applied sciences

MDPI

Article

Optimizing the Parameters of Long Short-Term Memory Networks Using the Bees Algorithm

Nawaf Mohammad H. Alamri *, Michael Packianather and Samuel Bigot

School of Engineering, Cardiff University, Queen's Buildings, 14-17 The Parade, Cardiff CF24 3AA, UK
* Correspondence: alamrinm@cardiff.ac.uk

Abstract: Improving the performance of Deep Learning (DL) algorithms is a challenging problem. However, DL is applied to different types of Deep Neural Networks, and Long Short-Term Memory (LSTM) is one of them that deals with time series or sequential data. This paper attempts to overcome this problem by optimizing LSTM parameters using the Bees Algorithm (BA), which is a nature-inspired algorithm that mimics the foraging behavior of honey bees. In particular, it was used to optimize the adjustment factors of the learning rate in the forget, input, and output gates, in addition to cell candidate, in both forward and backward sides. Furthermore, the BA was used to optimize the learning rate factor in the fully connected layer. In this study, artificial porosity images were used for testing the algorithms; since the input data were images, a Convolutional Neural Network (CNN) was added in order to extract the features in the images to feed into the LSTM for predicting the percentage of porosity in the sequential layers of artificial porosity images that mimic real CT scan images of products manufactured by the Selective Laser Melting (SLM) process. Applying a Convolutional Neural Network Long Short-Term Memory (CNN-LSTM) yielded a porosity prediction accuracy of 93.17%. Although using Bayesian Optimization (BO) to optimize the LSTM parameters mentioned previously did not improve the performance of the LSTM, as the prediction accuracy was 93%, adding the BA to optimize the same LSTM parameters did improve its performance in predicting the porosity, with an accuracy of 95.17% where a hybrid Bees Algorithm Convolutional Neural Network Long Short-Term Memory (BA-CNN-LSTM) was used. Furthermore, the hybrid BA-CNN-LSTM algorithm was capable of dealing with classification problems as well. This was shown by applying it to Electrocardiogram (ECG) benchmark images, which improved the test set classification accuracy, which was 92.50% for the CNN-LSTM algorithm and 95% for both the BO-CNN-LSTM and BA-CNN-LSTM algorithms. In addition, the turbofan engine degradation simulation numerical dataset was used to predict the Remaining Useful Life (RUL) of the engines using the LSTM network. A CNN was not needed in this case, as there was no feature extraction for the images. However, adding the BA to optimize the LSTM parameters improved the prediction accuracy in the testing set for the LSTM and BO-LSTM, which increased from 74% to 77% for the hybrid BA-LSTM algorithm.

Keywords: Deep Learning (DL); Long Short-Term Memory (LSTM); Convolutional Neural Network (CNN); Bees Algorithm (BA); Bees Algorithm Convolutional Neural Network Long Short-Term Memory (BA-CNN-LSTM)

Citation: Alamri, N.M.H.; Packianather, M.; Bigot, S. Optimizing the Parameters of Long Short-Term Memory Networks Using the Bees Algorithm. *Appl. Sci.* **2023**, *13*, 2536. https://doi.org/10.3390/app13042536

Academic Editor: Emilio Soria-Olivas

Received: 17 January 2023
Revised: 13 February 2023
Accepted: 14 February 2023
Published: 16 February 2023

1. Introduction

Artificial intelligence (AI) facilitates intelligent systems improvement and leads to an increase in the effectiveness and efficiency of processes [1]. AI has different techniques that can be applied in different contexts in real life, and the most popular ones are Artificial Neural Networks (ANNs), which are inspired by the human brain. ANNs can model high dimensional data, extract implicit patterns, and investigate complex relationships to predict a complex systems' future state [2]. Additionally, they can deal with nonlinear dynamic problems [3]. Deep Learning (DL) is an extension of ANN with better learning capability,

as it extracts the features automatically using nonlinear filters [4]. It can be applied in different fields, such as in manufacturing, thereby enabling advanced analytics for big data that result in smart manufacturing systems, whether during processing by diagnosing processes conditions to predict the future state of the products, or during postprocessing to the finished parts [5]. In addition, it can be applied in the pharmaceutical field to diagnose cancers and predict aqueous solubility [6]. On the other hand, there are some disadvantages of adopting DL. The most important one is the acquisition of datasets, as DL requires a large amount of data, so the availability is not guaranteed. In addition, applying data preprocessing for big data is challenging, especially if many irrelevant observations need to be removed, and applying improper data mining may affect the performance of the DL model [3].

Long Short-Term Memory (LSTM) is one of the DL networks that deals with time series or sequential data. It is an advanced Recurrent Neural Network (RNN) that retains information for a long period so that it can remember long-term dependencies [7]. The RNN can retain the information as well, but for a shorter period, so it cannot remember the dependencies in the long term. LSTM was established to address this problem in the RNN [8]. It consists of three gates: the first one is the forget gate that decides if it is necessary to remember the information coming from the previous time scale or not; the second one is the input gate that learns information from the cell input; and the last gate is the output gate, wherein the cell transfers the updated information to the next time cycle. In addition, it has a cell state that carries the information, along with all cycles, and a hidden state for short-term memory [9].

Improving the performance of the LSTM network is an ongoing challenge [10]. The traditional approach of assigning the parameters using trial and error is less accurate, since it depends on the user experience. However, using a nature-inspired algorithm to automatically select the optimum parameter values may improve the performance of the network [10]. The existing studies addressed optimizing hidden layers, the number of neurons, activation function, loss function, optimizers, batch size, and the number of epochs using different nature-inspired algorithms, as will be shown in Section 2.3. This study addresses optimizing the learning rate in each of the three gates and the cell candidate so that each step has a customized learning rate for more optimum performance.

In particular, the weights update in each of the three gates and cell candidate depends on the learning rate value [9], so using a customized learning rate for each part would result in more optimal updates for the weights [10]. A study proposed a nature-inspired algorithm that utilized the power of global, local, and intense searches in the BA to optimize the learning rate factor for the convolutional layers and the fully connected layer in order to adjust the global learning for a CNN, which resulted in a more optimum weights update [10]. When the proposed hybrid algorithm was applied to 'Cifar10DataDir', the validation accuracy was improved from 80.72% to 82.22% [10].

Hence, the paper aims to improve the performance of the LSTM in dealing with sequential problems by optimizing the parameters related to the three gates and cell state. This paper proposes a hybrid nature-inspired algorithm, which takes the advantages of the Bees Algorithm (BA), in order to improve the performance of the three gates and cell state. In particular, the BA was used to optimize the learning rate factor so that each part had its own learning rate that was determined based on the global learning rate. Having a more optimum learning rate means more optimum updates for the network weight [11]. Artificial porosity images were used for testing the algorithms; as the input data are images, a Convolutional Neural Network (CNN) was added in order to extract the features in the images and feed the LSTM to predict the porosity in sequential layers of artificial porosity images that mimicked real CT scan images of products manufactured by the Selective Laser Melting (SLM) process. The hybrid Bees Algorithm Convolutional Neural Network Long Short Term Memory, referred to as the BA-CNN-LSTM, combines the best DL networks for dealing with images and sequential data, as the CNN is the most powerful DL network for analyzing images [4]. In addition, the LSTM network is the best algorithm for dealing

with sequential data, as it retains the information for a long period by remembering long-term dependencies, which address the problem of normal RNNs [7]. It can be applied in other contexts, such as in signal processing to classify electrocardiogram (ECG) benchmark image data, as will be shown in Section 4. Also, it can deal with time series numerical data, wherein the CNN is not needed in this case, as there is no feature extraction task for the images.

The contribution of the paper is improving the performance of LSTM network in predicting sequential data using BA. As the input data are images, CNN is added to extract the image features yielding a hybrid algorithm (BA-CNN-LSTM) that provides a more accurate prediction of porosity percentage appearing in sequential layers of artificial porosity images that mimic CT scan images of parts manufactured by SLM process.

The paper structure starts with Section 2, which presents a review of the LSTM while showing the gaps that need to be fulfilled to improve its performance. Then, Section 3 describes the process for the proposed BA-CNN-LSTM, and Section 4 shows the results and discussion of the proposed model. Section 5 presents the study conclusion and future recommendations.

2. Long Short-Term Memory (LSTM)

This section shows a review of LSTM that presents the definition and process, the gaps and open issues, and state-of-the-art studies.

2.1. LSTM Definition and Process

The LSTM is one of the DL networks that deals with time series or sequence problems. It is an extension of RNNs that can remember long-term dependencies, as it retains the information for a long period. The normal RNN is not able to do so, thus resulting in a vanishing gradient problem [9]. This is a situation where the RNN is not able to propagate useful information from the end of the network back to the beginning of the network, as the information is stored only for a short period [12].

The structure of the LSTM consists of three gates: the first one is the forget gate that decides if the information coming from the previous time scale is relevant to be remembered or irrelevant to be forgotten; the second one is the input gate that tries learning new information from the cell input; and the last gate is the output gate, wherein the cell transfers the updated information from the previous time cycle to the next time cycle. In addition, it has a cell state that carries the information along with all cycles as it stores the information for a long period and has a hidden state for short-term memory; this cell state is present in RNN as well [9]. Reference [9] presented the following Figure 1, which illustrates the basic structure of the LSTM, where the first part is the forget gate, the second part is the input gate, and the last one is the output gate. In addition, C_{t-1} is the cell state for the previous time cycle, C_t is the cell state for the current time cycle, H_{t-1} is the hidden state for the previous time cycle, and H_t is the hidden state for the current time cycle.

The LSTM network starts with the forget gate to decide if the information coming from the previous time scale is relevant to be remembered or irrelevant to be forgotten. Then, the input gate is used to quantify the importance of the input of new information. After that, the cell transfers the updated information from the previous time cycle to the next time cycle in the output gate.

Reference [8] presented a more intuitive architecture of the LSTM network, as is shown in Figure 2.

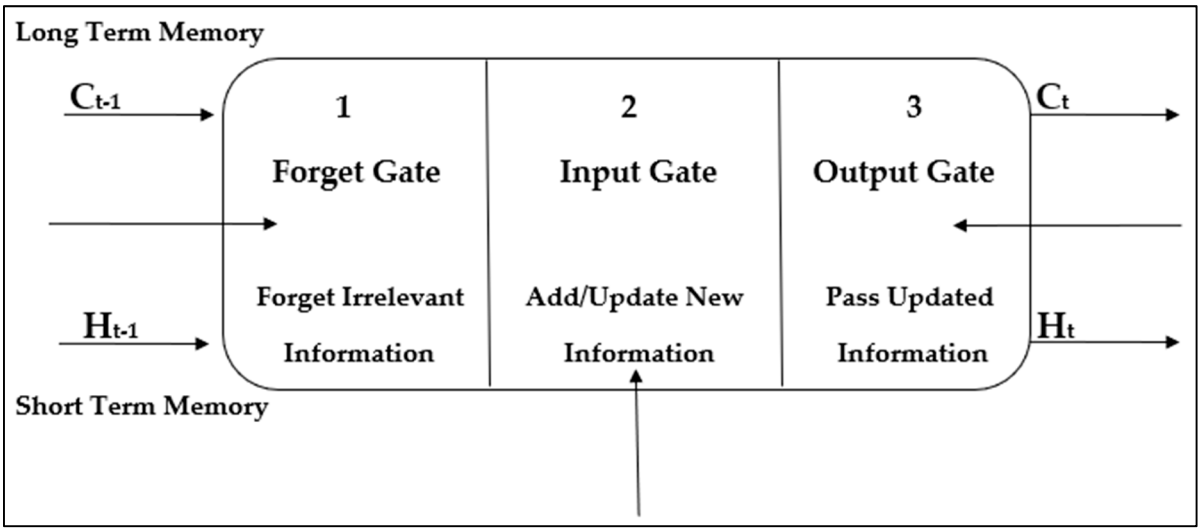

Figure 1. The basic structure of the LSTM [9].

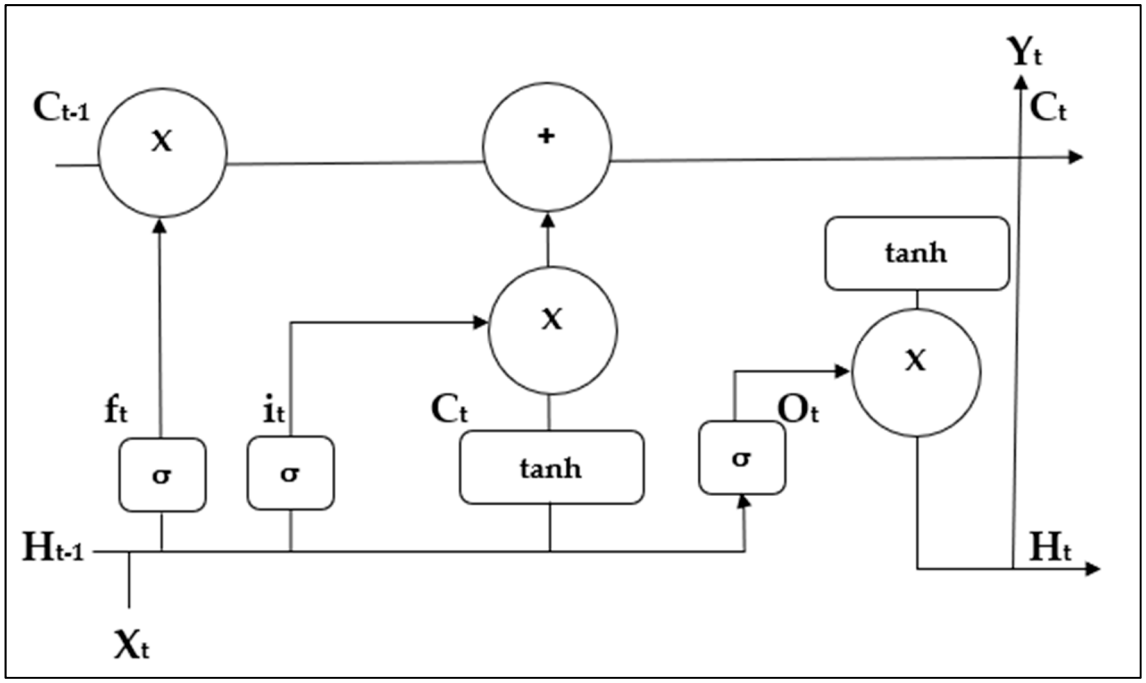

Figure 2. Intuitive diagram of the LSTM [8].

2.2. Gaps and Open Issues

One of the most important challenges in training the LSTM is reducing overfitting [13], which is the case where the model fits well enough for training data and performs poorly in validation and testing data so that it is not able to generalize to unseen data [14]. This issue can be overcome by adding a dropout layer [15,16], traying more optimal regularization values [17], reducing the number of epochs, or augmenting the datasets [13].

The exploding and vanishing gradient is a problem that occurs in RNNs when the model stops learning after a specific epoch, and the LSTM addresses this issue [9]. However, the optimum weight change can be improved further by having a more optimum learning rate as the gradient is multiplied by the learning rate, which results in the optimum set of weights [18]. Thus, having a more optimum learning rate means more optimum updates for the network weight [11].

Furthermore, improving the performance of the LSTM network is an ongoing challenge [19], as there is no approach found yet that develops the best architecture. However, adopting nature-inspired algorithms for optimizing the LSTM parameters may develop the model performance, as it reduces the need for human input when assigning the parameters [20]. The following section will present state-of-the-art studies on using nature-inspired algorithms to optimize LSTM parameters.

2.3. State-Of-The-Art Studies

Nature-inspired algorithms have a great impact on optimizing the parameters of the DL models. The need for trial and error when selecting the parameters is reduced, as the nature-inspired algorithms recognize the parameters automatically [21].

The Genetic Algorithm (GA) is one of the nature-inspired algorithms used to find the optimal parameters in the LSTM network for predictive maintenance [22]. The authors optimized the time steps, the number of LSTM layers, and the number of hidden neurons in each layer by using the GA. They suggested a procedure that starts with population initialization, followed by fitness computation for chromosomes, and finally concludes with genetic operator applications for new population creation, if needed. The design of the GA is based on chromosome structure, fitness function, crossover operators, mutation operators, and population updating and termination [22]. The proposed model achieved an accuracy of 98.14%.

Similarly, another study used the GA to optimize five parameters related to LSTM hidden layer size, the number of hidden layers, batch size, the number of times steps, and the number of epochs. The hybrid GA-LSTM was used to predict the next word in the sentence, which achieved an accuracy of 56% [23].

Furthermore, a swarm-based metaheuristic optimization algorithm called Particle Swarm Optimization (PSO) was applied to improve the performance of the LSTM network by optimizing hidden layers, the number of neurons, activation function, the loss function, optimizers, batch size, and the number of epochs. The hybrid PSO-LSTM was applied to predict a pollution level based on a weather dataset; it achieved a lower RMSE than the original LSTM by a value of 0.0007 [24].

A study used the Artificial Bee Colony (ABC) to optimize the weights of an ANN [25]. The author used the hybrid algorithm to propose a new intrusion detection system that achieved an accuracy of 95.02% [25]. Another study used the BA to train an RNN for sentiment classification, and it improved the accuracy from 60% for the traditional RNN to 90% for the BA-RNN algorithm [26]. Additionally, the ABC was used to optimize LSTM parameters (window size, LSTM units, dropout probability, number of epochs, batch size, and global learning rate) [27]. They used the hybrid ABC-LSTM algorithm for stock market prediction, which achieved a lower RMSE by 5.6836 [27].

In addition, the LSTM can be integrated with another DL network through the CNN, which is used if the input data are images in order to extract features and feed LSTMs that deal with sequential data [28]. A hybrid CNN-LSTM analyzed motion data in a video to recognize unsafe actions performed by workers. Only applying the CNN resulted in an accuracy of 82%, but adding the LSTM improved the model accuracy to 97%, as it stored the information for a long period, which made it possible to consider the long-term dependencies [28]. The hybrid CNN-LSTM was integrated as well with the ABC to optimize the type of network, the number of epochs, the LSTM hidden units, the global learning rate activation function, the step size, the fully connected layer, and the pooling size [29]. The authors used the hybrid ABC-CNN-LSTM algorithm to detect fake reviews of a product and resulted in an accuracy of 97% compared to 95% for the CNN-LSTM algorithm [29].

The following Table 1 summarizes the state-of-the-art studies. It shows the purpose of the study, along with the available model accuracy or RMSE difference between the original and hybrid algorithms.

Table 1. State-of-the-art studies.

Hybrid Algorithm	Purpose	Original Algorithm Accuracy	Hybrid Algorithm Accuracy	RMSE Difference	Reference
GA-LSTM	Predictive maintenance	-	98.14%	-	[22]
	Word prediction	-	56%	-	[23]
PSO-LSTM	Pollution level prediction	-	-	0.0007	[24]
ABC-ANN	New intrusion detection	-	95.02%	-	[25]
BA-RNN	Sentiment classification	60%	90%	-	[26]
ABC-LSTM	Stock market prediction	-	-	5.6836	[27]
CNN-LSTM	Worker unsafe action recognition	82%	97%	-	[28]
ABC-CNN-LSTM	Fake review detection	95%	97%	-	[29]

When looking at the literature, there is a lack of optimization of the learning rate adjustment factor for each gate in the LSTM network, which is an important gap, as the learning rate controls the weight update. Having a more optimum learning rate means more optimum updates for the network weight [11]. The following Section 3 will explain a hybrid BA-CNN-LSTM algorithm that addresses this gap in detail.

3. Proposed Novel Hybrid BA-CNN-LSTM Algorithm

This section describes a novel approach to improve the performance of the LSTM by optimizing the parameters related to the gates and cell state. The novel hybrid nature-inspired algorithm adopted the BA to improve the performance of three gates and the cell state, which it does specifically by optimizing the learning rate factor so that each part has its own learning rate that is determined based on the global learning rate. Having a more optimum learning rate means more optimum updates for the network weight [10]. The bidirectional LSTM layer was used, as will be explained later in Section 3.2, which trains the input as is and on the reverse copy of the input [30], so four parameters are related for the forward side, the other four parameters are related for the backward side, and one parameter is related to the fully connected layer. The following Figure 3 illustrates the general framework for the proposed BA-CNN-LSTM algorithm, followed by subsections explaining each part in detail.

3.1. CNN Architecture

The features in artificial porosity images were extracted using the CNN, which uses filters in the convolutional layer to activate some features in the images. The architecture of the CNN consists of nineteen layers, starting with five convolutional layers that are each accompanied by a rectified linear unit layer and a batch normalization layer with four average pooling layers in between.

The filter in the convolutional layers is a matrix that contains weights. This matrix is multiplied by the input matrix that will be described in the next section, which results in a matrix called a feature map [31]. Each convolutional layer is accompanied by a rectified linear unit layer that maps the negative values to zero and keeps the positive values for faster effective training [32]. Also, each convolutional layer is followed by a batch normalization layer to minimize the issue of overfitting [33]. The pooling layers come between the convolutions in order to reduce the output dimensions while keeping the features of interest in the images [34]. This reduction contributes to minimizing the computational cost.

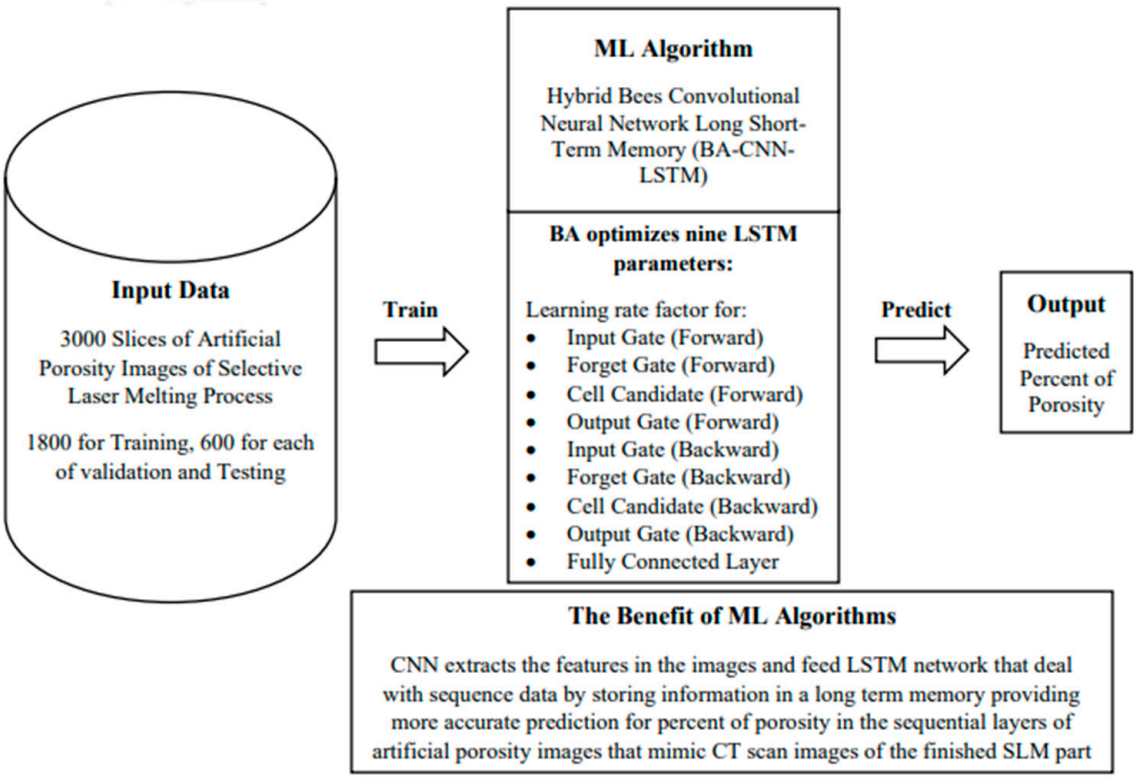

Figure 3. The general framework for the proposed BA-CNN-LSTM algorithm.

The normal filter size is 3 × 3 or 5 × 5, as advised in [32]. After trying both sizes, the 5 × 5 size was selected for all five convolutional layers, and the number of filters was 8, 16, 32, 64, and 128, so each layer is double the previous filter number, as advised in [10]. The edges of the images were detected using padding with a stride value of one, so the pixel shift is one cell. The pooling size is 4 × 4 with a four-stride value to reduce the computational time, since the image size is large (650 × 630 × 3). There are two types of pooling. Average pooling calculates the average pixel brightness value of the four numbers in the pooling matrix. It is compatible with lighter backgrounds, so it was used in the architecture rather than the max pooling type, which selects the maximum pixel value by taking the most activated feature, which works better with a dark background [35].

The most popular training algorithm is the stochastic gradient descent with momentum (SGDM) [33], so it was used to train on the artificial porosity images with 20 epochs [36]. A section depth value of 1 was selected for network depth controlling, with a 0.0101 initial learning rate value that allowed for feature learning, a 0.9568 momentum value for parameters updating, and a 0.0097 regularization value to mitigate the overfitting risk [37]. They were the same set of parameters found using the BA in the hybrid Bees Algorithm Regression Convolutional Neural Network (BA-RCNN) developed in [38]. The hybrid BA-RCNN was used to predict the porosity percentage in each layer of the SLM part. Since the layers are sequential, the LSTM was added after the CNN to deal with sequential data for better prediction accuracy, as will be described in the following subsection.

3.2. LSTM Architecture

The features were extracted from the artificial porosity images using the CNN to feed the LSTM, which retains the information for a long period so that it can remember long-term dependencies [7]. The following points are detailed mathematical explanations of the LSTM network.

- Forget Gate

The first step in the LSTM cell is to decide if the information coming from the previous time scale is relevant to be remembered or irrelevant to be forgotten. It is based on the following forget gate equation [9]:

$$f_t = \sigma \times (X_t \times U_f + H_{t-1} \times W_f) \quad (1)$$

where:

X_t is the current time cycle input;
U_f is the input weight;
H_{t-1} is the previous time cycle hidden state;
W_f is the hidden state weight matrix.

Then, the following sigmoid function is applied, resulting in a f_t value between 0 and 1 [39].

$$\sigma(x) = (1 + e^{-x})^{-1} \quad (2)$$

The f_t is multiplied by the cell state of the previous time cycle:

$$C_{t-1} \times ft = 0 \text{ (Forget everything)} \quad (3)$$

$$C_{t-1} \times ft = C_{t-1} \text{ (Forget nothing)} \quad (4)$$

- Input Gate

This gate is used to quantify the importance of the new input information. It is based on the following input gate equation [9]:

$$i_t = \sigma \times (X_t \times U_i + H_{t-1} \times W_i) \quad (5)$$

where:

X_t is the current time cycle input;
U_f is the input weight matrix;
H_{t-1} is the previous time cycle hidden state;
W_f is the hidden state weight matrix.

Similarly, the sigmoid function is applied, resulting in an i_t value between 0 and 1. Passing the information to the cell state is based on a function of the hidden state of the previous time cycle [24]:

$$N_t = \tanh(X_t \times U_c + H_{t-1} \times W_c) \text{ (new information)} \quad (6)$$

Using the tanh function results in a N_t value between -1 and 1. If it is positive, it will be added to the cell state, and, if it is negative, the information will be subtracted from the cell state, as shown in the following equation [24]:

$$C_t = f_t \times C_{t-1} + i_t \times N_t \text{ (updating cell state)} \quad (7)$$

- Output Gate

In this gate, the cell transfers the updated information from the previous time cycle to the next time cycle. It is based on the following output gate equation [9]:

$$O_t = \sigma \times (X_t \times U_o + H_{t-1} \times W_o) \quad (8)$$

Likewise, the O_t value is between 0 and 1, because the sigmoid function has been applied. The current hidden state is a function of the long-term memory and the output, and it is calculated based on the following equation:

$$H_t = O_t \times \tanh(C_t) \quad (9)$$

The output of the current time cycle is found using the SoftMax function, as shown in the following equation [23]:

$$\text{Output} = \text{SoftMax}(H_t) \quad (10)$$

The output with the maximum score is the predicted value.

The above-described mechanism is used to provide a more accurate prediction of the porosity percentages appearing in sequential layers of artificial porosity images [38] that mimic CT scan images of parts manufactured by the SLM process. The created artificial porosity images in [38] consist of 30 3D cubes. Each one was sliced into 100 2D sequential slices, thus resulting in 3000 slices. Therefore, there are 30 sequences with 100 layers in each sequence. A total of 18 sequences were used for the network training task, and 6 sequences were used for the validation set, as well as 6 for the testing set. Reference [38] presented an example of artificial porosity images that mimic the real porosity images with a similarity index of 0.9967, as shown in Figure 4 [38].

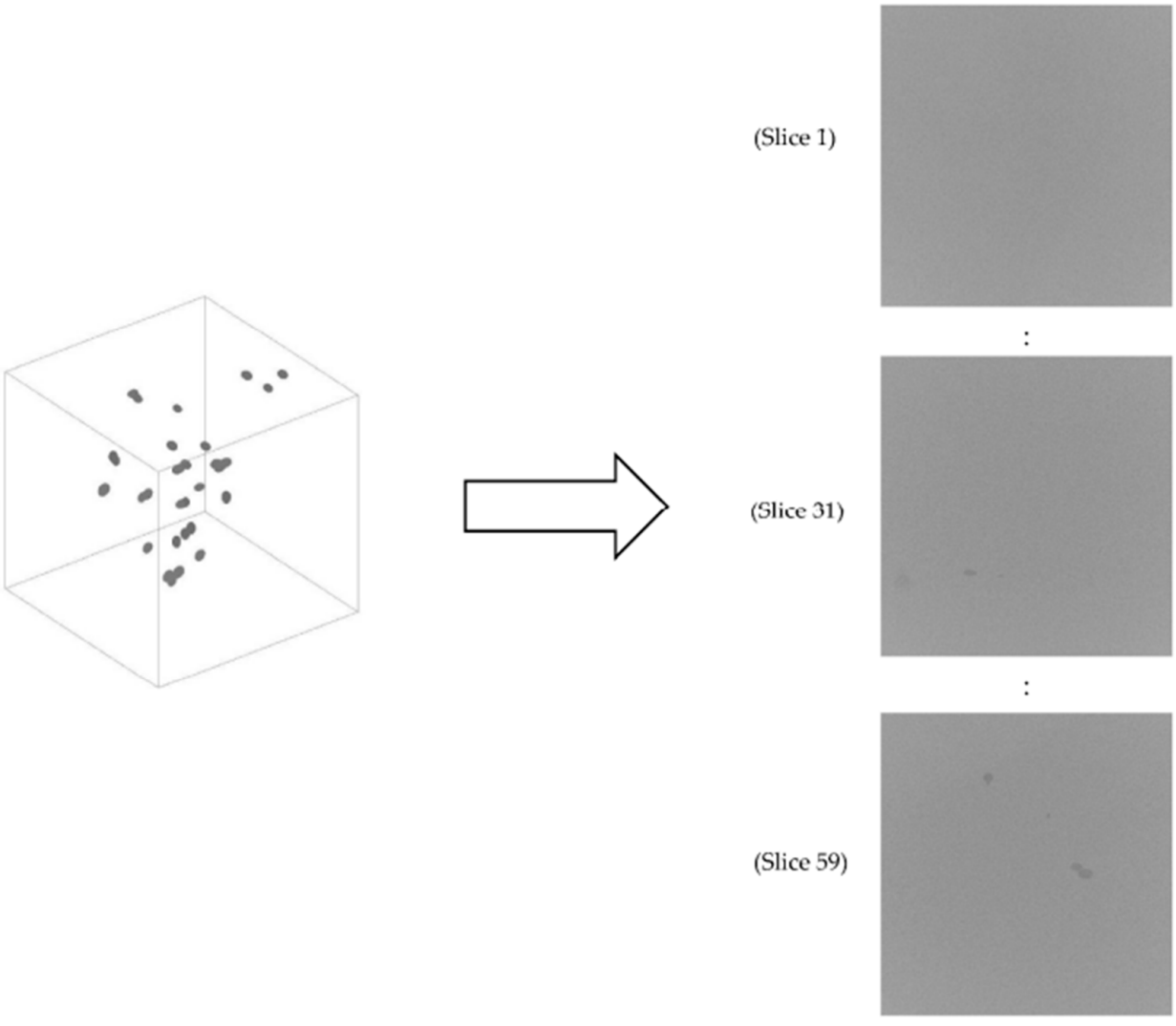

Figure 4. An illustrative example of artificial porosity images [38].

The cube was sliced from bottom to top, so the pattern of the slices is expected to start with no pores, and then, coming to the middle, the pores arise with part of a pore in a slice, and the remaining part of that pore comes in the following slice, as the slice thickness is 0.01 mm in a cube volume of 1 mm^3. The dependencies between the layers are addressed

in the LSTM, as the important information is stored in the cell state for a long period, as described in Section 2.1.

The labels of the slices are the actual percent of porosity calculated by dividing the element numbers that contain the pixel value of the pores (ranging between 110 and 124) by the image size (650 × 630 × 3), as described in [38]. For example, slice 59 has 2193 cells with pixel values between 110 and 124, so the actual percent of porosity of this slice is 0.1785 (2193/(650 × 630 × 3)) [38].

The design of the LSTM architecture consists of 8 layers. It starts with a sequence input layer that inputs the sequential data to the network, followed by a sequence folding layer that converts the image sequences to a batch of images so that the convolution operation described in the previous section can be performed on the layers independently [12]. After folding, the convolution is applied to input data, which is a matrix that contains the pixel value with a range between 0–255, where 0 represents the black regions and 255 represents the white regions. The size of this matrix depends on the image size, which is 650 × 630 [31]. After performing the convolution operations to extract the image features, the sequence unfolding layer is added to restore the sequence structure of the input data, followed by a flatten layer that collapses the input spatial dimensions to the channel dimension [12].

Then, the bidirectional LSTM layer is added, which takes the input from both directions (forward and backward). The output of both the forward and backward at each stage is transferred to an activation layer (neural network). The output from this activation layer considers the relationship between the past and future layers [40], which increases the prediction accuracy of the percent of porosity in each layer. Reference [41] showed the following Figure 5, which illustrates the process for the bidirectional LSTM layer:

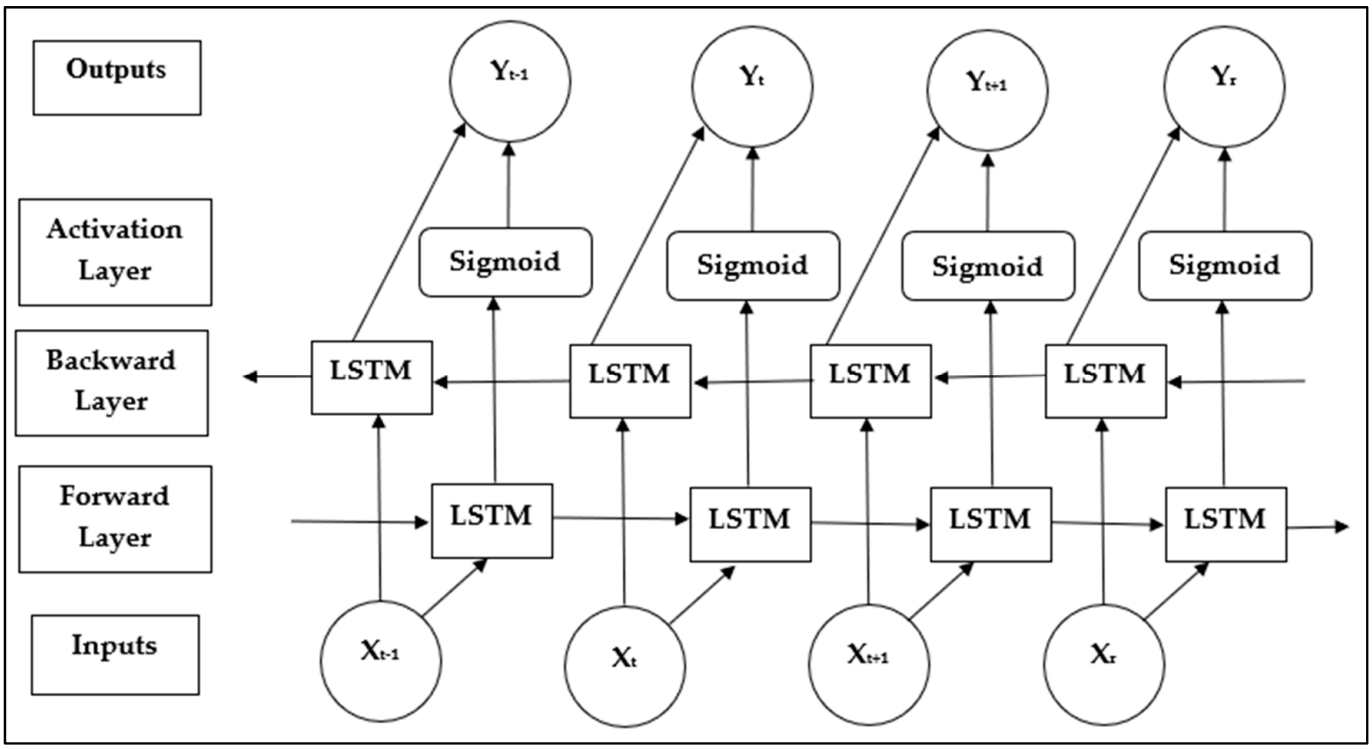

Figure 5. The process for the bidirectional LSTM layer [41].

The number of hidden units is 200. Having more hidden units increases the computation cost without an improvement in the prediction accuracy, and also it increases the probability of overfitting [13]. The bidirectional LSTM layer is followed by a dropout layer, which minimizes the risk of overfitting [13], a fully connected layer is added with one predictor (percent of porosity), and the last layer is the regression layer.

3.3. BA Process

The BA is a swarm-based optimization technique that performs global and local searches to find the optimal solution. It is an iterative process that starts with performing a global search using scout bees. They use the fitness value (validation error) to evaluate different positions. Since the objective function of this optimization problem is to minimize the validation error, it is a complex function that cannot be formulated. After that, the local search selects the best sites and abandons the remaining sites. Then, the intense search selects the best among the selected sites to recruit more bees in the most promising sites based on the neighborhood size and there being fewer bees in the other sites to conduct the local search that is performed simultaneously with the global search. The process is terminated if the optimal solution is reached, the number of iterations is exceeded, or if there is no improvement over sequential specified iterations [42].

The following Table 2 shows the values of BA parameters, which were assigned based on the capability of the computer and using the equations stated in [43].

Table 2. The values of BA parameters.

BA Parameter	Symbol	Equation	Value
Scout bees	n	-	4
Selected bees	m	$(1/2) \times m$	2
Elite bees	e	-	1
Recruited bees for elite sites	nep	$2 \times m$	4
Recruited bees for other sites	nsp	$(1/2) \times n$	2
Neighborhood size	ngh	$0.1 \times (\text{Maximum} - \text{Minimum}) = 0.1 \times (1.1 - 0.9)$	0.02

The challenge is in using the BA process to find the optimal values for nine LSTM parameters to improve the performance of three gates and the cell state, which is specifically achieved by optimizing the learning rate factor so that each part has its own learning rate that is determined based on the global learning rate. Having a more optimum learning rate means more optimum updates for the network weight described in Section 3.2 [11]. The bidirectional LSTM layer was used, as described in Section 3.2, which trains the input as is and on the reverse copy of the input [30], this resulted in four parameters related for the forward side, the other four parameters related to the backward side, and one parameter related to the fully connected layer. Consequently, the nine optimization variables became the following:

- Learning rate factor for input gate (Forward);
- Learning rate factor for forget gate (Forward);
- Learning rate factor for cell candidate (Forward);
- Learning rate factor for output gate (Forward);
- Learning rate factor for input gate (Backward);
- Learning rate factor for forget gate (Backward);
- Learning rate factor for cell candidate (Backward);
- Learning rate factor for output gate (Backward);
- Learning rate factor for fully connected layer.

The optimal adjustment factors obtained by the BA (ranging between 0.9 and 1.1) were multiplied by the global learning rate of 0.0101 [38] to result in a more optimum learning rate for each part stated above so that the network weights described in Section 3.2 were updated with more optimum values to improve the performance of the LSTM.

The pseudo-code for the proposed novel hybrid BA-CNN-LSTM algorithm is presented in the following Figure 6. Reference [38] was used to develop this figure.

Figure 6. The pseudo-code for the proposed novel hybrid BA-CNN-LSTM algorithm.

In addition to the pseudo-code, the following Figure 7 shows a flow chart for the workflow diagram for the proposed novel hybrid BA-CNN-LSTM algorithm.

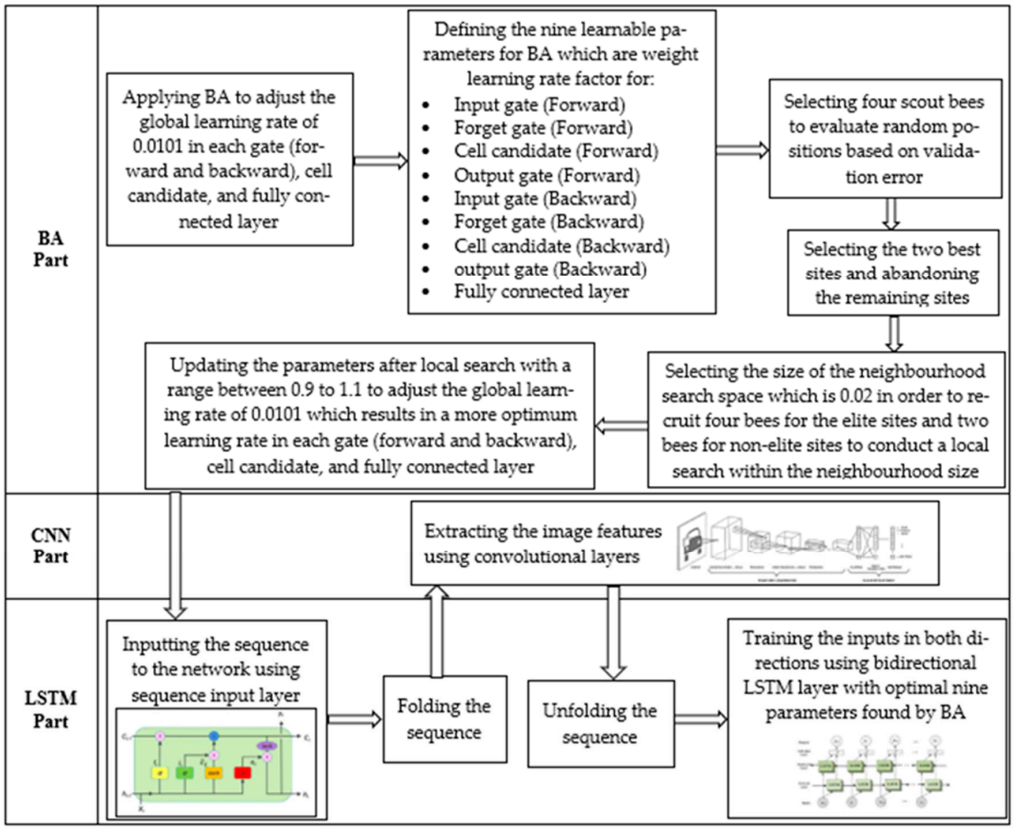

Figure 7. Workflow diagram for the proposed novel hybrid BA-CNN-LSTM algorithm.

4. Results and Discussion

This section presents the results of applying the LSTM network to three sets of data of artificial porosity images to predict the porosity percentage in the SLM part as mentioned previously. Also, it was applied in the signal processing context to classify electrocardiogram (ECG) benchmark image data [44]. In addition, the turbofan engine degradation simulation dataset was used to predict the Remaining Useful Life (RUL) of engines using the LSTM [45].

MATLAB platform was used to design the LSTM network, the CNN, and the BA, as well as to apply the hybrid algorithms to three benchmark datasets. The system configuration consisted of a single GPU with a memory of 256 GB to be able to handle the 3000 artificial porosity images of size (650 × 630 × 3). Given that the DL algorithms require high computations, limited BA evaluations were applied to optimize the LSTM parameters.

4.1. Artificial Porosity Images

The novel hybrid BA-CNN-LSTM algorithm was developed using the MATLAB platform. It was applied to the created artificial porosity images described in the previous section [38] to predict the percent of porosity in sequential layers of the SLM parts. The 30 sequences were divided into 18 sequences for the network training task, 6 sequences for the validation set, and the 6 sequences for the testing set. Since each sequence has 100 layers, 1800 slices were used for training, and 600 slices were used for each of the validation and testing sets. The following Table 3 shows the values of the LSTM parameters for the four evaluations of the BA.

Table 3. The values of LSTM parameters in the four evaluations of BA (artificial porosity images).

LSTM Parameter	1	2	3	4
Learning rate factor for input gate (Forward)	1.0618	0.9737	1.0629	1.0810
Learning rate factor for forget gate (Forward)	0.9472	0.9706	1.0812	1.0177
Learning rate factor for cell candidate (Forward)	0.9151	1.0451	0.9254	0.9291
Learning rate factor for output gate (Forward)	1.0349	1.0273	1.0827	1.0300
Learning rate factor for input gate (Backward)	1.0684	1.0604	1.0265	0.9429
Learning rate factor for forget gate (Backward)	1.0258	1.0236	0.9195	1.0386
Learning rate factor for cell candidate (Backward)	0.9749	0.9621	0.9557	1.0006
Learning rate factor for output gate (Backward)	1.0253	0.9779	1.0094	0.9224
Learning rate factor for fully connected layer	0.9623	0.9477	1.0915	0.9931
Prediction error for the validation set	0.0115	0.0120	0.0134	0.0135

As can be seen in the previous table, the first evaluation yielded the minimum prediction error on the validation set with a value of 0.0115, so the global learning rate of 0.0101 [38] was adjusted in the forward side of the input gate by multiplying it by 1.0618, which resulted in a more optimum learning rate value of 0.0107. Similarly, the learning rate in the forward side of forget gate was improved to 0.0095 by using an adjustment factor of 0.9472. The new learning rate value for the forward cell candidate was 0.0092 after multiplying the global learning rate by 0.9151. The adjustment factor for the forward output gate was 1.0348, which resulted in a learning rate value of 0.0104. The new values for the four backward parameters were 0.0108, 0.0103, 0.0098, and 0.0103 for input gate, forget gate, cell candidate, and output gate, respectively. They were adjusted using factors

of 1.0684, 1.0258, 0.9749, and 1.0253, respectively. Finally, the performance of the fully connected layer was improved as well by specifying a customized learning rate of 0.0097 after multiplying the global learning rate by 0.9623. The following Table 4 summarizes the new learning rate values for the LSTM parameters:

Table 4. The new learning rate values of LSTM parameters (artificial porosity images).

LSTM Parameter	Adjusted Learning Rate Value
Input gate (Forward)	0.0107
Forget gate (Forward)	0.0095
Cell candidate (Forward)	0.0092
Output gate (Forward)	0.0104
Input gate (Backward)	0.0108
Forget gate (Backward)	0.0103
Cell candidate (Backward)	0.0098
Output gate (Backward)	0.0103
Fully connected layer	0.0097

The following Figure 8 shows the training progress for the proposed BA-CNN-LSTM algorithm using the new learning rate values stated above. The blue line represents the training progress, and the black line represents the validation set.

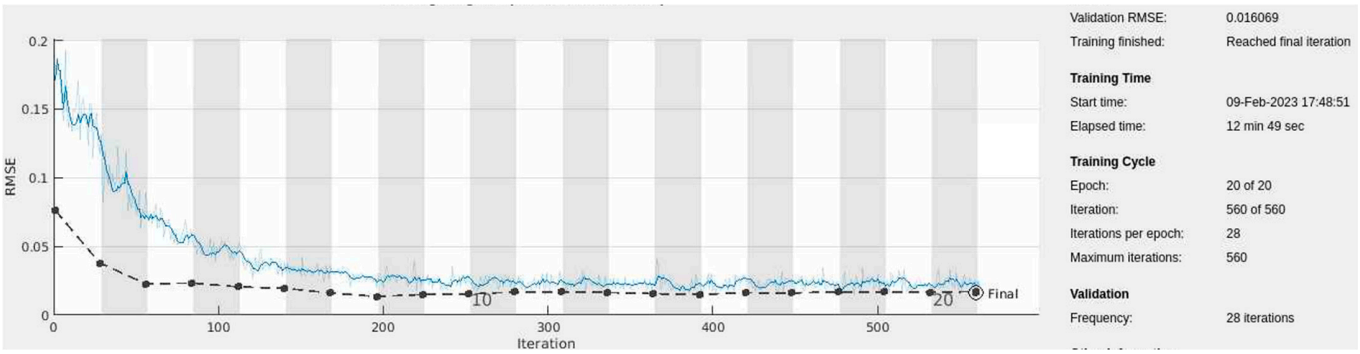

Figure 8. Training progress for the proposed hybrid BA-CNN-LSTM algorithm (artificial porosity images).

As can be seen from the figure, the training started with a root mean square error (RMSE) of 0.2 and decreased significantly after the first 200 iterations, and then the chart experienced a steady state around an RMSE value of 0.02. In the validation set, the chart started with an RMSE value of 0.05, and it was alternating in the first 150 iterations before reaching an RMSE value of 0.0160 at the end of the chart.

The following Table 5 shows the average porosity error (the difference between the actual and predicted percent of porosity) for the training, validation, and testing sets using 10-fold cross-validation. The novel hybrid BA-CNN-LSTM algorithm was compared with an existing algorithm that used Bayesian Optimization (BO) to optimize the same LSTM parameters (BO-CNN-LSTM) [46]. Additionally, it was compared with the CNN-LSTM algorithm without the BA and with the BA-RCNN algorithm that was developed in [38] by using the same CNN structure and parameters described in Section 3.1.

Table 5. The average error for percent of porosity (artificial porosity images).

	BA-RCNN	CNN-LSTM	BO-CNN-LSTM	BA-CNN-LSTM
Average error for percent of porosity in the training data	0.0228	0.0160	0.0166	0.0155
Average error for percent of porosity in the validation data	0.0216	0.0126	0.0128	0.0118
Average error for percent of porosity in the testing data	0.0223	0.0131	0.0131	0.0122

Adding the LSTM network to the CNN reduced the prediction error in all training, validation, and testing sets. The hybrid BO-CNN-LSTM did not perform better than the original CNN-LSTM, so the BO is not recommended to be used in regression problems, as it performs poorly with a high dimensional objective function of more than 20 dimensions [47]. The CNN-LSTM algorithm was further developed by adding the BA to optimize the LSTM parameters, which reduced the prediction error further in the validation and testing sets to reach the minimum error value of 0.0118 that was the result from the validation set of the novel hybrid BA-CNN-LSTM algorithm.

In order to test the significance of the error differences in the testing set, the two-sample *t*-test was conducted to investigate the difference between the BA-RCNN and CNN-LSTM and between the CNN-LSTM and BA-CNN-LSTM algorithms using the 95% confidence level. There is no point to performing the test between the CNN-LSTM and BO-CNN-LSTM, as there was no difference in the testing set. The Minitab software version 17 was used to conduct the tests, as shown in the following Figure 9.

Two-Sample T-Test and CI: BA-RCNN, CNN-LSTM

```
Two-sample T for BA-RCNN vs CNN-LSTM

            N    Mean   StDev  SE Mean
BA-RCNN   600  0.0223  0.0229  0.00094
CNN-LSTM  600  0.0131  0.0134  0.00055


Difference = μ (BA-RCNN) - μ (CNN-LSTM)
Estimate for difference:  0.00918
95% CI for difference:  (0.00705, 0.01131)
T-Test of difference = 0 (vs ≠): T-Value = 8.47  P-Value = 0.000  DF = 963
```

Two-Sample T-Test and CI: CNN-LSTM, BA-CNN-LSTM

```
Two-sample T for CNN-LSTM vs BA-CNN-LSTM

              N    Mean   StDev  SE Mean
CNN-LSTM      600  0.0131  0.0134  0.00055
BA-CNN-LSTM   600  0.0122  0.0116  0.00047


Difference = μ (CNN-LSTM) - μ (BA-CNN-LSTM)
Estimate for difference:  0.000871
95% CI for difference:  (-0.000545, 0.002287)
T-Test of difference = 0.01 (vs ≠): T-Value = -12.65  P-Value = 0.000  DF = 1174
```

Figure 9. The two-sample *t*-test between BA-RCNN, CNN-LSTM, and BA-CNN-LSTM algorithms.

With a *p*-value less than 0.05, it meant that the error difference in the testing set was significant between the BA-RCNN and CNN-LSTM and between the CNN-LSTM and BA-CNN-LSTM algorithms. Thus, adding the LSTM network reduced the prediction error significantly. Furthermore, optimizing the LSTM parameters using the BA had a significant contribution as well in error reduction.

The following Figure 10 shows the validation accuracy for all four algorithms within a 0.02 threshold (the acceptable error) using 10-fold cross-validation.

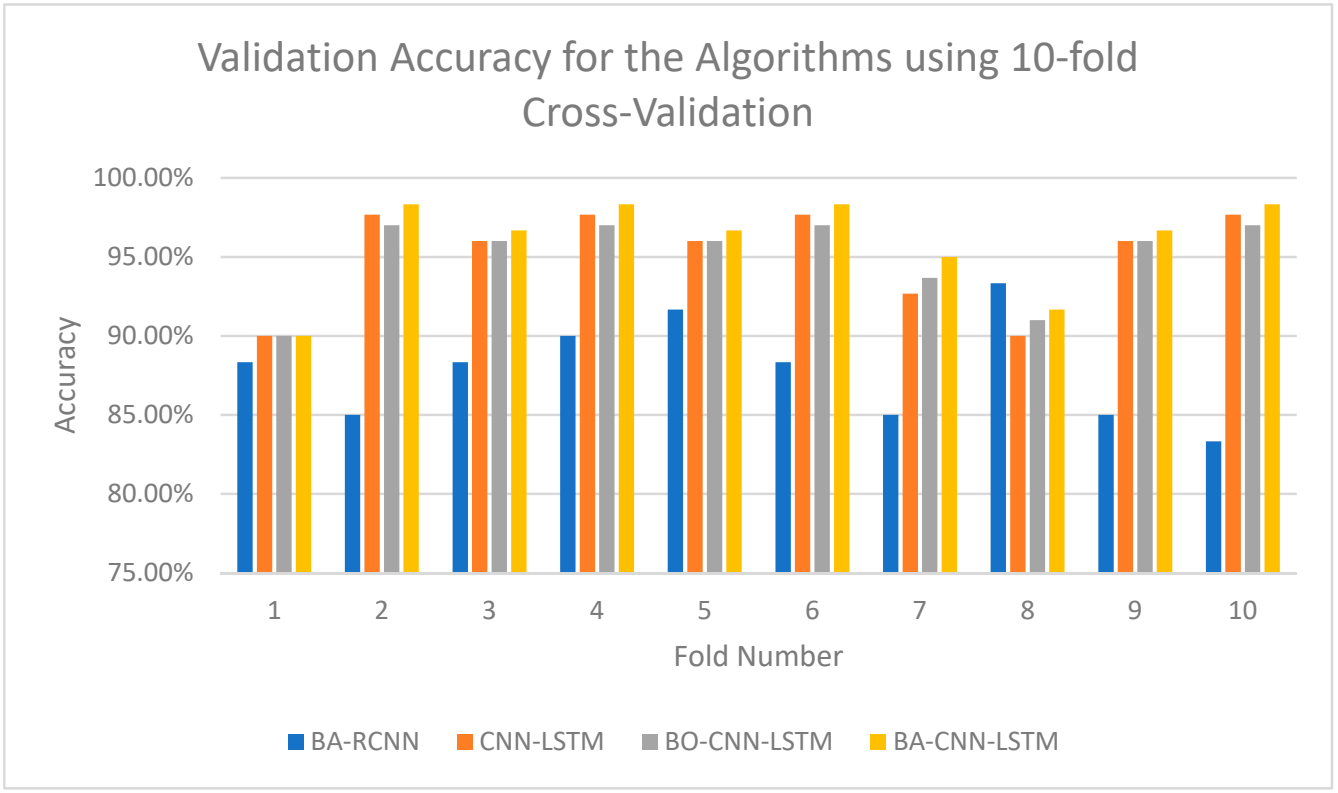

Figure 10. Validation accuracy for the algorithms using 10-fold cross-validation (artificial porosity images).

The following Table 6 presents the training, validation, and testing average prediction accuracies of the 10-fold cross-validations for all algorithms, in addition to the time taken for computations in the best iteration.

Table 6. The prediction accuracy and time for percent of porosity (artificial porosity images).

	BA-RCNN	CNN-LSTM	BO-CNN-LSTM	BA-CNN-LSTM
Training set	85.50%	88.33%	88.33%	88.33%
Validation set	87.33%	95.14%	95.07%	96%
Testing set	85.17%	93.17%	93%	95.17%
Computation time	13 min 5 s	13 min 4 s	13 min 13 s	12 min 49 s

As can be seen from the previous table, adding the LSTM network to the CNN improved the prediction accuracy in all training, validation, and testing sets. The testing accuracy was increased by 8% in the testing set to 93.17%. Optimizing the LSTM parameters using the BO did not improve the testing accuracy, and it was almost the same with a value of 93%. The CNN-LSTM algorithm was further developed by adding the BA to optimize

the LSTM parameters, which increased the prediction accuracy in the validation and testing sets from 95.14% and 93.17% to 96% and 95.17%, respectively, so the improvement in the testing set was 2% from the CNN-LSTM algorithm and 10% from the BA-RCNN algorithm. The improvement was similar to the improvement discussed in Section 2.3, as the hybrid ABC-CNN-LSTM algorithm used to detect fake reviews of a product yielded an accuracy of 97% compared to 95% for the CNN-LSTM algorithm [29]. The computational time was almost similar for all algorithms.

As a result, the performance of the LSTM network for predicting sequential data was improved after using the BA, as the hybrid BA-CNN-LSTM provided a more accurate prediction by 10% for the percent of porosity in sequential layers of artificial porosity images that mimicked CT scan images of parts manufactured by the SLM process.

4.2. Electrocardiogram (ECG) Dataset

The novel hybrid BA-CNN-LSTM algorithm developed using the MATLAB platform can be designed to deal with classification problems as well. It was applied to the Electrocardiogram (ECG) benchmark images described in [44] to classify human ECG time series signals into three classes of cardiac arrhythmia (ARR), congestive heart failure (CHF), and normal sinus rhythms (NSR). The following Figure 11 shows an example of the three classes of ECG time series signals.

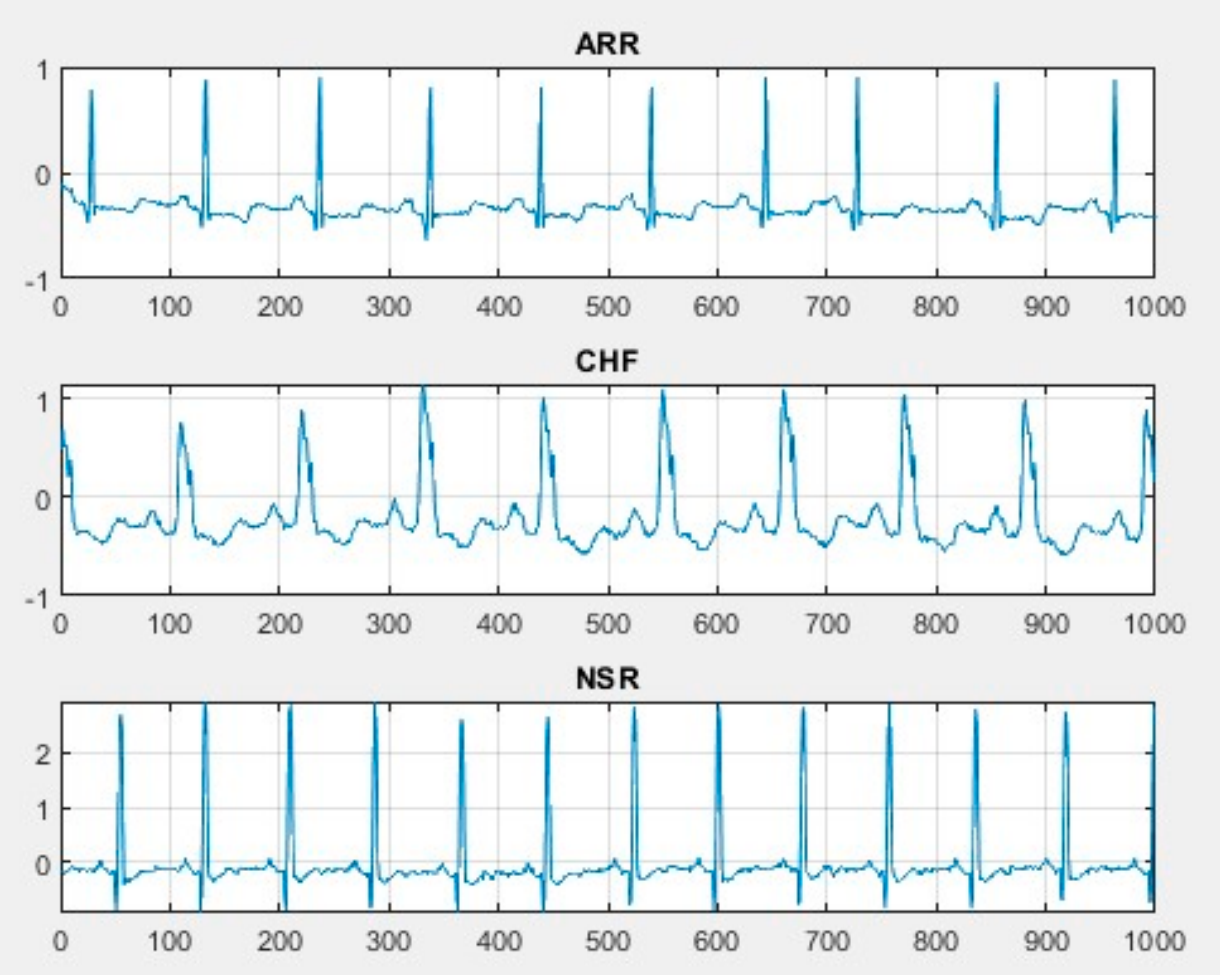

Figure 11. An illustrative example of the three classes of ECG time series signals.

There were 162 recordings with 96 observations from the ARR, 30 recordings from the CHF, and 36 from the NSR. Because the dataset was not large enough, the "SqueezeNet"

pre-trained CNN network was used to extract the features in the images. The data were divided into 81 recordings for the training, 41 recordings for the validation set, and 40 observations for the testing set. The following Table 7 shows the values of the LSTM parameters for the evaluations of the BA.

Table 7. The values of LSTM parameters for the evaluations of BA (ECG dataset).

LSTM Parameter	1	2
Learning rate factor for input gate (Forward)	0.9483	1.0422
Learning rate factor for forget gate (Forward)	0.9808	1.0948
Learning rate factor for cell candidate (Forward)	0.9193	0.9703
Learning rate factor for output gate (Forward)	0.9264	1.0198
Learning rate factor for input gate (Backward)	1.0884	1.0804
Learning rate factor for forget gate (Backward)	1.0912	1.0722
Learning rate factor for cell candidate (Backward)	1.0150	0.9603
Learning rate factor for output gate (Backward)	0.9120	1.0578
Learning rate factor for fully connected layer	0.9470	0.9413
Classification error for the validation set	0.1463	0.1220

As can be seen in the previous table, the second evaluation yielded the minimum classification error for the validation set with a value of 0.1220, so the global learning rate of 0.00029 was adjusted in the forward side of the input gate by multiplying it by 1.0422, which resulted in a more optimum learning rate value of 0.0003. Similarly, the learning rate in the forward side of the forget gate was improved to 0.00031 using an adjustment factor of 1.0984. The new learning rate value for the forward cell candidate was 0.00028 after multiplying the global learning rate by 0.9703. The adjustment factor for the forward output gate was 1.0198, which resulted in a learning rate value of 0.00029. The new values for four backward parameters were 0.00031, 0.00031, 0.00027, and 0.0003 for the input gate, forget gate, cell candidate, and output gate, respectively. They were adjusted using factors of 1.0804, 1.0722, 0.9603, and 1.0578, respectively. Finally, the performance of the fully connected layer was improved as well by specifying a customized learning rate of 0.00027 after multiplying the global learning rate by 0.9413. The following Table 8 summarizes the new learning rate values for the LSTM parameters.

Table 8. The new learning rate values of LSTM parameters (ECG dataset).

LSTM Parameter	Adjusted Learning Rate Value
Input gate (Forward)	0.000302238
Forget gate (Forward)	0.000317492
Cell candidate (Forward)	0.000281387
Output gate (Forward)	0.000295742
Input gate (Backward)	0.000313316
Forget gate (Backward)	0.000310938
Cell candidate (Backward)	0.000278487
Output gate (Backward)	0.000306762
Fully connected layer	0.000272977

The following Figure 12 shows the training progress for the proposed BA-CNN-LSTM algorithm using the new learning rate values stated above. The blue line represents the training progress, and the black line represents the validation set.

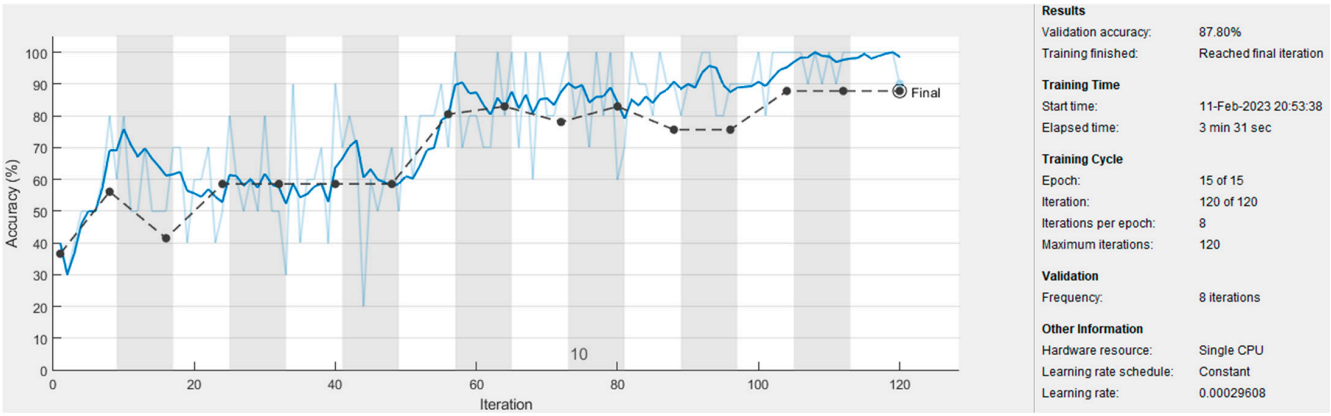

Figure 12. Training progress for the proposed hybrid BA-CNN-LSTM algorithm (ECG dataset).

As can be seen from the figure, the training started with low classification accuracy and increased steadily until it reached a percentage value of 100%. In the validation set, the chart had almost the same pattern and reached a validation accuracy of 87.80% at the end of the chart. The following three Figures 13–15 show the confusion matrix for the BA-CNN-LSTM algorithm for all three sets:

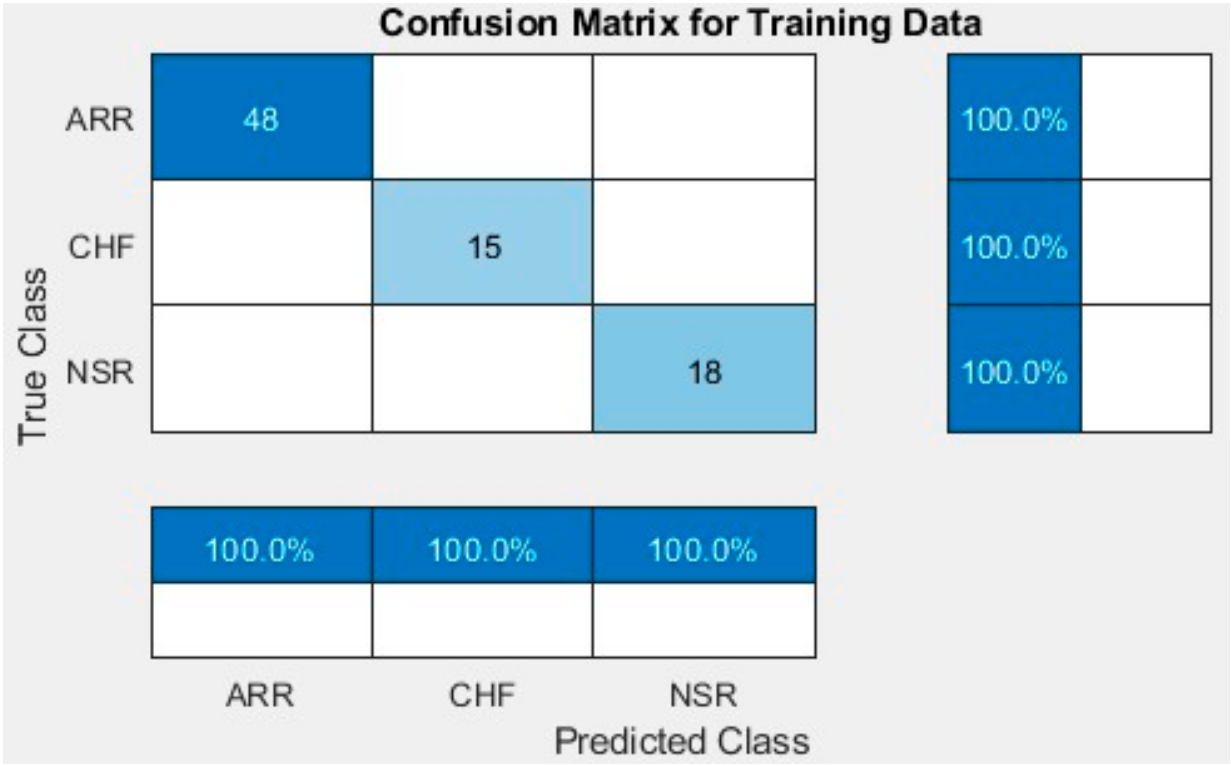

Figure 13. Confusion matrix for the training set of the proposed hybrid BA-CNN-LSTM algorithm.

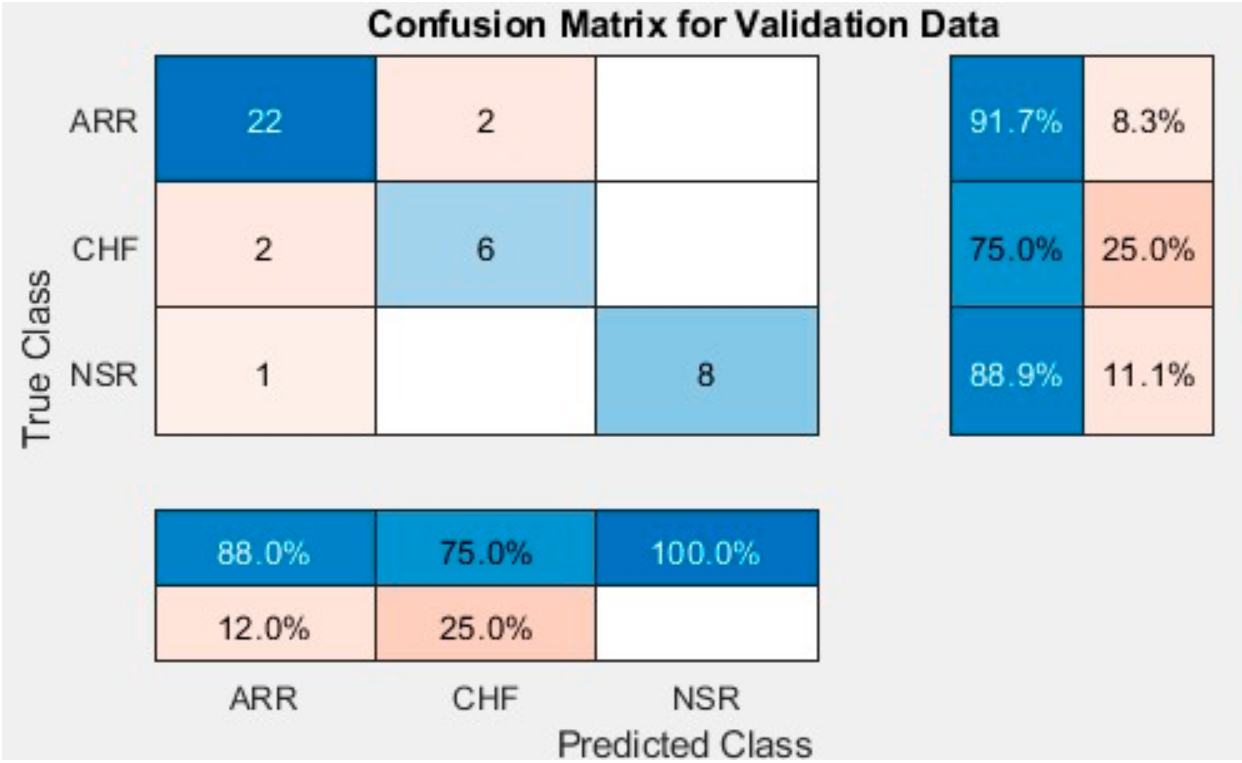

Figure 14. Confusion matrix for the validation set of the proposed hybrid BA-CNN-LSTM algorithm.

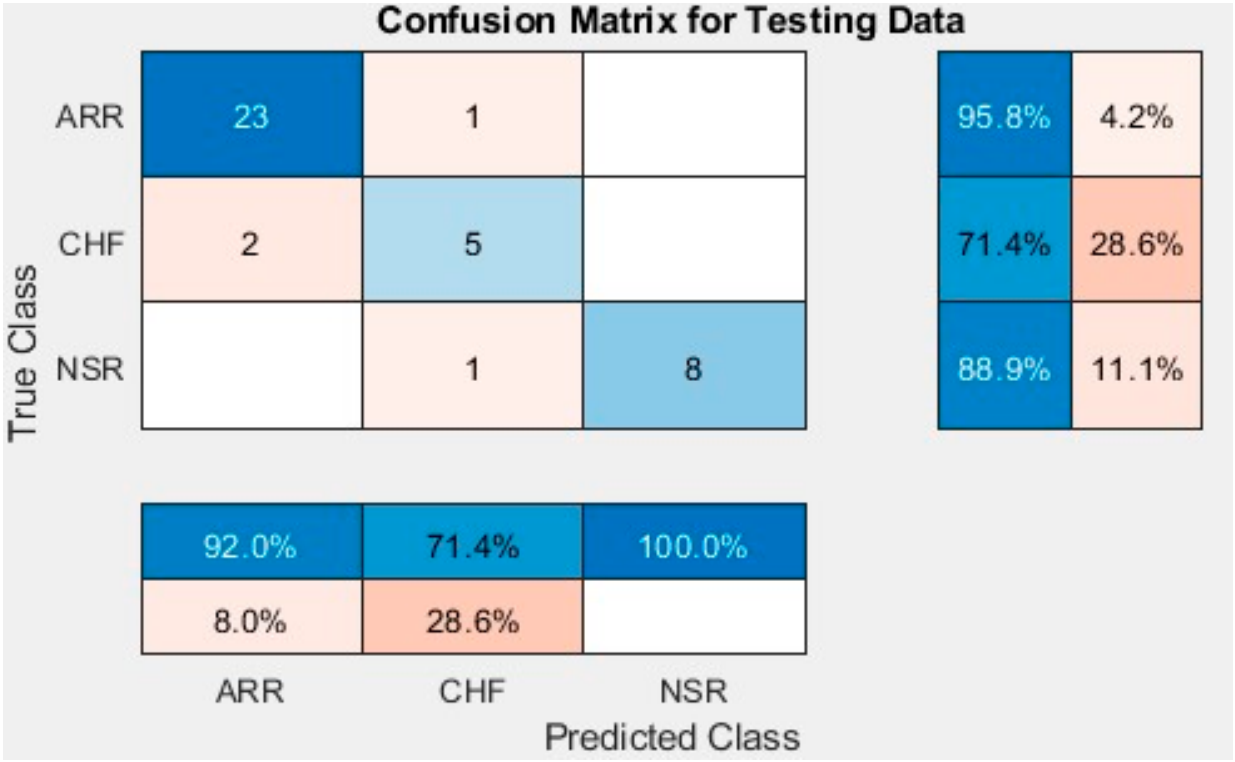

Figure 15. Confusion matrix for the testing set of the proposed hybrid BA-CNN-LSTM algorithm.

Since it is a classification problem, the *t*-test was not applicable, as the error is either zero or one. The following Figure 16 shows the validation accuracy for all four algorithms using 10-fold cross-validation.

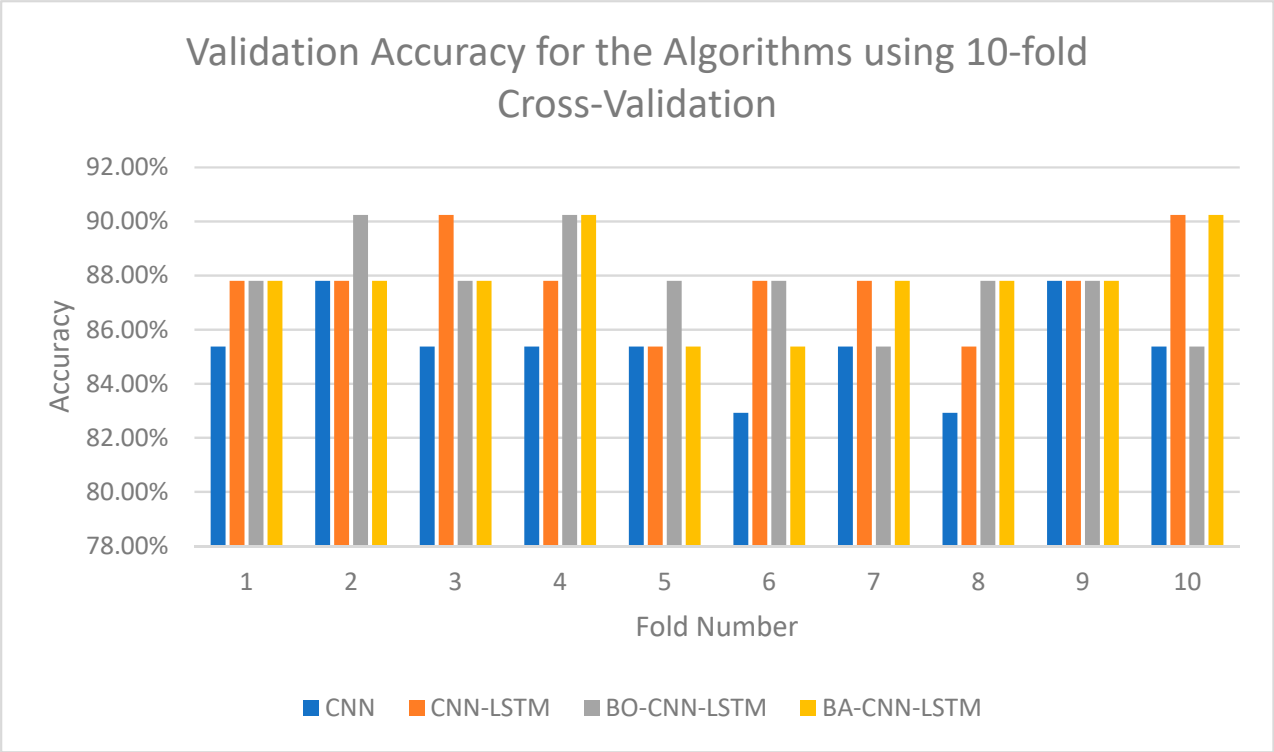

Figure 16. Validation accuracy for the algorithms using 10-fold cross-validation (ECG dataset).

The following Table 9 presents the training, validation, and testing average classification accuracy of the 10-fold cross-validation for CNN, CNN-LSTM, BO-CNN-LSTM, and BA-CNN-LSTM algorithms, in addition to the time taken for computations in the best iteration.

Table 9. The classification accuracy and time (ECG dataset).

	CNN	CNN-LSTM	BO-CNN-LSTM	BA-CNN-LSTM
Training set	100%	100%	100%	100%
Validation set	85.37%	87.80%	87.80%	87.80%
Testing set	92.50%	92.50%	95%	95%
Computation time	3 min 47 s	3 min 52 s	4 min 2 s	3 min 31 s

As can be seen from the previous table, adding the LSTM network to the CNN improved the validation accuracy in the validation set by 2.5%. Optimizing the LSTM parameters using the BO or BA improved the testing accuracy from 92.50% to 95%, meaning that both algorithms perform well in classification problems. The computational time was almost similar in all algorithms.

4.3. Turbofan Engine Degradation Simulation Dataset

The dataset consists of time series data of 100 engines that start normally at the beginning, and then some faults appear during the series. The numerical data contain 26 columns starting with a unit number, time in cycles, 3 operational settings, and 21 sensor measurements [48]. The datasets contain 100 sequences, as each engine represents a sequence that varies in length. There are 100 observations for each of the training, validation, and testing sets. Because the dataset is numerical, a CNN was not needed to extract the features, so the hybrid BA-LSTM algorithm was applied using the MATLAB platform to predict the remaining operational cycles before engine failure. Since the forward LSTM was used in developing the LSTM architecture [45], only four parameters related to this

layer were optimized using the BA, which were the learning rate factors for the input gate, forget gate, cell candidate, and fully connected layer. The following Table 10 shows the values of the LSTM parameters for the four evaluations of the BA.

Table 10. The values of LSTM parameters in the four evaluations of BA (engine dataset).

LSTM Parameter	1	2	3	4
Learning rate factor for input gate	1.0712	0.9557	1.0413	0.9048
Learning rate factor for forget gate	1.0167	0.9147	1.0244	0.9898
Learning rate factor for cell candidate	1.0747	1.0939	1.0716	1.0878
Learning rate factor for output gate	1.0526	0.9989	0.9378	1.0041
Learning rate factor for fully connected layer	0.9915	1.0902	0.9613	1.0519
Prediction error for the validation set	4.4272	5.2740	6.7158	11.6459

As can be seen in the previous table, the first evaluation yielded the minimum prediction error for the validation set with a value of 4.4272, so the global learning rate of 0.01 [45] was adjusted for the input gate by multiplying it by 1.0712, which resulted in a more optimum learning rate value of 0.0107. Similarly, the learning rate for the forget gate was improved to 0.0101 by using an adjustment factor of 1.0167. The new learning rate value for the cell candidate was 0.0107 after multiplying the global learning rate by 1.0747. The adjustment factor for the output gate was 1.0526, which resulted in a learning rate value of 0.0105. Finally, the performance of the fully connected layer was improved as well by specifying a customized learning rate of 0.0099 after multiplying the global learning rate by 0.9915. The following Table 11 summarizes the new learning rate values for the LSTM parameters.

Table 11. The new learning rate values of LSTM parameters (engine dataset).

LSTM Parameter	Adjusted Learning Rate Value
Input gate (Forward)	0.0107
Forget gate (Forward)	0.0101
Cell candidate (Forward)	0.0107
Output gate (Forward)	0.0105
Fully connected layer	0.0099

The following Figure 17 shows the training progress for the BA-LSTM algorithm using the new learning rate values stated above. The blue line represents the training progress, and the black line represents the validation set.

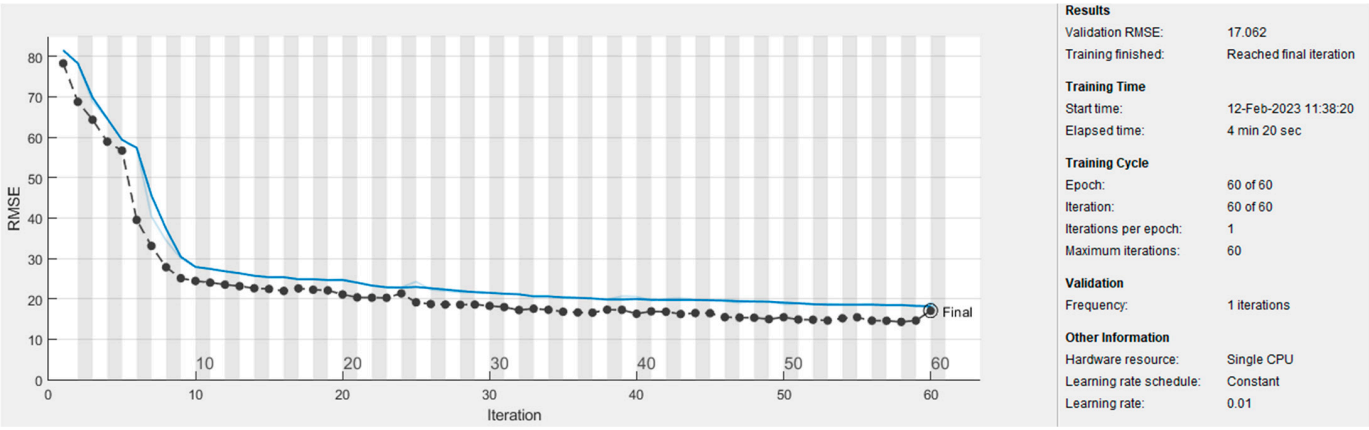

Figure 17. Training progress for the hybrid BA-LSTM algorithm (engine dataset).

As can be seen from the figure, the training started with a root mean square error (RMSE) of almost 80 that decreased significantly during the first 20 iterations, and then the chart experienced a steady state around an RMSE value of 20. In the validation set, the chart followed the same pattern and reached an RMSE value of 17.062 at the end of the chart.

In order to test the significance of the error differences in the testing set, the two-sample *t*-test was conducted to investigate the difference between the LSTM and BA-LSTM algorithms using the 95% confidence level. There was no point to performing the test between the LSTM and BO-LSTM, as there was no difference between the algorithms. The Minitab software version 17 was used to conduct the tests, as shown in the following Figure 18.

Two-Sample T-Test and CI: LSTM, BA-LSTM

```
Two-sample T for LSTM vs BA-LSTM

           N  Mean  StDev  SE Mean
LSTM     100  15.4   14.3      1.4
BA-LSTM  100  13.5   13.4      1.3


Difference = μ (LSTM) - μ (BA-LSTM)
Estimate for difference:  1.91
95% CI for difference:  (-1.95, 5.77)
T-Test of difference = 20 (vs ≠): T-Value = -9.25  P-Value = 0.000  DF = 197
```

Figure 18. The two-sample *t*-test between LSTM and BA-LSTM algorithms.

With a *p*-value less than 0.05, it meant that the error difference in the testing set was significant between the LSTM and BA-LSTM algorithms. Therefore, optimizing the LSTM parameters using the BA had a significant contribution to error reduction.

The following Figure 19 shows the validation accuracy for all algorithms within a 20 cycle threshold (the acceptable error) using 10-fold cross-validation.

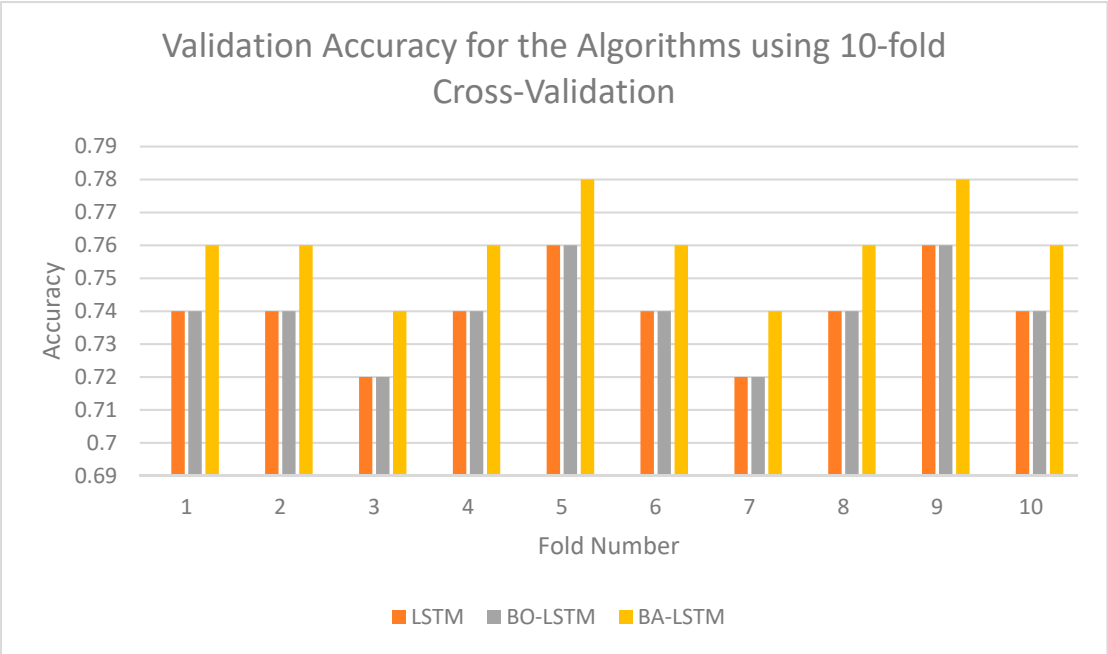

Figure 19. Validation accuracy for the algorithms using 10-fold cross-validation (engine dataset).

The following Table 12 presents the training, validation, and testing average prediction accuracy of the 10-fold cross-validation for the LSTM, BO-LSTM and BA-LSTM algorithms within a 20 cycle threshold (the acceptable difference between the actual and predicted RUL), in addition to the time taken for computations in the best iteration.

Table 12. The prediction accuracy and time for RUL (engine dataset).

	LSTM	BO-LSTM	BA-LSTM
Training set	72%	72%	73%
Validation set	74%	74%	76%
Testing set	74%	74%	77%
Computation time	4 min 16 s	4 min 18 s	4 min 20 s

As can be seen from the previous table, using the BO to optimize the LSTM parameters did not improve the performance of LSTM, which confirms the conclusion that it is not recommended to use the BO in regression problems, as it performs poorly with a high dimensional objective function of more than 20 dimensions [47]. Adding the BA to optimize the LSTM parameters improved the prediction accuracy in all training, validation, and testing sets. The testing accuracy was increased by 3% in the testing set to 77% for the BA-LSTM algorithm. The computational time was almost similar for all algorithms.

5. Conclusions

Improving the performance of Deep Learning (DL) algorithms is an ongoing challenge. However, DL is applied to different types of Deep Neural Networks, and Long Short-Term Memory (LSTM) is one of them that deals with time series or sequential data. This paper addressed this issue by optimizing LSTM parameters using one of the most popular nature-inspired algorithms known as the Bees Algorithm (BA) which mimics the foraging behavior of honey bees. Artificial porosity images were used for testing the algorithms; since the input data were images, a Convolutional Neural Network (CNN) was added in order to extract the features in the images and feed the LSTM to predict the percent of porosity in sequential layers of artificial porosity images that mimicked real CT scan images of products manufactured by the Selective Laser Melting process.

The MATLAB platform was used to develop and apply the Convolutional Neural Network Long Short-Term Memory (CNN-LSTM), which yielded a porosity prediction accuracy of 93.17%. Using the BO to optimize LSTM parameters did not improve the performance of the LSTM, as the BO performs poorly with a high dimensional objective function of more than 20 dimensions, which is the case in regression problems. However, Adding the BA to optimize the same LSTM parameters improved their performance in predicting the porosity, which yielded an accuracy of 95.17% using the hybrid Bees Algorithm Convolutional Neural Network Long Short-Term Memory (BA-CNN-LSTM). Hence, this work has contributed to improving the performance of the LSTM network for predicting sequential data using the BA. As the input data were images, a CNN was added to extract the image features to yield a hybrid algorithm (BA-CNN-LSTM) that provided a more accurate prediction and an improvement of 10% for the percent of porosity in sequential layers of artificial porosity images that mimicked CT scan images of parts manufactured by the SLM process.

Furthermore, the hybrid BA-CNN-LSTM algorithm can be designed to deal with classification problems as well. Applying it to Electrocardiogram (ECG) benchmark images improved the test set classification accuracy from 92.50% for the CNN-LSTM algorithm to 95% for both the BO-CNN-LSTM and BA-CNN-LSTM algorithms. In addition, the turbofan engine degradation simulation dataset was used to predict the Remaining Useful Life (RUL) of the engines using the LSTM network. A CNN was not needed in this case, as there was no feature extraction for the images. However, adding the BA to optimize the LSTM parameters improved the prediction accuracy of the testing set for the LSTM and BO-LSTM, which increased from 74% to 77% for the hybrid BA-LSTM algorithm.

Further improvement to the novel hybrid BA-CNN-LSTM algorithm will be done in the future by optimizing the adjustment factor of regularization to reduce the overfitting, which in turn is expected to improve the overall performance of the LSTM network.

Author Contributions: Conceptualization, N.M.H.A., M.P. and S.B.; methodology, N.M.H.A., M.P. and S.B.; software, N.M.H.A.; validation, M.P. and S.B.; formal analysis, N.M.H.A., M.P. and S.B.; investigation, N.M.H.A. and M.P.; resources, N.M.H.A., M.P. and S.B.; data curation, N.M.H.A. and S.B.; writing—original draft preparation, N.M.H.A.; writing—review and editing, M.P. and S.B.; visualization, N.M.H.A., M.P. and S.B.; supervision, M.P. and S.B.; project administration, N.M.H.A., M.P. and S.B.; funding acquisition, N.M.H.A., M.P. and S.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Saudi Arabian Government.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to thank the H2020 project partners of the Additive Manufacturing using Metal Pilot Line (MANUELA) project (grant agreement No. 820774) for their collaboration.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Li, B.H.; Hou, B.C.; Yu, W.T.; Lu, X.B.; Yang, C.W. Applications of artificial intelligence in intelligent manufacturing: A review. *Front. Inf. Technol. Electron. Eng.* **2017**, *18*, 86–96. [CrossRef]
2. De Filippis, L.A.C.; Serio, L.M.; Facchini, F.; Mummolo, G. ANN Modelling to Optimize Manufacturing Process. In *Advanced Applications for Artificial Neural Networks*; El-Shahat, A., Ed.; IntechOpen: London, UK, 2017; pp. 201–226.
3. Wuest, T.; Weimer, D.; Irgens, C.; Thoben, K.-D. Machine learning in manufacturing: Advantages, challenges, and applications. *Prod. Manuf. Res.* **2016**, *4*, 23–45. [CrossRef]
4. Singh, A.K.; Ganapathysubramanian, B.; Sarkar, S.; Singh, A. Deep Learning for Plant Stress Phenotyping: Trends and Future Perspectives. *Trends Plant Sci.* **2018**, *23*, 883–898. [CrossRef] [PubMed]
5. Wang, J.; Ma, Y.; Zhang, L.; Gao, R.X.; Wu, D. Deep learning for smart manufacturing: Methods and applications. *J. Manuf. Syst.* **2018**, *48*, 144–156. [CrossRef]
6. Ekins, S. The Next Era: Deep Learning in Pharmaceutical Research. *Pharm. Res.* **2016**, *33*, 2594–2603. [CrossRef] [PubMed]
7. Long Short-Term Memory (LSTM) Deep Learning. Available online: https://the-learning-machine.com/article/dl/long-short-term-memory (accessed on 4 January 2023).
8. Thakur, D. LSTM and Its Equations. 2018. Available online: https://medium.com/@divyanshu132/lstm-and-its-equations-5ee9246d04af (accessed on 4 January 2023).
9. Introduction to Long Short-Term Memory (LSTM). Available online: https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/ (accessed on 4 January 2023).
10. Alamri, N.M.H.; Packianather, M.; Bigot, S. Deep Learning: Parameter Optimization Using Proposed Novel Hybrid Bees Bayesian Convolutional Neural Network. *Appl. Artif. Intell.* **2022**, *36*, 2031815. [CrossRef]
11. Brownlee, J. Understand the Impact of Learning Rate on Neural Network Performance. 2020. Available online: https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/ (accessed on 29 November 2021).
12. What Is Vanishing Gradient Problem in RNN? Available online: https://www.engati.com/glossary/vanishing-gradient-problem#:~{}:text=The%20vanishing%20gradient%20problem%20is%20essentially%20a%20situation,layers%20near%20the%20input%20end%20of%20the%20model (accessed on 4 January 2023).
13. How to Diagnose Overfitting and Underfitting of LSTM Models. Available online: https://machinelearningmastery.com/diagnose-overfitting-underfitting-lstm-models/ (accessed on 4 January 2023).
14. Joshi, S.; Verma, D.K.; Saxena, G.; Paraye, A. Issues in training a convolutional neural network model for image classification. In *International Conference on Advances in Computing and Data Sciences*; Springer: Singapore, 2019.
15. Liang, J.; Liu, R. Stacked denoising autoencoder and dropout together to prevent overfitting in deep neural network. In Proceedings of the 2015 8th International Congress on Image and Signal Processing (CISP), Shenyang, China, 14–16 October 2015; pp. 697–701.
16. Cogswell, M.; Ahmed, F.; Girshick, R.; Zitnick, L.; Batra, D. Reducing overfitting in deep networks by decorrelating representations. *arXiv* **2015**, arXiv:1511.06068.

17. Wu, J. *Introduction to Convolutional Neural Networks*; National Key Lab for Novel Software Technology, Nanjing University: Nanjing, China, 2017.
18. Varikuti, M. LSTM Network. 2021. Available online: https://medium.com/mlearning-ai/lstm-networks-75d44ac8280f (accessed on 4 January 2023).
19. Mattioli, F.E.; Caetano, D.J.; Cardoso, A.; Naves, E.L.; Lamounier, E.A. An Experiment on the Use of Genetic Algorithms for Topology Selection in Deep Learning. *J. Electr. Comput. Eng.* **2019**. [CrossRef]
20. Zhang, H.; Kiranyaz, S.; Gabbouj, M. Finding better topologies for deep convolutional neural networks by evolution. *arXiv* **2018**, arXiv:1809.0324.
21. Chiroma, H.; Gital, A.Y.U.; Rana, N.; Shafi'i, M.A.; Muhammad, A.N.; Umar, A.Y.; Abubakar, A.I. Nature inspired meta-heuristic algorithms for deep learning: Recent progress and novel perspective. In *Science and Information Conference*; Springer: Cham, Switzerland, 2019.
22. Kim, D.G.; Choi, J.Y. Optimization of Design Parameters in LSTM Model for Predictive Maintenance. *Appl. Sci.* **2021**, *11*, 6450. [CrossRef]
23. Gorgolis, N.; Hatzilygeroudis, I.; Istenes, Z.; Gyenne, L.G. Hyperparameter optimization of LSTM network models through genetic algorithm. In Proceedings of the 2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA), Patras, Greece,, 15–17 July 2019; pp. 1–4.
24. Pranolo, A.; Mao, Y.; Wibawa, A.P.; Utama, A.B.P.; Dwiyanto, F.A. Optimized Three Deep Learning Models Based-PSO Hyperparameters for Beijing PM2.5 Prediction. *Knowl. Eng. Data Sci.* **2022**, *5*, 53–66. [CrossRef]
25. Qureshi, A.-U.; Larijani, H.; Mtetwa, N.; Javed, A.; Ahmad, J. RNN-ABC: A New Swarm Optimization Based Technique for Anomaly Detection. *Computers* **2019**, *8*, 59. [CrossRef]
26. Zeybek, S.; Pham, D.; Koç, E.; Seçer, A. An Improved Bees Algorithm for Training Deep Recurrent Networks for Sentiment Classification. *Symmetry* **2021**, *13*, 1347. [CrossRef]
27. Kumar, R.; Kumar, P.; Kumar, Y. Integrating big data driven sentiments polarity and ABC-optimized LSTM for time series forecasting. *Multimedia Tools Appl.* **2022**, *81*, 34595–34614. [CrossRef]
28. Ding, L.; Fang, W.; Luo, H.; Love, P.E.; Zhong, B.; Ouyang, X. A deep hybrid learning model to detect unsafe behavior: Integrating convolution neural networks and long short-term memory. *Autom. Constr.* **2018**, *86*, 118–124. [CrossRef]
29. Jacob, M.S.; Rajendran, P.S. Fuzzy artificial bee colony-based CNN-LSTM and semantic feature for fake product review classification. *Concurr. Comput. Pract. Exp.* **2022**, *34*, e6539. [CrossRef]
30. Brownlee, J. How to Develop a Bidirectional LSTM for Sequence Classification in Python with Keras. 2021. Available online: https://machinelearningmastery.com/develop-bidirectional-lstm-sequence-classification-python-keras/ (accessed on 7 January 2023).
31. Hui, J. Convolutional Neural Networks (CNN) Tutorial. 2017. Available online: https://jhui.github.io/2017/03/16/CNNConvolutional-neural-network (accessed on 15 November 2021).
32. MathWorks-1. Convolutional Neural Network. Available online: https://uk.mathworks.com/solutions/deep-learning/convolutional-neural-network.html (accessed on 7 October 2019).
33. Yamashita, R.; Nishio, M.; Do, R.K.G.; Togashi, K. Convolutional neural networks: An overview and application in radiology. *Insights Imaging* **2018**, *9*, 611–629. [CrossRef]
34. McDermott, J. Convolutional Neural Networks—Image Classification w. Keras. 2021. Available online: https://www.learndatasci.com/tutorials/convolutional-neural-networks-image-classification (accessed on 15 November 2021).
35. Ouf, H. Maxpooling vs. Minpooling vs. Average Pooling. 2017. Available online: https://hany-ouf.blogspot.com/2020/08/maxpooling-vs-minpooling-vs-average.html (accessed on 15 November 2021).
36. MathWorks-2. Training Options SGDM. Available online: https://uk.mathworks.com/help/deeplearning/ref/nnet.cnn.trainingoptionssgdm.html (accessed on 15 November 2021).
37. MathWorks-3. Deep Learning Using Bayesian Optimization. Available online: https://www.mathworks.com/help/deeplearning/ug/deep-learning-using-bayesian-optimization.html (accessed on 4 April 2020).
38. Alamri, N.M.H.; Packianather, M.; Bigot, S. Predicting the Porosity in Selective Laser Melting Parts Using Hybrid Regression Convolutional Neural Network. *Appl. Sci.* **2022**, *12*, 12571. [CrossRef]
39. MathWorks-4. Long Short-Term Memory Networks. Available online: https://www.mathworks.com/help/deeplearning/ug/long-short-term-memory-networks.html (accessed on 8 January 2023).
40. Mungalpara, J. What Does It Mean by Bidirectional LSTM? 2021. Available online: https://medium.com/analytics-vidhya/what-does-it-mean-by-bidirectional-lstm-63d6838e34d9 (accessed on 7 January 2023).
41. Newman, L. Classifying Toxicity in Online Comment Forums: End-to-End Project. 2020. Available online: https://towardsdatascience.com/classifying-toxicity-in-online-comment-forums-end-to-end-project-57720af39d0b (accessed on 26 January 2023).
42. Al-Musawi, A. The Development of New Artificial Intelligence Based Hybrid Techniques Combining Bees Algorithm, Data Mining and Genetic Algorithm for Detection, Classification and Prediction of Faults in Induction Motors. Ph.D. Thesis, Cardiff University, Cardiff, UK, 2019.
43. MathWorks-5. Bees Algorithm (BeA) in MATLAB. Available online: https://uk.mathworks.com/matlabcentral/fileexchange/52967-bees-algorithm-bea-in-matlab (accessed on 9 April 2020).

44. MathWorks-6. Classify Time Series Using Wavelet Analysis and Deep Learning. Available online: https://www.mathworks.com/help/wavelet/ug/classify-time-series-using-wavelet-analysis-and-deep-learning.html (accessed on 16 January 2023).
45. MathWorks-7. Sequence-to-Sequence Regression Using Deep Learning. Available online: https://www.mathworks.com/help/deeplearning/ug/sequence-to-sequence-regression-using-deep-learning.html (accessed on 10 January 2023).
46. MathWorks-8. Choose Training Configurations for LSTM Using Bayesian Optimization. Available online: https://www.mathworks.com/help/deeplearning/ug/exp-mgr-sequence-regression-example.html (accessed on 28 January 2023).
47. Why Does Bayesian Optimization Perform Poorly in More than 20 Dimensions? Available online: https://stats.stackexchange.com/questions/564528/why-does-bayesianoptimization-perform-poorly-in-more-than-20-dimensions#:~{}:text=Disadvantages%20of%20Bayesian%20Optimization%3A%201%20Requires%20the%20true,20%20di (accessed on 17 November 2022).
48. Saxena, A.; Goebel, K.; Simon, D.; Eklund, N. Damage Propagation Modeling for Aircraft Engine Run-to-Failure Simulation. In Proceedings of the Ist International Conference on Prognostics and Health Management (PHM08), Denver, CO, USA, 6–9 October 2008.