o are bias vectors; I_t is the input gate; and $\tilde{C}_t$ is a candidate value for the state of the memory cells.

$$I_t = \sigma\left(W_c \cdot x_t + U_c \cdot h_{t-1} + b_i\right) \tag{3}$$

$$\tilde{C}_t = tanh\left(W_c \cdot x_t + U_c \cdot h_{t-1} + b_c\right) \tag{4}$$

Now, forget gate activation f_t at time *t* will be:

$$f_t = \sigma\left(W_f \cdot x_t + U_i \cdot h_{t-1} + b_f\right) \tag{5}$$

So, the C_t memory cells in the new state at time *t* will be:

$$C_t = I_t \cdot \tilde{C}_t + f_i \cdot C_{t-1} \tag{6}$$

Followed by the new outputs o_t and h_t:

$$O_i = \sigma\left(W_o \cdot x_t + U_o \cdot h_{t-1} + V_o \cdot C_t + b_o\right) \tag{7}$$

$$h_t = O_t \cdot tanh\left(\tilde{C}_t\right) \tag{8}$$

By design, LSTM does not have the vanishing or exploding gradient problems [20]. LSTM is also difficult as it requires more computation to complete one training loop than a general DNN. However, use of proper weight initialization with a suitable activation function is effective at creating a faster LSTM [21].

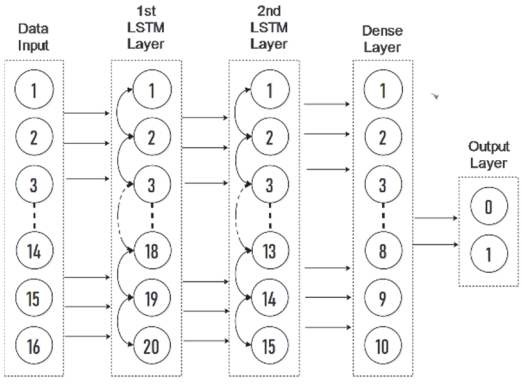

Figure 2 Proposed Hybrid Network

The proposed neural network combines DNN and LSTM into a single architecture that makes use of the advantages of both models. The proposed neural network has not been designed to be too deep or to have too many hidden layers, as the dataset it uses has comparatively few dimensions. The production quality cycle data used in this study has 16 features, as described in detail in later sections. So, the proposed model has those 16 features as input, and the output LSTM node of the first consecutive layer has 20 nodes with 1 step time sequencing return. This LSTM then works as the input of the next layer, which consists of 15LSTM node layers that goes to the first fully connected dense general network layer with10 nodes. A prior study suggested that neural network-based binary classification gives better results if the output classification is trained as a two-class classification. This is due to easy separation of the data dimensions [22]. The proposed neural network follows the same general principle, in that it gives 2-node output with softmax activation. The fully connected layers used in this neural network had Relu activation [23]. The main addition in this ensemble is the introduction of batch normalization [24] to reduce over-fitting and convergence of data during training. A simplified block diagram of the network is shown in Fig. 2. Adjustment of the hyper-parameters for the learning model also includes selection according to the experience of a few primary hyper-parameters, the designation of candidate values, and then several parallel experiments to determine the values that lead to a good result. The results are always very significant. Hyper-parameter optimization is critical for deep learning. So, for this ensemble, we selected the hyper-parameters based on a small sub-data set with limited iteration to identify the most effective hybrid DNN parameters. Tab. 1 describes the process of algorithm hyper-parameter selection based on grid searching.

The output of the proposed method is slightly different from the general regression of the activation-based model. Because of the highly imbalanced nature of the dataset, a new approach to output prediction was taken. The LSTM module transfers the data into a classical fully connected layer or a dense branch. These layers detect or classify the class or label, and the final output layer has two nodes that give a soft max-based output of the given input. The detailed layer-wise implemented model is shown in Tab. 2.

Table 1 Algorithm Hyper-parameter Selection Based on Batch Data by Grid Searching

Hyper-Parameters	Considered Values	Adopted Values
LSTM Nodes	20, 15, 10	20
LSTM Activation	tanh, sigmoid, Relu	Sigmoid
Batch Normalization Momentum	0.75, 0.80, 0.99	0.99
Batch Normalization Epsilon	0.01, 0.001, 0.0001	0.001
Fully Connected Layer Node Number	5,10,15	10
Fully Connected Layer Activation	tanh, sigmoid, Relu	Relu

The main dataset on production quality control in the automotive industry, which is a large database, consists of 20 separate columns with multiple data types and 10382 rows of data. The main column and the data types are shown below. The whole model for the raw data contains 20 distinct columns, and some of the data has missing values. These data are then transferred into the new clearing and processing model. Not all data points in this database have the same degree of importance for model learning and processing. So, some hand picking was done to reduce the amount of insignificant data. After the

selection of relevant data, 16 columns were selected that have the most significant values and contain data that is important to show and display. After determining the 16 columns for the main display and learning process, the values in the16 columns were called, extracted and saved.

Table 2 Proposed Neural Network Configuration

Layer Style	Nodes in Layer	Comment
Data-Input	16	Input Features
LSTM	20	Input Layer
LSTM	15	Hidden Layer
Dense-Layer	10	Hidden Layer
Dense-Layer	2	Output Layer

Tab. 3 shows the dataset in columns. All of the data are categorical except for the SAMPLE_COUNT, MIN, MAX, INSP_CYCLE and all VALUES. The data were transformed into numerical key pair values and transferred for pre-processing. Here, JUDGE_CODE is the final output of the quality pass report. In this example, we have 10342 instances, of which only 40 instances are faulty. Fig. 4 shows the feature-wise histogram of the data. The dataset is very diverse, with each feature having a relatively wide scale of values. For this reason, the dataset is scaled to normalize the input. The scaled dataset is easier for a machine learning model to process. Feature scaling was done by standardization. It was done in a column-wise manner, as whole-matrix scaling often destroys feature divergence. As the data are highly inconsistent and broad ranging, as seen in Fig. 3, it is important to normalize the data across all columns or data feature spans.

Table 3 Data Column Name and Index

Column Index	Column Name
0	QMSINDEX
1	SAMPLECOUNT
2	INSPMETHOD
3	INSPCYCLE
4	MIN
5	MAX
6	VALUE01
7	JUDGE01
8	VALUE02
9	JUDGE02
10	VALUE03
11	JUDGE03
12	VALUE04
13	JUDGE04
14	VALUE05
15	JUDGE05
16	JUDGECODE

Imbalanced groups are a common problem in machine learning classification where the classes have unequal numbers of observations. The solution to this underlying problem is to re-balance the dataset for proper training. The most famous and frequently used technique is the over- and under-sampling technique. This attempts to rebalance the ratios in an imbalanced dataset of different classes. Natural under-sampling eliminates a percentage of majority cases based on the number of minority instances. In an extremely imbalanced dataset, certain important trends may be discarded behind the majority of instances. Furthermore, the decrease in the total number of cases in the undersized training dataset may limit the recognition capability of the algorithm. In contrast, some randomly selected minority instances replicate over-sampling, which is prone to over-fitting. These make the general balancing techniques impractical for our proposed ensemble. SMOTE was proposed to improve over-sampling at random. It utilizes approximation to create new synthetic cases based on the spatial distribution of existing minority instances [25].

Algorithm 1 Batch-wise Data SMOTE+ENN Re-sampling Technique

Batch $\mathcal{S} = s$
Input: (X, y)
procedure BATCHSMOTENN($\mathcal{S}, X, y$)
  initialize the dataset
  while training is not terminal **do**
    $(X, y) \leftarrow$ generate sub-samples randomly
    $data(\hat{X}, \hat{y}) \leftarrow$SMOTE+ENN$(X, y, S)$
Output: $\leftarrow (\hat{X}, \hat{y})$

Figure 3 Batch-wise SMOTE+ENN Re-sampling Algorithm

This algorithm first takes a minority instance x_0 and makes a list of its k-nearest minority neighbours. The new function values can then be determined from the original and its minority neighbours. SMOTE has shown lasting effectiveness in many different applications, but it is associated with certain cluster initialization problem since it creates new minority instances based on a plurality of the preceding instances. Wilson's ENN rule is a simple yet effective technique that gives re-saliency to label balancing [26]. According to ENN, for each artificial instance, its three nearest neighbours can be found. An instance to which more than two neighbours belong is marked as the predicted class instance. If the predicted class contradicts an instance's actual class, it should be removed. This creates boundaries for SMOTE data generation and ensures a robust dataset that enables ensemble learning. The newly proposed batch-wise SMOTE+ENN method works as described in Algorithm 1. Batch-wise data is generated randomly from the training dataset. The data is then cycled through the hybrid ensemble to learn the underline patterns to classify the different labels. A comparison of the original dataset labels with the dataset labels after application of SMOTE and ENN can be seen in Fig. 3, below.

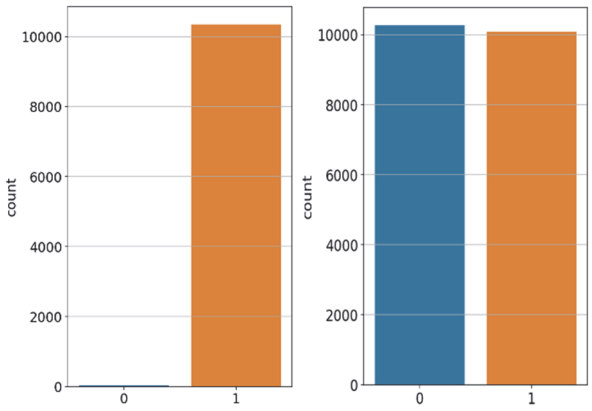

Figure 4 Comparison of the original dataset vs. the dataset after application of proposed SMOTE+ENN method (orange, good products; blue, faulty products)

An edge server-based application is essentially a quick compartmentalized program that is effective in terms of both memory and computation and has the versatility to deploy without a significant degree of reprocessing or

preparation. It also has the ability to scale up as required. This scaling up and load balancing can be achieved by the virtual environment. Nowadays, a virtual environment-based module is a common application-based service deployment medium. A virtual machine (VM)-based server is not bound to traditional hardware limitations and backward compatibility. Most traditional programming languages do not allow for development of this type of application for cloud-based micro-services. On the other hand, Python programming allows for creation of virtual environments and cloning of those environments with relative ease. As a result, developing our proposed algorithm in a Python-based application opened the doorway to development of machine learning services and deployment in a smart factory-based edge micro services system. Fig. 4 shows the multiple creations that arose from single hardware instances in a remote edge where multiple VMs provided separate micro-services.

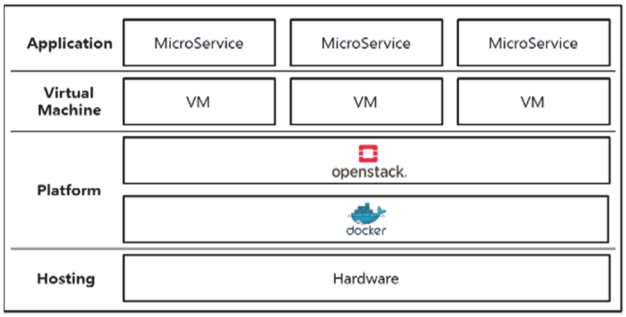

Figure 5 Creation of Multiple Virtual Machines (VM) in a Single Server.

A cloud-based virtual platform for real-time product monitoring and prediction is crucial for automation of modern entities. The edge-based servers and cloud platform enable low latency and high performance in the real-time environmental status monitoring system. This cloud-based module uses the open-source OpenStack framework. This virtual platform, which is created on the server site for multiple instances, gives the program a great deal of flexibility. The data processed and guided application database development in and Python-based program is durable and future-proof in a sense that for future review in the cloud platform based on OpenStack [27] cloud computing instances, it can be easily modified and auto scaled. This enables the whole model to be deployed in the virtual platform-based module Docker [28] and compartmentalized to achieve a proper edge-based servers-based application for factory environments. To facilitate such a scenario, this study proposes a layered architecture with a VM, as shown in Fig. 5, below.

The proposed application has 4 layers, which were created by partial virtualization and physical methods. The bottom layer contains the core data and factory production information; that information is then fed to the Edge layer, which consists of the Edge processor. This data pre-processing program was developed according to the standard set in data pre-processing research. These modules are open stack-based Docker containers that have Python scripts running in the virtual environment. These programs are run and trained on the Edge layer, and the learned model and weight with the final processed data frame are saved in the cloud layer. The information can then be used for future prediction and production visualization. The outermost layer is a virtual layer that can be used to improve predictive analysis and production quality estimation.

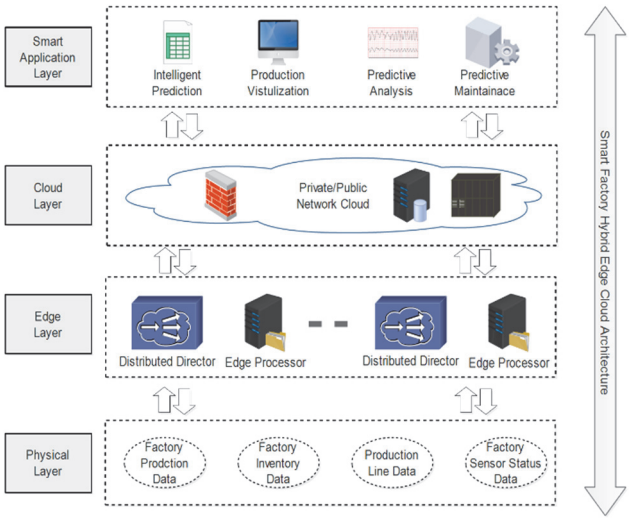

Figure 6 Virtual Environment-Based Multi-layered Quality Estimation System Equipped Smart Factory Architecture.

4 RESULTS

Throughout this paper, the virtual framework used to simulate the proposed process in a smart factory edge-based environment has been thoroughly explained for reproduction. A Python-based virtual environment (Anaconda) creator was used to develop the neural network module. A Python-based Keras [29] with a Tensor flow backend library were then used for neural network design. A dadelta optimization and a categorical cross entropy loss function were chosen to maximize accuracy. The batch size for the neural network was 200, and it was trained on 300 epochs or iterations. The main addition in this ensemble is the introduction of batch normalization [24] to reduce over-fitting and convergence of data during training.

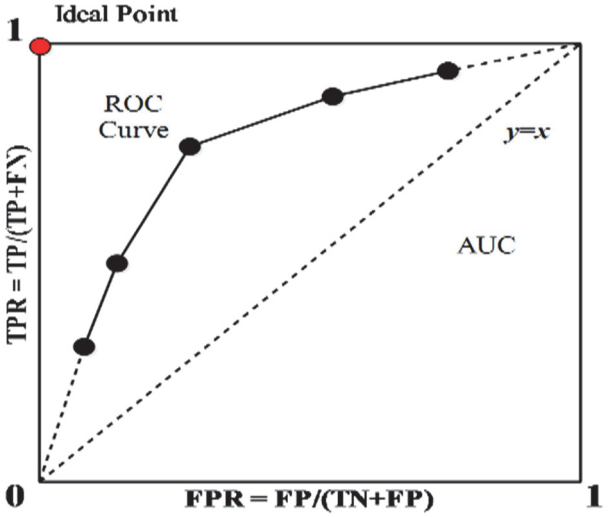

Figure 7 ROC Curve Analysis for Quantifying Performance Evaluation

In this section, the efficacy of the suggested hybrid ensemble is tested by analysing the impact on the overall detection efficiency of different components. Some recommendations for further enhancement of the proposed

detector's efficiency are also given based on the results. It is assumed that accuracy cannot adequately characterize the output of a machine learning model in the sense of an imbalanced classification [30]. This is primarily because both groups are treated with the same care in the precision calculations. Nevertheless, the performance applicable to malicious instances is typically greater with respect to guided production quality estimation detection. Since f_1, the harmonic mean of the precision P and the recall R, can be used to evaluate the ensemble from the perspective of negative or, in this case, faulty automobile parts, it is more suitable as a performance indicator for guided quality estimation systems. TP is the number of true positives, i.e., the number of correctly classified faulty instances. FP is the number of false positives, i.e., the number of normal instances which are wrongfully classified as faulty ones. FN stands for false negatives, which is the number of faulty instances classified as normal ones. The properly labelled cases are denoted as True Negative, TN. Receiving operating characteristic (ROC) curves can be used to compare the general learning capability of different underlying algorithms. In Fig. 6, below, the x-axis shows the false positive rate (FPR) and the y-axis is the true positive rate (TPR) (from Eq. (9) and Eq. (10).

The ROC curve can first be over-sampled by minority instances (if necessary, plurality instances) and then by sequential under-sampling. The optimal value of the ROC curve is (0, 1). This point indicates accurate classification of all positive and negative instances. The algorithm with the largest area under the curve (AUC) is often taken as having the best overall learning power of different learning algorithms. The line $y = x$ is the spontaneous devaluation case. Anything important is discovered by an algorithm with an ROC curve that is below the diagonal from bottom left to the top right, as shown in Fig. 6.

$$FPR = FP/(TN + FP) \tag{9}$$

$$TPR = TP/(TP + FN) \tag{10}$$

As shown in Tab. 4, the f_1 score is only 0.237 in the test case where no data rebalancing is done. This poor result is probably due to insufficient data regarding the bad class in the distribution, as approximately 80% of the malicious instances are overwhelmed by the normal class instances, without misclassification of a single normal instance.

Table 4 Effects of Data Balancing in the Hybrid Ensemble

Algorithm	*TP*	*FP*	*FN*	*TN*	*AC*	*P*	*R*	f_1
Baseline DNN	39	246	5	10031	0.97	0.137	0.886	0.237
SMOTE	31	22	9	10320	0.997	0.585	0.775	0.667
Proposed	39	3	1	10339	0.999	0.928	0.975	0.951

When SMOTE variants are employed, a huge boost is achieved; this is demonstrated by the f_1 score, which verifies the effectiveness of the proposed detector's data re-balancing procedure. However, the data was still somewhat imbalanced, and the learning was not robust enough to use as a trustworthy quality estimation ensemble. Introduction of the proposed batch-wise SMOTE + ENN training to the hybrid model drastically improves the f_1 score and gives the best results with our dataset. Notably, the accuracy of all methods is not severely affected. As a matter of fact, introduction of SMOTE improves the accuracy in exchange for lower recall. This effect can be explained by the offer-fitting of the mode. However, the difference in accuracy between SMOTE and our proposed method is virtually non-existent, but all other metrics, such as recall, precision and f_1 score, drastically improve, further proving the correct evaluation selection. The indifference of the accuracy over learning is readily explained by the definition of accuracy itself. It is simply a ratio of correctly predicted observations to total observations. This does not take class difference into account, thus making it virtually useless in highly imbalanced binary classification problems.

Modern deep learning methods are highly focused on error loss and accuracy improvement [31]. Training accuracy and loss are most often the evaluation metrics used for this type of typical setup. But the detailed experiment of our performance metrics and hybrid ensemble actually contradict the traditional approach. The data shown above prove that this type of training edge-based server mechanism for a general neural network can be rendered useless in a highly imbalanced dataset. So, naturally, from this point forward, a connection should be made between the training and testing f_1 based evaluation metric and the performance analysis to understand how the general neural network training process can be improved based on this result. Moreover, the tuning hyper-parameter in a neural network also relies solely on accuracy. Tuning based on a different metric may further improve the results.

5 DISCUSSION

Modern deep learning research has been focused solely on accuracy-based classification and regression. Given the growth of image-based data, size and computational complexity are also increasing rapidly. In this research, we introduced the classical QES as a deep learning problem. QES is traditionally focused on classical machine learning or statistical process control approaches. So, first, applying a deep learning module is easier as it eliminates the necessity of human intervention for each measurement. Secondly, it speeds up the process by eliminating human intervention and making computation faster. The later sections of the research discussed the main problem of this deep learning-based QES. With an imbalanced dataset, accuracy-based training for neural networks and evaluation is rendered useless, as a deep learning module cannot reliably detect imbalanced classes. On the other hand, modern QES applications are improving, making the proportion of faulty data smaller. Because of the nature of the dataset, a new method of training and testing needs to be developed. Our proposed system takes an imbalanced dataset, makes it into a balanced dataset, and trains the hybrid model to learn from the data to ensure proper detection. This model needs to be evaluated with non-traditional accuracy metrics. So, an evaluation metric was discussed and the high performance of the module was interpreted based on the evaluation criteria. This study also provides insight into the theoretical limits of these kinds of models. A deep learning model essentially learns by examining the variation in the features, and the proposed method achieves its accuracy by utilizing the dataset to generate data from samples; this can

sometimes generate data that has virtually the same features. Naturally, there is a tipping point of the model regarding how long this process can be used to successfully distinguish between different classes. Another interesting application for this learning is reinforcement-based training. Rewards based on an imbalanced dataset will increase the scope of learning of the neural network as compared to a general training strategy. This will pave the way for future work in reinforcement learning-based QES systems in future.

6 CONCLUSION

In the near future, cloud-based edge servers for predicting and monitoring resources will be in high demand due to the ever-growing array of smart markets and factories. Cloud-based real-time process management can be very useful for this type of system. In this research, we proposed a new method of collecting factory-based production quality data and discussed how these data can be applied at scale to implement machine learning algorithms. As the dataset was strongly imbalanced, we launched our novel small-batch data balancing scheme. Moreover, this research described a radical new deep learning neural network ensemble that takes production condition data from a smart factory and predicts production data, making it a truly unbiased QES system. To the best of our knowledge, the results and the proposed algorithm are novel and the system represents the most accurate framework for guided quality estimation that has been developed to date. Accuracy alone cannot adequately characterize the output of a machine learning model when applied to classification of an imbalanced dataset. The evaluation metrics of various models and the future work for this type of model may benefit greatly from our baseline research in this field. Finally, we discussed the implementation strategy and the edge service implementation of the proposed framework.

Acknowlegments

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Centre) support program (IITP-2020-2016-0-00314) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation) and also supported by an Electronics and Telecommunications Research Institute (ETRI) grant funded by the ICT R&D program of MSIT/IITP (2019-0-00260, Hyper-Connected Common Networking Service Research Infrastructure Testbed]. Finally, this research was supported by the National Research Foundation of Korea (NRF) funded by the MSIT (NRF-2018R1A4A1025559)).

7 REFERENCES

[1] Gao, H., Xu, Y., Yin, Y., Zhang, W., Li, R., & Wang, X. (2019). Context-aware qos prediction with neural collaborative filtering for internet-of-things services. *EEE Internet of Things Journal*. https://doi.org/10.1109/JIOT.2019.2956827

[2] Mohammadi, M., Al-Fuqaha, A., Sorour, S., & Guizani, M. (2018). Deep learning for iot big data and streaming analytics: A survey. *IEEE Communications Surveys & Tutorials*. https://doi.org/10.1109/COMST.2018.2844341

[3] Chandra, M. J. (2001). Statistical quality control. *CRC Press*. https://doi.org/10.1201/9781420038675

[4] Yin, Y., Xu, Y., Xu, W., Gao, M., Yu, L., & Pei, Y. (2017). Collaborative service selection via ensemble learning in mixed mobile network environments. *Entropy*, *19*(7), 358. https://doi.org/10.3390/e19070358

[5] Yu, J., Li, J., Yu, Z., & Huang, Q. (2019). Multimodal transformer with multi-view visual representation for image captioning. *IEEE Transactions on Circuits and Systems for Video Technology*. https://doi.org/10.1109/TCSVT.2019.2947482

[6] Wan, J., Zhang, J., Zhou, L., Wang, Y., Jiang, C., Ren, Y., & Wang, J. (2013). Orthrus: a light weighted block-level cloud storage system. *Cluster computing*, *16*(4), 625-638. https://doi.org/10.1007/s10586-012-0234-7

[7] Gao, H., Duan, Y., Shao, L., & Sun, X. (2019). Transformation-based processing of typed resources for multimedia sources in the IOT environment. *Wireless Networks*, 1-17. https://doi.org/10.1007/s11276-019-02200-6

[8] Yin, Y., Yu, F., Xu, Y., Yu, L., & Mu, J. (2017). Network location-aware service recommendation with random walk in cyber-physical systems. *Sensors*, *17*(9), 2059. https://doi.org/10.3390/s17092059

[9] Zini, G. & d'Onofrio, G. (2003). Neural network in hematopoietic malignancies. *Clinica chimica acta*, *333*(2), 195-201. https://doi.org/10.1016/S0009-8981(03)00186-4

[10] Hirose, Y., Yamashita, K., & Hijiya, S. (1991). Back-propagation algorithm which varies the number of hidden units. *Neural Networks*, *4*(1), 61-66. https://doi.org/10.1016/0893-6080(91)90032-Z

[11] Subashini, S. & Kavitha, V. (2011). A survey on security issues in service delivery models of cloud computing. *Journal of network and computer applications*, *34*(1), 1-11. https://doi.org/10.1016/j.jnca.2010.07.006

[12] Zhang, W., Xiong, N., Yang, L. T., Jia, G., & Zhang, J. (2012). Bched-energy balanced sub-round local topology management for wireless sensor network. *Journal of Internet Technology*, *13*(3), 385-394.

[13] Gao, H., Liu, C., Li, Y., & Yang, X. (2020). V2vr: Reliable hybrid-network-oriented v2v data transmission and routing considering rsus and connectivity probability. *IEEE Transactions on Intelligent Transportation Systems*. https://doi.org/10.1109/TITS.2020.2983835

[14] Chen, J., Yang, L., Zhang, Y., Alber, M., & Chen, D. Z. (2016). Combining fully convolutional and recurrent neural networks for 3d biomedical image segmentation. *Advances in neural information processing systems*, 3036-3044.

[15] Wang, M., Song, L., Yang, X., & Luo, C. (2016). A parallel-fusion rnn-lstm architecture for image caption generation. *2016 IEEE International Conference on Image Processing* (*ICIP*). *IEEE*, 4448-4452. https://doi.org/10.1109/ICIP.2016.7533201

[16] Huang, X., Zou, Y., & Wang, Y. (2016). Cost-sensitive sparse linear regression for crowd counting with imbalanced training data. *2016 IEEE International Conference on Multimedia and Expo* (*ICME*). *IEEE*, 1-6. https://doi.org/10.1109/ICME.2016.7552905

[17] Zhu, Y., Zhang, W., Chenand, Y., & Gao, H. (2019). A novel approach to workload prediction using attention-based lstm encoder-decoder networ in cloud environment. *EURASIP Journal on Wireless Communications and Networking*, *2019*(1), 274. https://doi.org/10.1186/s13638-019-1605-z

[18] Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

[19] Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with lstm. *Neural computation*, *12*(10), 2451-2471.

https://doi.org/10.1162/089976600300015015
[20] Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *6*(02), 107-116.. https://doi.org/10.1142/S0218488598000094
[21] Li, M., Zhang, T., Chen, Y., & Smola, A. J. (2014). Efficient mini-batch training for stochastic optimization. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 661-670. https://doi.org/10.1145/2623330.2623612
[22] Kuan, K., Ravaut, M., Manek, G., Chen, H., Lin, J., Nazir, B., Chen, C., Howe, T. C., Zeng, Z., & Chandrasekhar, V. (2017). Deep learning for lung cancer detection: Tackling the kaggle data science bowl 2017 challenge. *arXiv preprint arXiv*: 1705.09435.
[23] Clevert, D. A., Unterthiner, T., & Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprintarXiv*:1511.07289.
[24] Ioffe, S. & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International Conference on Machine Learning*, 448–456.
[25] Chawla, N. V. Bowyer, K. W. Hall, L. O., & Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, *16*, 321-357. https://doi.org/10.1613/jair.953
[26] Batista, G. E., Prati, R. C., & Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDDexplorations newsletter*, *6*(1), 20-29. https://doi.org/10.1145/1007730.1007735
[27] Sefraoui, O., Aissaoui, M., & Eleuldj, M. (2012). Openstack: toward an open-source solution for cloud computing. *International Journal of Computer Applications*, *55*(3), 38-42. https://doi.org/10.5120/8738-2991
[28] Merkel, D. (2014). Docker: lightweight linux containers for consistent development and deployment. *Linux Journal*, *2014*(239), 2.
[29] Chollet, F. (2016). keras, https://github.com/fchollet/keras, 2015, accessed: 2016-12-25.
[30] Wang, J., Sun, Z., Bao, B., & Shi, D. (2019). Malicious synchro phasor detection based on highly imbalanced historical operational data. *CSEE Journal of Power and Energy Systems*, *5*(1), 11-20. https://doi.org/10.17775/CSEEJPES.2018.00200
[31] Sarkhel, R., Das, N., Saha, A. K., & Nasipuri, M. (2016). A multi-objective approach towards cost effective isolated handwritten Bangla character and digit recognition. *Pattern Recognition*, *58*, 172-189. https://doi.org/10.1016/j.patcog.2016.04.010

Contact information:

Akm ASHIQUZZAMAN, M.Sc
Department of ICT Convergence System Engineering,
Chonnam National University, Gwangju, South Korea
E-mail: zamanashiq3@chonnam.ac.kr

Hyunmin LEE, PhD
(Corresponding author)
Human IT Convergence Research Center,
Korea Electronics Technology Institute, South Korea
E-mail: hyunmw@keti.re.kr

Tai-Won UM, PhD, Professor
Department of Cyber Security,
Duksung Women's University, South Korea
E-mail: twum@duksung.ac.kr

Kwangki KIM, PhD, Professor
School of IT Convergence,
Korea Nazarene University, South Korea
E-mail: k2kim@kornu.ac.kr

Hye-Young KIM, PhD, Professor
School of Game/Game Software,
Hongik University, South Korea
E-mail: hykim@hongik.ac.kr

Jinsul KIM, PhD, Professor
(Corresponding author)
Department of ICT Convergence System Engineering,
Chonnam National University, Gwangju, South Korea
E-mail: jsworld@jnu.ac.kr