JOURNAL OF SOCIAL COMPUTING
ISSN 2688-5255 04/06 pp139−149
Volume 4, Number 2, June 2023
DOI: 10.23919/JSC.2023.0011

Developing an Integrated IoT Cloud Based Predictive Conservation Model for Asset Management in Industry 4.0

Karnam Shanmugam*, Kachhti Satyam, and Thimma Reddy Sreenivasula Reddy

Abstract: With the advent of Industry 4.0 (I4.0), predictive maintenance (PdM) methods have been widely adopted by businesses to deal with the condition of their machinery. With the help of I4.0, digital transformation, information techniques, computerised control, and communication networks, large amounts of data on operational and process conditions can be collected from multiple pieces of equipment and used to make an automated fault detection and diagnosis, all with the goal of reducing unscheduled maintenance, improving component utilisation, and lengthening the lifespan of the equipment. In this paper, we use smart approaches to create a PdM planning model. The five key steps of the created approach are as follows: (1) cleaning the data, (2) normalising the data, (3) selecting the best features, (4) making a decision about the prediction network, and (5) producing a prediction. At the outset, PdM-related data undergo data cleaning and normalisation to get everything in order and within some kind of bounds. The next step is to execute optimal feature selection in order to eliminate unnecessary data. This research presents the golden search optimization (GSO) algorithm, a powerful population-based optimization technique for efficient feature selection. The first phase of GSO is to produce a set of possible solutions or objects at random. These objects will then interact with one another using a straightforward mathematical model to find the best feasible answer. Due to the wide range over which the prediction values fall, machine learning and deep learning confront challenges in providing reliable predictions. This is why we recommend a multilayer hybrid convolution neural network (MLH-CNN). While conceptually similar to VGGNet, this approach uses fewer parameters while maintaining or improving classification correctness by adjusting the amount of network modules and channels. The projected perfect is evaluated on two datasets to show that it can accurately predict the future state of components for upkeep preparation.

Key words: Industry 4.0; predictive maintenance; golden search optimization; multilayer hybrid convolution neural network; data cleaning

1 Introduction

Often mentioned to as the "next industrial revolution", Industry 4.0 encompasses the use of smart manufacturing. It rapidly alters procedures and business models in many fields and uses cutting-edge technology to fully materialise smart industrial systems[1]. Industry 4.0 relies heavily on several foundational enabling technologies, including electronic contracts (ECs). Machine learning is used to train replicas services[2, 3], while Internet of Things (IoT) is utilised for real-time data collection, data storage,

• Karnam Shanmugam and Kachhti Satyam are with the Department of Computer Applications, Annamacharya Institute of Technology and Sciences, Karakambadi, Tirupathi 517520, India. E-mail: karnam.shanmugam2019@gmail.com; satyammca32@gmail.com.
• Thimma Reddy Sreenivasula Reddy is with the Department of Computer Science and Engineering, Annamacharya Institute of Technology and Sciences, Karakambadi, Tirupathi 517520, India. E-mail: seenu4linux@gmail.com.
* To whom correspondence should be addressed.
Manuscript received: 2023-03-24; revised: 2023-06-03; accepted: 2023-06-25

and analysis. This data processing pipeline is employed by the vast majority of Industry 4.0 use cases. One of the most common applications of Industry 4.0 is predictive maintenance. However, a major roadblock in commercial settings is the fact that most state-of-the-art AI or machine learning (ML) mockups necessitate massive volumes of data for enhanced accuracy[4]. Data silos form when businesses and manufacturers are reluctant to share information, restricting its use to internal operations. Additionally, AI representations trained on local data are unable to handle unknown scenarios from different contexts[5, 6], and data obtained from a single context are not unavoidably a true reflection of global settings. One of the more recent machine learning (ML) techniques, federated learning (FL), allows for the training of distributed models among a large sum of participants or data silos, without the need for any of the training data to leave, which can be a crucial method for enabling open AI models to be trained on data and devices from all around the world[7].

Industry 4.0's widespread adoption in areas like predictive maintenance has already resulted in cost savings, improved brand image, and even lifesaving outcomes[8]. By combining the CNN and BLSTME, the hybrid deep learning model is created. Based on the relationship between the input images' temporal and spatial components, the models may effectively comprehend[9].

Considering a predictive maintenance system designed for a single type of machine found in numerous production settings, they plan to apply AI to develop a model for predictive machine, etc., it will fail quite training[10]. However, the likelihood of successfully developing a strong predictive model increases greatly if the model is trained using data from a large number of machines (whether they are all built by the same business or not) across many different industries and locations. But most industries do not exchange data because of fears of IP leakage, increased rivalry, and increased insurance claims. A system or arrangement that safeguards the personal information of contributors is beneficial for everyone involved[11].

Computing to achieving Industry 4.0, and here is where "the cloud" comes in. Instead, "the cloud" or remote shared storage is the foundation of cloud computing[12]. The goal of lean management is to eliminate waste at every stage of the value chain from receiving customer orders to fulfilling those orders. Integrating systems both horizontally and vertically and analysing data are crucial for Industry 4.0 enablers. By connecting and integrating business, IT, equipment, operational systems, and gadgets, we may gain a holistic perspective of the value chain[13, 14]. Data privacy and security concerns: with IoT devices constantly collecting and transmitting data to the cloud, there is an increased risk of data breaches and unauthorized access. Sensitive information about assets, operations, and processes may be exposed, leading to potential security vulnerabilities. In this paper, we will look at how Industry 4.0 tools can be utilised to create flexible manufacturing environments, e.g., tools for mapping and navigating, sustainable data fusion, and Slovak national circumstances. The growing popularity of deep learning based techniques can be put to good use in predictive maintenance (PdM).

The primary charities of this study are as trails.

- PdM planning components' future states are anticipated using hybrid deep learning.
- A multilayer hybrid convention neural network (MLH-CNN) is projected to enhance prediction in PdM planning.
- Using the suggested golden search optimization (GSO) algorithm, optimal feature selection is carried out in order to address the issues of overfitting and data redundancy. When determining which features to use, the goal is to pick those that have the lowest correlation with one another.

Industry 4.0 will engulf the entire planet like a tsunami. Future manufacturers will face a significant challenge that calls for extremely reliable gear and equipment. In order to improve the current predictive maintenance approach, Industry 4.0 and its specialised technologies are presented in this article[15].

The remaining parts of this work are structured as follows. It is mentioned in Section 2 that how conventional PdM prediction methods have contributed to the field. Section 3 explains how to use the proposed model in PdM planning. In Section 4, we see the outcomes of comparing the suggested model to the already used methods. Section 5 presents findings and offers recommendations for further study.

2 Related Work

Data collection, feature removal, a prediction model, cloud storage, and analysis are all parts of the decision support system (DSS) proposed by Rosati et al.[16] Unlike previous studies, our approach uses data from both the lower and higher tiers of the production scheme to fuel a feature extraction approach and ML prediction model, which is a significant departure from the status quo. Predictive performance (mean absolute error (MAE): 0.089) and computational effort (average latency of quality) were all found to be optimal by the experiments. These characteristics, together with the machine learning (ML) architecture, allow the operator/maintainer to directly optimise the machining quality procedures. With these advantages, manufacturers can reduce service costs by increasing uptime and productivity while minimising service needs.

In order to facilitate secure knowledge exchange amongst authorised machine tools in the swarm with ultra-low latency performance, Zhang et al.[17] presented a fresh framework for the construction of a knowledge-sharing intelligent machine tool swarm by combining digital twin with multi-access edge computing (MEC). The framework's three primary enablers are MEC-enhanced system deployment, knowledge-based cloud brain learning, and the construction of a digital twin of a machine tool swarm. Finally, a prototype scheme is created, and its effectiveness is shown through assessment studies and usage demonstrations.

Ullah et al.[18] proposed multi-level strategy which shows how federated learning, IoT, and crowdsourcing could work together to produce a self-sustaining federated learning ecosystem well-suited to Industry 4.0. All of these help pave the way for future intelligent applications in the Industry 4.0 sphere. System problem identification and predictive maintenance are two examples of the value of smart manufacturing facilities. Moreover, it showcases a variety of Industry 4.0 applications for multi-level federated learning. If the plan is carried out as intended, it will not only boost performance but also help achieve a broader goal, such as Sustainable Development Goal 13 set by the United Nations, which is to lessen humanity's carbon footprint on the earth.

Converso et al.[19] provided a novel approach for combining machine workload information into a well-established procedure. The projected technique uses a neural network computer model to estimate breakdown likelihood in light of scheduled activities and a logistic regression model to assess the health of equipment. Prototype testing showed that this approach improves forecasting of work completion rates, gives cyber physical systems (CPSs) more leeway in accepting or rejecting jobs based on their expected health status, reduces maintenance costs, and maximises the efficiency of available production resources. This study contributes to the present body of knowledge since it introduces predictive capabilities at the plant shop floor and fully uses the primary enabling knowledges of Industry 4.0.

The standard procedures for predictive maintenance were combined into a single method by Murugiah et al.[20] Because of this, it can be challenging to tackle both the preventative maintenance and prognostic analysis jobs concurrently. For this reason, we propose a new predictive manufacturing system within the framework of Industry 4.0 to analyse the machinery. Data are initially gathered through IoT industry sensors. The multi-scale dilation attention convolutional neural network (MSDA-CNN) cleans the data extensively and extracts deep features. The probabilistic beetle swarm-butterfly optimization approach is then used to optimise the weights of the retrieved deep weighted features. Finally, the deep neural network (DNN) and deep belief network (DBN) perform optimized hybrid fault detection (OHFD), receiving the weighted features as input. In the end, the system notifies industrialists of impending machine failures so they can take preventative measures. Industry 4.0 real-world metrics are used to test the proposed model's efficacy. The majority of in-training techniques in the literature are based on what is known as the a priori methodology from the perspective of multi-objective optimisation (MOO), where the decision-making preference for one objective (the level of fairness) must be specified before optimising the other (accuracy)[21].

Cao et al.[22] have achieved important advancements in the sector with their novel knowledge -based scheme for predictive. The innovative hybrid approach used to develop a knowledge-based system for predictive maintenance in Industry 4.0 (KSPMI) brings together the best features of statistical AI and symbolic AI. The hybrid approach utilises statistical AI technologies like machine learning and text mining to obtain machine deterioration models from industrial data (a form of sequential pattern mining approach). Yet, the recovered chronicle patterns will be used by symbolic AI technologies like domain ontologies and logic rules to query and reason on the system's incoming data with comprehensive domain and contextual knowledge. Using a hybrid approach to ontology reasoning, anomalies in machinery may be detected automatically, and future occurrences can be predicted by merging domain ontologies with semantic web rule language (SWRL) rules derived from chronicle patterns. Using both real-world and synthetic data, we put KSPMI through its paces.

In order to apply predictive maintenance in the injection moulding process, Farahani et al.[23] used cloud and edge computing to integrate the various data sources relevant to this procedure. As an example of the usefulness of the framework, the monitoring of the cooling system for the injection moulding process is presented. These results show that monitoring other process variables outside of mould temperature is an effective way to detect cooling issues. In general, the cloud-based system makes a mean inaccuracy of 3.29% when comparing the predicted mould temperature to the matching sensor value; this is something that may be improved by providing more training data to the scheme.

3 Proposed System

Predictive asset maintenance or PdM is used to optimise maintenance schedules by utilising data-driven algorithms to foresee when assets would break down. Downtime is decreased and product quality is enhanced when PdM is put into practise. Condition-based maintenance or PdM looks for patterns in the deterioration of components over time by analysing past data, therefore it is important to take speedy actions into account. Predicting system failures and fixing components to extend their useful life are two of the more difficult jobs in PdM. More importantly, this strategy is dependent on sensor data collection and processing. There are two ways that condition data are gathered when doing constant monitoring and inspection. Moreover, PdM decision-making necessitates the integration of disparate sources of data, such as work orders, cause and effect chains, monitoring data, and maintenance logs.

To collect the most accurate forecasting information, the projected PdM planning model makes use of datasets. (1) Data cleaning, (2) data normalisation, (3) optimal feature selection, (4) decision-making in the prediction network, and (5) prediction are the five main phases of the proposed model. The datasets are first cleaned by removing anomalies and completing any missing data. Next, normalisation occurs, wherein the cleaned data are placed within a predetermined range (0−1). Optimal feature selection, in which duplicate information is discarded, is conducted following normalisation. As such, we engage in a process of optimal feature selection.

3.1 Data cleaning

Data cleaning entails locating and fixing inconsistencies across various data processes and operations. In this case, data cleaning is accomplished through the elimination of outliers and the addition of missing values.

Outlier detection[24]. In statistics, outliers are equivalent to "noisy data", which tend to obscure meaningful trend. There have been a number of distinct outlier identification methods developed for specific uses. There is a degree of generalizability to each of these techniques. Data outliers are patterns that deviate significantly from the norm. In this case, we utilise the MATLAB function fill outliers to identify outliers and then fill them in with new data. Early on, the outliers are found so that the leftover components can be extracted and decomposed.

Missing data[25]. When there are blanks where numbers should be, we have a missing data problem. More processing time is needed to deal with missing data, and more investigation is needed to understand the issues that arise from them. The fill missing MATLAB function is used to insert a constant value into an array entry that is lacking data. Decomposing residuals and approximations is the job of the function. In order to separate the approximation and residual parts of the elements, a harmonic analysis is performed. The next step is to perform data reconstruction. The approximation part is the primary value, and the residual part is some random noise. A random number is generated by first calculating the standard deviation and the mean of the residual component. Then, the approximation components themselves, or the noise components added together, are used to fill in the blanks. In the end, this model was able to recover data.

The cleaned data are represented as Dt_u, $u = 1, 2, \ldots, ND$, where N is the population size and D is the number of dimensions in the problem space.

3.2 Data normalization

The purpose of data normalisation is to supply "normalised" data, which is a copy of a set of records that is an identical save for a few numeric values. The frequency of events is used to standardise the data. Here, normalisation at the level of individual records yields a representation that is consistent with what we would expect to find in a set of records that is otherwise very comparable to the one being studied. Using field-level normalisation, the most common value for each field in the normalised record is chosen.

Once data cleaning is complete, data normalisation is carried out, yielding a range of numbers as output. Adjusting values unhurried on multiple scales to a notionally common scale, frequently before averaging, is what is meant by the term "data normalisation". The minimum and maximum normalised values, represented by the variables be = 0 and ae = 1, respectively, are considered here. In a nutshell, the equation for normalising data looks like this:

$$\mathrm{Dt}_u^{\mathrm{nrm}} = (\mathrm{ae} - \mathrm{be}) \times \frac{\mathrm{Dt}_u - \mathrm{Dt}^{\min}}{\mathrm{Dt}^{\max} - \mathrm{Dt}^{\min}} + \mathrm{be} \tag{1}$$

Here, we refer to normalised data as $\mathrm{Dt}_u^{\mathrm{nrm}}$, the value to be normalised as Dt_u, and the minimum and supreme values for each greatest value as $\mathrm{Dt}^{\min}$ and $\mathrm{Dt}^{\max}$, respectively.

3.3 Optimal feature selection using GSO

This research proposes the golden search optimization (GSO) algorithm, a straightforward approach to optimization that still achieves impressive results by drawing on the established practises and principles of metaheuristic algorithms. The novel approach strikes a nice balance between global exploration and local exploitation and avoids premature convergence by combining several important features of previously reported algorithms particle swarm optimization (PSO) and sine cosine algorithms (SCA). The objects' positions in the GSO method are updated using a step size parameter that is nearly equal to the velocity in the PSO algorithm. However, the GSO uses sine and cosine functions rather than random numbers. One object can be repositioned around another using the oscillation behaviour of sine and cosine functions, and the area delineated by these two solutions can be used with confidence. Expanding the range of sine and cosine functions, which permit a solution to update its position outside the space between itself and another solution, will further enhance the exploration capability of the algorithm. The algorithm's simplicity belies its superiority over competing metaheuristics when it comes to finding the optimal solution for a given problem in a global context.

Like other population-based metaheuristic optimization methods, GSO starts the search with an apparently arbitrary selection of objects (candidate solutions). To achieve its goal, the algorithm repeats a series of iterations in which the positions of the objects are modified according to a step size parameter. The mathematical expression of the GSO procedures is as follows.

3.3.1 Algorithmic steps

Given that it is a GSO, it should ideally cover the exploration and exploitation phases and be able to find a reasonable balance between these seemingly conflicting skills.

There are three key phases to the algorithm: populace creation, evaluation, and population update. The proposed GSO method is laid out in sequential order below.

Step 1: Population initialization

Using the following equation, GSO generates a random sample of objects (potential solutions) in the search space to begin the search process.

$$O_i = \mathrm{lb}_i + \mathrm{rand} \times (\mathrm{ub}_i - \mathrm{lb}_i), i = 1, 2, \ldots, N \tag{2}$$

where O_i represents the i-th object's position in the search space. The object's lower and upper limits, ub_i and lb_i, are also denoted.

Step 2: Population evaluation

At this point, the fitness of the starting population is calculated using the objective function, and the best object is chosen as Obest_i.

Step 3: Golden change

The third stage involves sorting the objects by fitness and replacing the least fit one with a random alternative.

Step 4: Step size evaluation

The step size operator (St_i) is used to incrementally relocate the objects in the optimization process in order to get them closer and closer to the optimal solution. The St equation is a three-part formula. The first half of this expression is the previous value of the step size multiplied by the transform operator (T), which progressively reduces in order to strike a good compromise between the algorithm's global and local searches. The second section displays, as the cosine of a random number is between 0 and 1, the distance among the i-th object's present position and its best position ever obtained.

The last piece is the sine of a random number between 0 and 1 multiplied by the difference between the i-th object's current position and the best position obtained so far among all objects. Starting with a random value for St_i, which will be changed according to the following equation at each iteration of the optimization procedure:

$$\text{St}_i(t+1) = T\cdot\text{St}_i(t) + C_1\cdot\cos(r_1)\cdot(\text{Obest}_i - x_i(t)) + C_2\cdot \sin(r_2)\cdot(\text{Obest}_i - x_i(t)) \tag{3}$$

where C_1 and C_2 are random statistics in the range [0, 2], r_1 and r_2 are also random numbers in the range [0, 1], and Obest_i is the best previous position obtained by the i-th object local search in late iterations. In reality, T is a declining function, and its value may be calculated by Eq. (4)

$$T = 100\times\exp\left(-20\times\frac{t}{t_{\max}}\right) \tag{4}$$

where $t_{\max}$ is the supreme sum of iterations.

Step 5: Step size limitation

The algorithm advances by a step for each iteration by modifying the distance that each object travels along each dimension of the issue hyperspace. By solving Eq. (3), we see that the step size is a stochastic variable that can affect how closely the objects follow cycles of increasing size. An acceptable interval is introduced to restrict the object's movement according to the divergence:

$$-\text{St}_{i\max} \leqslant \text{St}_i \leqslant \text{St}_{i\max} \tag{5}$$

where $\text{St}_{i\max}$ is the maximum allowable motion, defining the most shift an object can make in its positional coordinates in a single iteration according to the equation:

$$\text{St}_{i\max} = 0.1\times(\text{ub}_i - \text{lb}_i) \tag{6}$$

Step 6: Refresh current status

At this point, the objects are propagating towards the global optimum in the search space in accordance with the following equation:

$$O_i(t+1) = O_i(t) + \text{St}_i(t+1) \tag{7}$$

3.3.2 Time complexity analysis

(1) The effectiveness of a novel optimization method can be assessed in a variety of ways through computational time complexity analysis. The "Big O notation" is a mathematical notation used in computer science to indicate the needed running time of an algorithm taking into account the growth rate when dealing with diverse inputs.

(2) Most algorithms' temporal complexity may be broken down into three distinct parts, each of which can be analysed separately. The time complexity study of the proposed GSO necessitates similar examinations of the following three factors: Complexity of population initialization in time is typically expressed as $O(ND)$, where N is the population size and D is the number of dimensions in the problem space.

(3) Initial fitness assessment time complexity is typically measured in terms of $O(N)F(X)$, where $F(X)$ is the objective function.

(4) Time complexity of the main loop, generally intended by $O(\text{tmax}\times(N\times DCN\times F(X)))$, where tmax is the maximum sum of difficulty of GSO algorithm, is $O(\text{tmax}(N\times DCN\times F(X)))$.

3.4 Classification using MLH-CNN

In order to extract features from images, this article primarily makes use of the convolution layer and the batch normalisation (BN) layer. To boost the network's generalisation capability, shake up the training data, and quicken the model's convergence, the BN[26] layer is employed. Every individual training batch is used to determine the BN. Mean and standard deviation values for each training batch are recorded and then utilised to derive values for the complete training set in the following way:

$$\mu_\beta = \frac{1}{m}\sum_{i=0}^{m} x_i,\ \delta^2 = \frac{1}{m}\left(x_i-\mu_\beta\right)^2 \tag{8}$$

$$E[x] \leftarrow E_\beta\left[\mu_\beta\right], \text{Var}[x] \leftarrow \frac{m}{m-1}E_\beta\left[\delta_\beta^2\right] \tag{9}$$

where μ_β is the mean of the entire dataset, δ is the variance of the entire dataset, $E[\cdot]$ is the feature map of the output, $E_\beta[\cdot]$ is the feature map of the batch size, x is the input to layer one, β is a dataset of batch size m, and m is the small batch size. Each feature map undergoes a batch standardisation, meaning the same process is performed at various locations across all of them. If the feature map has a size of p and q, then BN for this feature map is identical to normalising the feature batch to have a size of $m' = |\beta| = mpq \cdot$BN. To effectively avoid gradient disappearance and explosion, which is independent of the parameters' starting values and has a regularisation impact, BN is chosen.

It was pointed out in VGGNet that the field of view of two 3×3 convolution kernels is equivalent to that of a single 5×5 convolution kernel. Hence, a 3×3 convolution kernel is used to guarantee the perceived field of view while simultaneously decreasing the convolution layer's parameters. Hence, the network architecture of this paper employs the 3×3 convolution kernels.

The schematic for the whole proposed module is shown in Fig. 1. Typical channel counts for these mixers range from 32 to 256.

$$Z_{u,v}^{l} = \sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty} X_{i+u,j+v}^{l-1} \cdot K^{l} \cdot \chi(i,j) + b^{l} \tag{10}$$

$$\chi(i,j) = \begin{cases} 1, & 0 \leqslant i,j \leqslant n; \\ 0, & \text{others} \end{cases} \tag{11}$$

After the convolution layer's output feature has been down-sampled by the BN layer, it moves on to the maximum pooling layer. Each convolution kernel unit now has a weight of l+1, and the output is further augmented by a bias unit b^{l+1}. This is what the layer of sampling produces as its output:

$$Z_{i,j}^{l+1} = \beta^{l+1} \sum_{u=ir}^{(i+1)r-1}\sum_{v=jr}^{(j+1)r-1} a_{u,v}^{l} + b^{l+1} \tag{12}$$

$$a_{u,v}^{l} = f\left(Z_{u,v}^{l}\right) \tag{13}$$

$$a_{i,j}^{l+1} = f\left(Z_{i,j}^{l+1}\right) \tag{14}$$

After the sampling layer comes the convolution layer, now we get the following output:

$$Z_{u,v}^{l+2} = \sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty} a_{i+u,j+u}^{l+1} \cdot K_{i,j}^{l+2} \cdot \chi(i,j) + b^{l+2} \tag{15}$$

where $\chi(i,j)$ is a matrix of order $m \times m$, $K_{i,j}^{l+2}$ is a matrix of order $n \times n$, and $a_{i+u,j+u}^{l+1}$ is a function of $Z_{i+u,j+u}^{l+1}$. The range of (u,v) is $0 \leqslant u,v \leqslant n$.

Following the combination of the modules, the data transitioning from the convolution layer to the complete connection layer are "flattened" by means of a flattening layer. The next step involves employing two full connection layers, with 128 and 64 channels, respectively. While the number of channels may be high, the parameters and total quantity of calculations are greatly diminished. At the end, classification is performed using the Softmax classifier.

While fine-tuning CNN has the potential to improve classification accuracy, they suffer from drawbacks such as high complexity and a huge sum of parameters. In this paper, we begin with a simple and low-complexity convolutional neural network and improve its receptive field of view and network parameters by employing a convolution kernel size of 3×3. Data compression and parameter reduction are accomplished by adding a maximum pool layer basic module, as depicted in Fig. 1. This can help keep the model's generalizability while decreasing the amount of processing required.

There are four separate parts to this network architecture. Little 3×3 convolution kernels with a stride of 1 are used in each convolution layer. Each convolution layer uses 0 padding to prevent the loss of edge feature information that occurs when pixels at the image's corners are skipped over during the convolution procedure. Filters and steps of size 2×2 are used in the maximum pool layer. Ultimately, the output network has a relatively low number of parameters

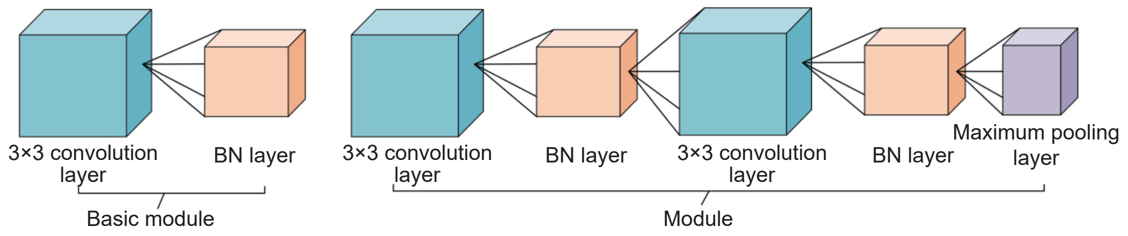

Fig. 1 Construction of the projected modules.

(17 099.26) compared to the CNN.

Selection of optimizer

When it comes to whether or not the training of the network model can converge fast and produce improved accuracy and recall rate, the optimizer plays a crucial role. Adam, Gradient descent, and Momentum are all popular optimizers. Based on the given model, this work primarily examines and compares Adam[27].

Under various conditions, the comparisons are carried out between proposed model, Adam, SGD, and SGDM with Nesterov. Early on, SGD produces the best results, whereas Adam and SGD become steadier as training progresses, and SGDM + Nesterov peaks near the end of the process. As a whole, SGDM + Nesterov performs well in terms of classification accuracy. The SGDM + Nesterov optimizer is clearly superior in terms of accuracy and average iteration time. The suggested model's feature classification is then performed using SGDM + Nesterov as the optimizer.

In order to drastically minimise the sum of parameters and shorten the training period, this work uses input data with a size of 64×64×3. As the optimizer, we have settled on the SGDM, with the momentum parameter set at 0.9, the learning rate at 0.1. The rate of learning is also slowed down, and an early stop mechanism is included. The patience parameter is arranged to 30 in this work. At the present learning rate, a 0.1 reduction in the rate of learning is implemented if the loss function value in the most recent training is not smaller than the value in the most recent training. A batch size of 32 has been selected. The suggested model is built on Keras and trained using NVIDIA's GeForce 940MX graphics processing units. Table 1 presents the values of these hyper-parameter optimizations.

4 Result and Discussion

4.1 Experimental setup

MATLAB 2018a was used to actualize the hybrid DL-based PdM planning method. The effectiveness of the suggested model was measured using datasets from aircraft engines and Li-ion batteries.

Table 1 Value of hyper-parameter optimizations.

Optimizer (momentum parameter)	Value
Batch size	32
Learning rate	0.1
Patient value	30
Batch normalization	Momentum = 0.99, epsilon = 0.001
Number of epochs	100

In the first dataset, known as "plane engines", data were culled from Github[28]. Many multivariate time series data were given, with examples ranging from a single engine to a hundred. There was some variation in the run length, which ranged from 128 cycles to 356 cycles.

In Dataset 2, 18 650 Li-ion cells with a nominal capacity of 2 A·h were charged using a standard CC-CV methodology and discharged in a variety of ways at three distinct temperatures (4 °C, 24 °C, and 43 °C). Terminal current, voltage, and cell temperature were all measured in-cycle, and discharge capacity and EIS impedance were measured between cycles.

4.2 Evaluating models

Accuracy (ACC), precision (PRE), recall (REC), and $F1$-score ($F1$) are used to evaluate the proposed models, which is shown in Eqs. (16)−(19). The following equations are used to regulate these values.

$$\text{Accuracy} = \frac{\text{TP}+\text{TN}}{\text{TP}+\text{FP}+\text{TN}+\text{FN}} \tag{16}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}} \tag{17}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}} \tag{18}$$

$$F1 = \frac{2\cdot\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}} \tag{19}$$

where TP is true positive, TN is true negative, FP is false positive, and FN is false negative.

4.3 Performance analysis of projected model with existing techniques

In this analysis, the validation analysis is carried out with and without GSO model for two datasets. Table 2 presents the comparative analysis of existing models and proposed model without GSO.

When the analysis is carried out, the existing models such as SVM, DBN, AE, RNN, and CNN achieved nearly 78% to 88%, MSDA-CNN achieved 91.22%, and proposed model achieved 92.24% for Dataset 1. When the models are tested with various metrics, the existing model MSDA-CNN achieved 90% to 91% of precision, recall, and $F1$-score, and proposed model achieved 92% of precision, recall, and $F1$-score for Dataset 1. But the same proposed model achieved

Table 2 Analysis of existing models and proposed model without GSO.

Method	Dataset 1				Dataset 2			
	ACC (%)	PRE (%)	REC (%)	*F*1 (%)	ACC (%)	PRE (%)	REC (%)	*F*1 (%)
SVM	78.78	80.44	78.91	78.60	73.15	74.00	73.15	72.97
DBN	78.95	82.70	78.95	78.34	74.17	75.68	75.50	65.70
AE	74.76	78.02	74.76	74.03	81.86	81.88	81.86	81.66
RNN	87.83	87.83	87.83	87.83	81.33	81.60	81.33	81.27
CNN	88.48	88.48	88.48	88.48	82.10	82.63	82.10	82.05
MSDA-CNN	91.22	90.07	91.99	91.99	82.35	82.93	82.35	82.30
MLH-CNN	92.24	92.39	92.24	92.24	83.12	83.85	83.12	83.06

nearly 83% of accuracy, precision, recall, and *F*1-score for Dataset 2, where the existing techniques such as RNN, AE, CNN, and MSDA-CNN achieved nearly 81% to 82% of accuracy, precision, recall, and *F*1-score for Dataset 2. The reason for poor performance is that the whole features including irrelevant and unwanted information are also considered. When the irrelevant features are removed by using GSO, the performance of all techniques is improved, which is shown in Table 3.

The existing models such as SVM, DBN, and AE achieved nearly 84% to 87% of all metrics such as accuracy, precision, recall, and *F*1-score, where the other models such as RNN, CNN, and MSDA-CNN achieved nearly 94% to 96% of accuracy, precision, recall, and *F*1-score for Dataset 1. But the proposed model achieved 98% of accuracy, precision, recall, and *F*1-score for Dataset 1, and also the same model achieved only 92% to 93% of accuracy, precision, recall, and *F*1-score for Dataset 2. When comparing with all existing models, DBN achieved only 77% to 78% of accuracy, precision, recall, and *F*1-score for Dataset 2. The SVM and AE achieved nearly 84% to 86% of accuracy, precision, recall, and *F*1-score, where other models such as RNN, CNN, and MSDA-CNN achieved 90% to 91% of accuracy, precision, recall, and *F*1-score. Figures 2−5 present the graphical analysis of proposed model with existing models.

5 Conclusion

Five steps, including data cleaning, normalisation,

Table 3 Comparative analysis of proposed model with GSO.

Method	Dataset 1				Dataset 2			
	ACC (%)	PRE (%)	REC (%)	*F*1 (%)	ACC (%)	PRE (%)	REC (%)	*F*1 (%)
SVM	87.51	87.63	87.51	87.50	86.09	86.13	86.09	86.09
DBN	84.23	85.10	84.23	84.10	77.91	78.95	77.91	77.43
AE	87.12	87.12	87.12	87.12	84.05	86.79	84.05	82.45
RNN	94.92	94.92	94.92	94.92	90.18	90.18	90.18	90.17
CNN	96.96	96.90	95.71	95.03	91.89	91.90	91.89	91.89
MSDA-CNN	94.89	94.87	94.87	94.87	91.60	92.61	90.60	90.76
MLH-CNN	98.08	98.09	98.08	98.08	92.22	93.23	92.22	93.22

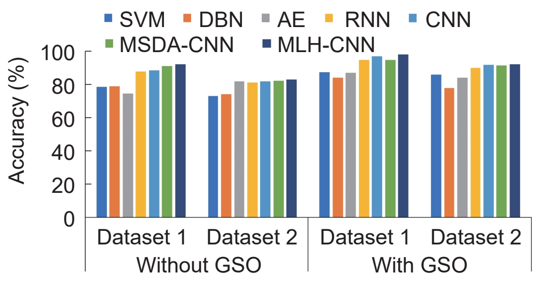

Fig. 2 Accuracy comparison with and without GSO on two datasets.

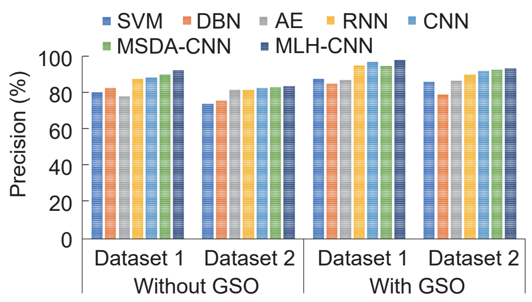

Fig. 3 Precision validation with and without GSO on two datasets.

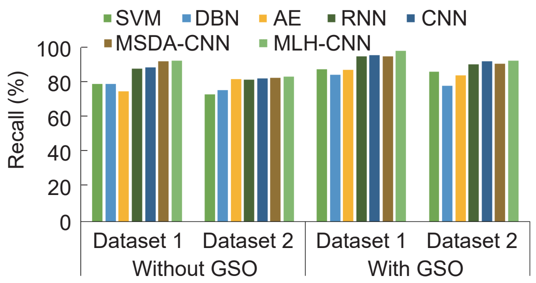

Fig. 4 Recall analysis with and without GSO on two datasets.

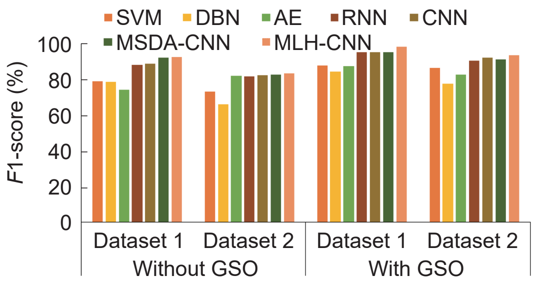

Fig. 5 *F*1-score comparison with and without GSO on two datasets.

optimal feature selection, precision network decision-making, and forecast, make up the intelligent PdM planning model presented in this research. Predictive maintenance helps boost production sustainability by decreasing the frequency of breakdowns, the severity of failures, and the amount of material wasted. To that end, an effective PdM can cut down time and material waste. Two datasets were used in the study: one pertaining to aviation engines and the other to Li-ion batteries. In the beginning, PdM data were cleaned, and then the cleaned data were normalised. The proposed GSO was used to pick features optimally, which helped to cut down redundant information. It was challenging for the DL algorithm to give optimal results because of the wide range of predicted values. To find a network that could deal with a wide variety of prediction values, MLH-CNN was used. Analysis shows that the suggested model outdid the state-of-the-art models by a wide margin, with an accuracy of 98.08% for Dataset 1 and 92.22% for Dataset 2, respectively. Hence, the suggested GSO-MLH-CNN was validated and shown to be effective for PdM planning. Combining multiple DL models is an area for potential improvement over using a single model.

References

[1] M. Nagy and G. Lăzăroiu, Computer vision algorithms, remote sensing data fusion techniques, and mapping and navigation tools in the Industry 4.0-based Slovak automotive sector, *Mathematics*, vol. 10, no. 19, p. 3543, 2022.

[2] H. Nordal and I. El-Thalji, Modeling a predictive maintenance management architecture to meet Industry 4.0 requirements: A case study, *Syst. Eng.*, vol. 24, no. 1, pp. 34–50, 2021.

[3] M. Drakaki, Y. L. Karnavas, P. Tzionas, and I. D. Chasiotis, Recent developments towards Industry 4.0 oriented predictive maintenance in induction motors, *Procedia Comput. Sci.*, vol. 180, pp. 943–949, 2021.

[4] S. Sajid, A. Haleem, S. Bahl, M. Javaid, T. Goyal, and M. Mittal, Data science applications for predictive maintenance and materials science in context to Industry 4.0, *Mater. Today*, vol. 45, pp. 4898–4905, 2021.

[5] A. T. Prihatno, H. Nurcahyanto, and Y. M. Jang, Predictive maintenance of relative humidity using random forest method, in *Proc. 2021 Int. Conf. Artificial Intelligence in Information and Communication* (*ICAIIC*), Jeju Island, Republic of Korea, 2021, pp. 497–499.

[6] A. Sahli, R. Evans, and A. Manohar, Predictive maintenance in Industry 4.0: Current themes, *Procedia CIRP*, vol. 104, pp. 1948–1953, 2021.

[7] R. M. Koushik, A. Perichiappan, H. Om, A. Banerji, S. Eswaran, and P. Honnavalli, Generation of true random numbers using entropy sources present within portable computers, in *Proc. 2021 IEEE Int. Conf. Electronics, Computing and Communication Technologies* (*CONECCT*), Bangalore, India, 2021, pp. 1–6.

[8] Y. K. Teoh, S. S. Gill, and A. K. Parlikad, IoT and fog-computing-based predictive maintenance model for effective asset management in Industry 4.0 using machine learning, *IEEE Internet Things J.*, vol. 10, no. 3, pp. 2087–2094, 2021.

[9] G. Kothai, E. Poovammal, G. Dhiman, K. Ramana, A. Sharma, M. A. AlZain, G. S. Gaba, and M. Masud, A new hybrid deep learning algorithm for prediction of wide traffic congestion in smart cities, *Wirel. Commun. Mob. Comput.*, vol. 2021, p. 5583874, 2021.

[10] E. M. Bourezza and A. Mousrij, Towards a platform to implement an intelligent and predictive maintenance in the context of Industry 4.0, in *Proc. Int. Conf. Artificial Intelligence & Industrial Applications* (*A2IA2020*), Meknes, Morocco, 2020, pp. 33–44.

[11] N. Boddu, V. Boba, and R. Vatambeti, A novel georouting potency based optimum spider monkey approach for avoiding congestion in energy efficient mobile ad-hoc network, *Wirel. Pers. Commun.*, vol. 127, pp. 1157–1186, 2022.

[12] M. Pech, J. Vrchota, and J. Bednář, Predictive maintenance and intelligent sensors in smart factory: Review, *Sensors* (*Basel*), vol. 21, no. 4, p. 1470, 2021.

[13] V. G. Biradar, H. C. Nagaraj, S. G. Mohan, and P. K. Pareek, Industrial fluids components health management using deep learning, https://www.intechopen.com/chapters/85083, 2022.

[14] A. Pollak, S. Temich, W. Ptasiński, J. Kucharczyk, and D. Gąsiorek, Prediction of belt drive faults in case of predictive maintenance in Industry 4.0 platform, *Appl. Sci.*, vol. 11, no. 21, p. 10307, 2021.

[15] K. Aksa, S. Aitouche, H. Bentoumi, and I. Sersa, Developing a web platform for the management of the predictive maintenance in smart factories, *Wirel. Pers. Commun.*, vol. 119, no. 2, pp. 1469–1497, 2021.
[16] R. Rosati, L. Romeo, G. Cecchini, F. Tonetto, P. Viti, A. Mancini, and E. Frontoni, From knowledge-based to big data analytic model: A novel IoT and machine learning based decision support system for predictive maintenance in Industry 4.0, *J. Intell. Manuf.*, vol. 34, no. 1, pp. 107–121, 2023.
[17] C. Zhang, G. Zhou, J. Li, F. Chang, K. Ding, and D. Ma, A multi-access edge computing enabled framework for the construction of a knowledge-sharing intelligent machine tool swarm in Industry 4.0, *J. Manuf. Syst.*, vol. 66, pp. 56–70, 2023.
[18] I. Ullah, U. U. Hassan, and M. I. Ali, Multi-level federated learning for Industry 4.0—A crowdsourcing approach, *Procedia Comput. Sci.*, vol. 217, pp. 423–435, 2023.
[19] G. Converso, M. Gallo, T. Murino, and S. Vespoli, Predicting failure probability in Industry 4.0 production systems: A workload-based prognostic model for maintenance planning, *Appl. Sci.*, vol. 13, no. 3, p. 1938, 2023.
[20] P. Murugiah, A. Muthuramalingam, and S. Anandamurugan, A design of predictive manufacturing system in IoT-assisted Industry 4.0 using heuristic-derived deep learning, *Int. J. Commun. Syst.*, vol. 36, no. 5, p. e5432, 2023.
[21] S. Liu and L. N. Vicente, Accuracy and fairness trade-offs in machine learning: A stochastic multi-objective approach, *Comput. Manag. Sci.*, vol. 19, no. 3, pp. 513–537, 2022.
[22] Q. Cao, C. Zanni-Merk, A. Samet, C. Reich, F. D. B. D. Beuvron, A. Beckmann, and C. Giannetti, KSPMI: A knowledge-based system for predictive maintenance in Industry 4.0, *Robotics Comput. Integr. Manuf.*, vol. 74, p. 102281, 2022.
[23] S. Farahani, V. Khade, S. Basu, and S. Pilla, A data-driven predictive maintenance framework for injection molding process, *J. Manuf. Process.*, vol. 80, pp. 887–897, 2022.
[24] K. Singh and S. Upadhyaya, Outlier detection: Applications and techniques, *Int. J. Comput. Sci.*, vol. 9, no. 3, pp. 307–323, 2012.
[25] W. Deng, Y. Guo, J. Liu, Y. Li, D. Liu, and L. Zhu, A missing power data filling method based on improved random forest algorithm, *Chin. J. Electr. Eng.*, vol. 5, no. 4, pp. 33–39, 2019.
[26] S. Ioffe and C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in *Proc. 32nd International Conference on Machine Learning*, Lille, France, 2015, pp. 448–456.
[27] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv: 1412.6980, 2014.
[28] A. Desai, Predictive maintenance (PdM) of aircraft engine, https://github.com/archd3sai/Predictive-Maintenance-of-Aircraft-Engine, 2022.

Karnam Shanmugam received the BSc degree in computer science from Sri Venkateswara University, Tirupathim, India in 2002, and the Master of Computer Applications (MCA) degree in computer applications from Sathyabama University, Tamil Nadu, India in 2005. He is an assistant professor at Annamacharya Institute of Technology and Sciences, Karakambadi, Tirupathi, India. He is a member of ISTE and IAENG. His current research includes machine learning, IoT, and data mining.

Kachhti Satyam received the BSc degree in computer science from Acharya Nargarjuna University, Guntur, Andhra Pradesh, India in 2007, and the Master of Computer Applications (MCA) degree in computer applications from Acharya Nargarjuna University, Guntur, Andhra Pradesh, India in 2010. He is an assistant professor at Annamacharya Institute of Technology and Sciences, Karakambadi, Tirupathi, India. He is a member of IAENG. His current research includes machine learning techniques, IoT, and artificial intelligence.

Thimma Reddy Sreenivasula Reddy received the BSc degree in computer science from Sri Venkateswara University, Tirupathi, India in 2001, the Master of Computer Applications (MCA) degree in computer applications from Madurai Kamaraj University, Tamil Nadu, India in 2007, and the Master of Technology (MTech) degree in computer science and engineering from Jawaharlal Nehru Technological University Anantapur (JNTUA), Ananthapuramu, India in 2010. He has been pursuing the PhD degree at the Department of Computer Science and Engineering, Annamalai University, Annamalai Nagar, Tamil Nadu, India since 2018. He is an assistant professor at Annamacharya Institute of Technology and Sciences, Karakambadi, Tirupathi, India. He is a member of ISTE, IAENG, and CSTA. His current research includes data mining.