



# Performance comparison of feedforward neural network training algorithms in modeling for synthesis of polycaprolactone via biopolymerization

Y. J. Wong<sup>1</sup> · Senthil Kumar Arumugasamy<sup>2</sup> · J. Jewaratnam<sup>1</sup>

Received: 13 December 2017 / Accepted: 2 July 2018 / Published online: 9 July 2018  
© Springer-Verlag GmbH Germany, part of Springer Nature 2018

## Abstract

This paper reports the biopolymerization of  $\epsilon$ -caprolactone, using lipase Novozyme 435 catalyst at varied impeller speeds and reactor temperatures. A multilayer feedforward neural network (FFNN) model with 11 different training algorithms is developed for the multivariable nonlinear biopolymerization of polycaprolactone (PCL). In previous works, biopolymerization carried out in scaled-up bioreactors is modeled through FFNN. No review discussed the role of different training algorithms in artificial neural network on the estimation of biopolymerization performance. This paper compares mean absolute error, mean square error, and mean absolute percentage error (MAPE) in the PCL biopolymerization process for 11 different training algorithms that belong to six classes, namely (1) additive momentum, (2) self-adaptive learning rate, (3) resilient backpropagation, (4) conjugate gradient backpropagation, (5) quasi-Newton, and (6) Bayesian regulation propagation. This paper aims to identify the most effective training method for biopolymerization. Results show that the quasi-Newton-based and Levenberg–Marquardt algorithms have the best performance with MAPE values of 4.512, 5.31, and 3.21% for the number of average molecular weight, weight average molecular weight, and polydispersity index, respectively.

**Keywords** Polycaprolactone (PCL) · Biopolymerization · Ring opening polymerization · Artificial neural network modeling · Bioreactor

## Introduction

Polycaprolactone (PCL), which is largely produced through chemical polymerization, is emerging as a preferred biodegradable plastic in the recent years. PCL has remarkable resistance to water, oil, solvent, and chlorine; hydrophobic; and relatively high biodegradability and biocompatibility, making it a highly potential biomaterial and environment-friendly thermoplastic (Bassi et al. 2011). Although the synthetic production of PCL has been established, it has a major

drawback of having metallic catalyst residues, which can cause adverse effect, especially in biomaterial application (Labet and Thielemans 2009). As a result, green methods of producing PCL via biopolymerization are investigated widely. Biopolymerization using enzyme as catalysts is known as enzymatic polymerization. The enzymatic polymerization of PCL, as an alternative to the conventional chemical polymerization, addresses the issue of metallic residues on the final product. Biopolymers produced using biocatalyst show no signs of toxic residues (Varma et al. 2005).

Several enzymatic polymerization techniques, such as condensation, anionic, cationic, and ring opening polymerizations (ROP), are available. Among these techniques, ROP is the most preferred (Varma et al. 2005). In the past decade, PCL production via ROP has been extensively investigated, and different enzymes, such as lipase *Candida antarctica* (CA), lipase *Candida cylindracea* (CC), lipase *Pseudomonas fluorescens* (PF) ( ), lipase PP (PP porcine pancreas), lipase PC (*Pseudomonas cepacia*), and lipase RJ (*Rhizopus japonicus*) (Uyama et al. 1996; Varma et al. 2005; Torres et al. 2010) and non-lipase enzymes, such as Humicola Insolens

✉ Senthil Kumar Arumugasamy  
SenthilKumar.Arumugasamy@nottingham.edu.my

✉ J. Jewaratnam  
jegalaxmi24@um.edu.my

<sup>1</sup> Department of Chemical Engineering, University of Malaya, Jalan Universiti, 50603 Kuala Lumpur, Wilayah Persekutuan, Malaysia

<sup>2</sup> Department of Chemical and Environmental Engineering, University of Nottingham Malaysia Campus, Jalan Broga, 43500 Semenyih, Selangor Darul Ehsan, Malaysia

Cutinase (HIC) (Hunsen et al. 2007), have been used. The scope and significance of enzymatic polymerization are wide, and its other aspects should be explored. The molecular weight of a polymer determines its quality and application. This paper addresses molecular weight prediction accuracy. The accuracy of molecular weight prediction in a polymer can be affected by the nonlinear relationship in the biopolymerization process parameters, such as reaction temperature, reactor impeller speed, polymerization time, and catalyst type (Kobayashi and Makino 2009).

The artificial neural network (ANN) is an emerging technique used for accurately predicting the molecular weight of PCL. ANN mimics the biological neural network in the human brain. Although ANNs are not smart, they can analyze and recognize the patterns of the input and target data inserted and create rules to solve complex problems (Lee-Cosio et al. 2012). Most research in this field has been conducted for the production of PCLs with desired molecular weights through the optimization of the main polymerization parameters. Arumugasamy and Ahmad (2010) modeled a feedforward neural network (FFNN) for the large-scale production of PCL to investigate the effect of temperature and impeller speed on PCL biopolymerization. Arumugasamy et al. (2012) conducted a modeling study on PCL production in a flask and reactor and used lipase Novozyme 435 as catalyst. Khatti et al. (2017) carried out a study on the analysis of ANN and response surface methodology to determine effective parameters, such as polymer concentration, voltage, and nozzle-to-collector distance, on the diameter of electrospun PCL (Khatti et al. 2017).

A remarkable drawback in using ANN is the long duration required for training a network and ensuring the convergence of a local minimum value. Different training algorithms that improve performance in different aspects have been proposed for this drawback. Xinxing et al. (2013) compared 12 different ANN training algorithms based on the mean absolute percentage error (MAPE) and training time for electricity load forecasting. Forouzanfar et al. (2010) compared the different training algorithms for oscillometric blood pressure estimation. Ghaffari et al. (2006) performed four different training algorithms in the modeling of bimodal drug delivery on predictive ability. The outcome of these studies determines the best training algorithm, which provides optimal training duration and accurate prediction that are suitable for their processes. The same approach is attempted for biopolymerization in this work.

The objective of this work is to conduct experimental studies in a bioreactor setup in order to optimize the parameters, namely time, temperature, and impeller speed. We used the experimental data to develop an empirical model for the biopolymerization of  $\epsilon$ -caprolactone ( $\epsilon$ -CL) to PCL. FFNN was used. A total of 11 different training algorithms that belong to six classes, namely additive momentum,

self-adaptive learning rate, resilient backpropagation, conjugate gradient, quasi-Newton, and Bayesian regulation propagation were evaluated. The dataset collected from 84 samples from the biopolymerization experiments was used for the evaluation. Number average molecular weight ( $M_n$ ), weight average molecular weight ( $M_w$ ), and polydispersity index (PDI) estimation from biopolymerization using different training algorithms are compared in term of estimation errors, which are mean absolute error (MAE), mean square error (MSE), and mean absolute percentage error (MAPE) with the experimental results.

## Materials and methods

### Preparation of PCL

Infors-HT Labfors bioreactor with an effective volume of 2 L was used for the scale-up production of PCL. Parameters, such as reactor temperature, reactor impeller speed, and polymerization time, were selected. The monomer used in this work was  $\epsilon$ -CL, which was purchased from Merck Private Limited. Toluene was used as solvent. Toluene had undergone drying over calcium hydride and distillation under nitrogen atmosphere prior to its use as solvent. *Candida antarctica lipase B* (CALB) used was purchased from Science Technics Private Limited, Penang, Malaysia. The ratio of solvent to monomer to catalyst was fixed at 2:1:10 (v/v/wt) (Ajay and Gross 2000).

### Biopolymerization process

The monomer, enzyme, and solvent were fed into the bioreactor. The desired bioreactor temperature was set, and a computer was connected to the bioreactor. When the desired temperature was attained, the stirrer was switched on at the preset speed. During polymerization, sampling was conducted every hour for 7 h. Polymerization was performed at 60, 70, and 80 °C. The temperatures were selected on the basis of the conclusion from the conical flask results obtained in a previous study (Arumugasamy and Ahmad 2010). The rotational speed of the impeller was varied (250, 500, 750, or 1000 rpm) (Chaibakhsh et al. 2012).

### PCL analysis

Gel permeation chromatography (GPC) analysis was performed with Waters 2414 Refractive Index Detector, Waters 1525 Binary HPLC Pump, and three Waters GPC columns, two Styragel<sup>®</sup> HR-4 columns, and one HR-5E column for the observation of molecular weights (particle size: 5  $\mu$ m,  $\varnothing$  4.6 mm  $\times$  300 mm). The analysis conditions were as follows: tetrahydrofuran as eluent, flow rate of 1.0 mL/min, column

temperature of 40 °C, and analysis time of 45 min/cycle. The calibration curves for GPC analysis were obtained by using PCL standards.

We performed GPC analysis on three different PCL standards PCL with known molecular weights to plot the calibration curve. Table 1 shows the molecular weight of PCL standard provided by the supplier and the values of molecular weight from the GPC analysis. After calibration, the molecular weight of the samples obtained from the bio-reactor experiments was analyzed (Figs. 1, 2).

### ANNs

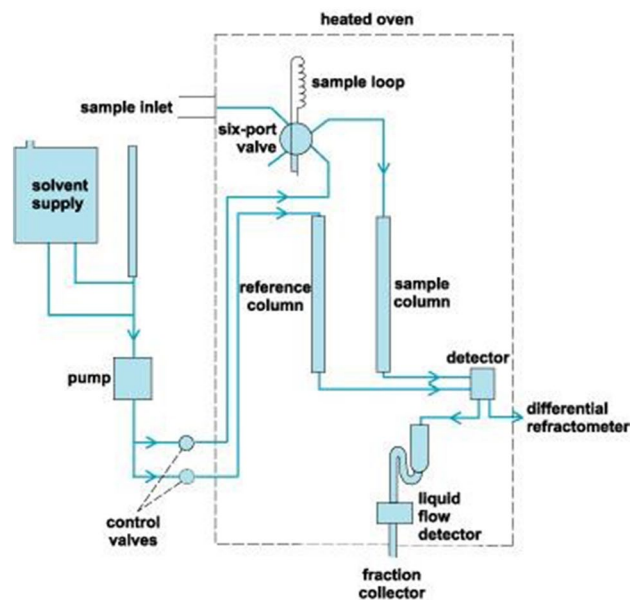
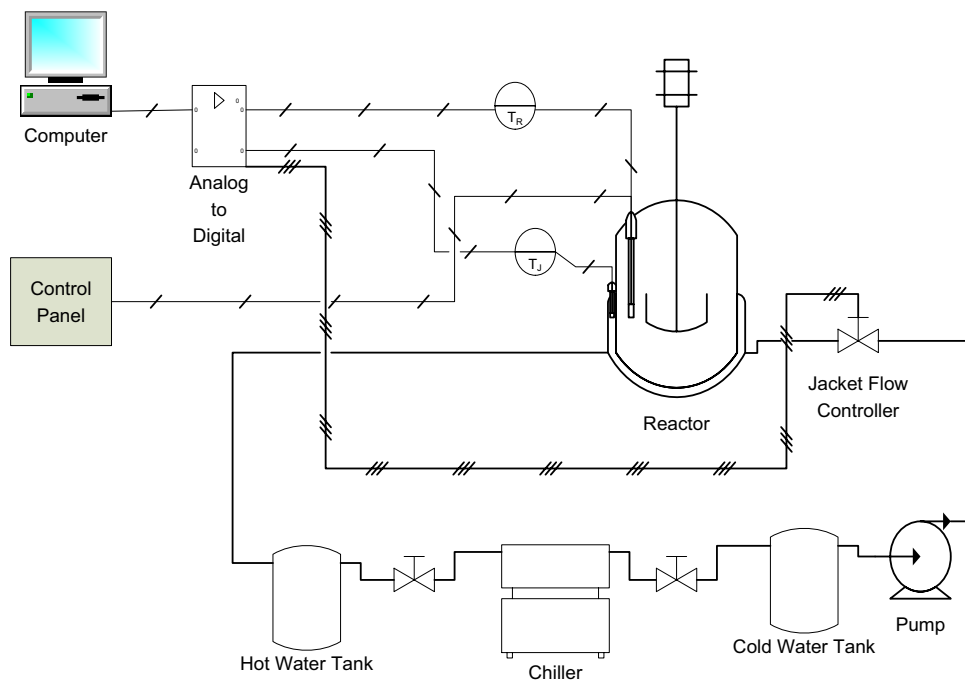
ANN can learn a vast range of application and provide solution by training samples from a particular application. ANN can predict and generalize and has been used to solve complex problems in control, optimization, and classification. It is a data-driven nonlinear model that is formed through the use of artificial neurons and by mapping biological-inspired computational model. Similar to biological neurons, a neural network is composed of three common layers, namely input, hidden, and output. The neurons are connected via

**Table 1** Commercial standard PCL molecular weight with GPC

	$M_n$ from supplier	$M_n$ from GPC
*PCL A	~ 14,000	21,120
*PCL B	~ 65,000	63,686
*PCL C	~ 70,000–90,000	71,841

Standard PCL is purchased from Sigma-Aldrich

**Fig. 1** Schematic diagram of batch bio-reactor



**Fig. 2** Schematic diagram of gel permeation chromatography (GPC) system

coefficients (weights), which are composed of neural structures. The connection among the neurons has a remarkable impact to the ANN. Generally, connection has two main types: feedback (recurrent) and feedforward connections. Feedback is a connection where the output of either the previous or the same layer returns to the input layer. In feedforward, the output does not return to the input neurons. FFNN is a popular and successful neural network architecture that

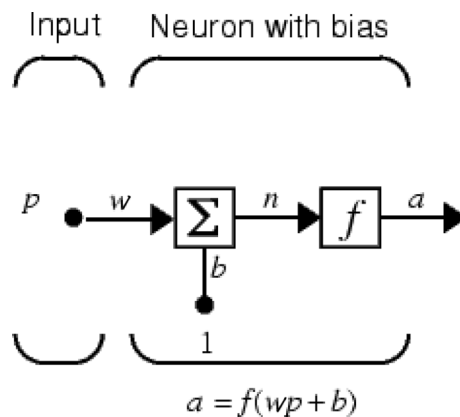
suits extensive range of applications and is proven to be a universal approximator (Hornik et al. 1989).

Three inputs: polymerization time, reactor temperature, and impeller stirring speed and three outputs:  $M_n$ ,  $M_w$ , and PDI were considered in this work. An FFNN with two-to-five hidden layers (depending on the training algorithm) is used.

Figure 3 presents a feedforward multilayer perception network trained with the Levenberg–Marquardt (LM) training algorithm. The network has an architecture structure of 3–4–3, which corresponds to the number of nodes in the input layer, hidden layer 1, hidden layer 2, and output layer, respectively. This indicates that there are three neurons in the input layer referring to time, temperature, and impeller speed, followed by four neurons in the hidden layer and three neurons in the output layer which are the  $M_n$ ,  $M_w$ , and PDI. Figure 4 shows the schematic of fundamental building block for neural networks. First, the scalar input  $p$  represents time, impeller speed, or temperature. It is multiplied to a scalar connection weight  $w$ , which can be positive or negative, for the formation of a weighted input  $wp$ . The weighted input is combined with a scalar bias  $b$ , which has a constant input of 1, to form a net input. Finally, the net input is passed through a transfer function  $f$  to produce scalar output  $a$  for a selected neuron.

**Data normalization**

Data normalization is generally performed for reducing or eliminating the influence of the variations of attributes, such as large numeric ranges that dominate small numeric ranges and scaling of input variables performed prior to forecasting (Sulaiman and Wahab 2018). Min–max normalization



**Fig. 4** Schematic representation of fundamental building block for neural networks

is adopted in this research. The data are normalized in the range of  $[-1, 1]$ , as shown in Eq. (1):

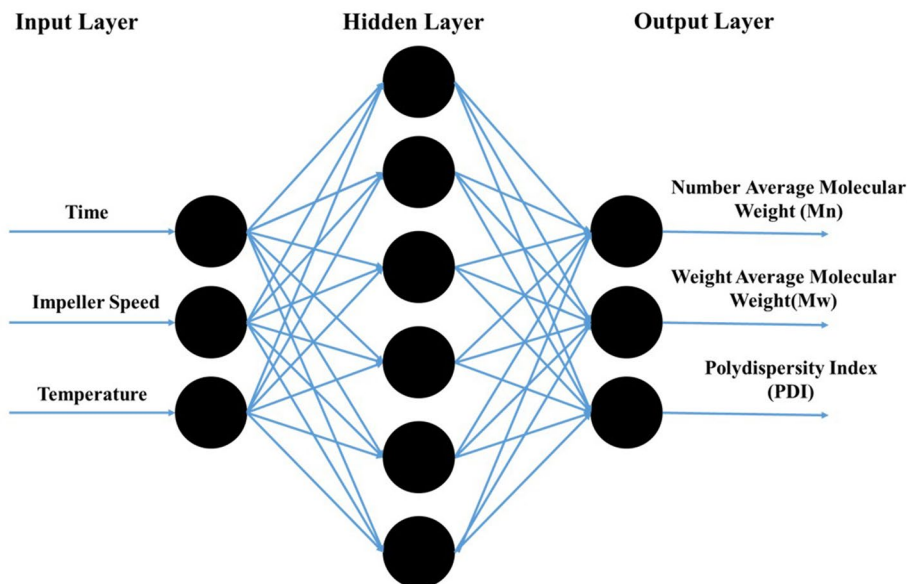
$$y = (y_{\max} - y_{\min}) \frac{x - x_{\min}}{x_{\max} - x_{\min}} + y_{\min}, \tag{1}$$

where  $x$  = original data,  $x_{\max}$  = minimum value in the original data,  $x_{\min}$  = maximum value in the original data,  $y$  = normalized data,  $y_{\max}$  = maximum value in the normalized data (default is 1),  $y_{\min}$  = minimum value in the normalized data (default is -1).

**Hidden layer selection**

The hidden layer is an intermediate layer between the input and output layers. Neurons in the hidden layer are activated

**Fig. 3** Schematic representation of a multilayer perceptron feedforward network consisting of three inputs, one hidden layer with six neurons and three outputs



by a function. Table 2 summarizes the capabilities of neural network architectures with various hidden layers (Heaton 2008). In general, one hidden layer is selected for most of the process. Two hidden layers are generally used for modeling data with discontinuities, such as the sawtooth wave pattern. The use of two hidden layers rarely improves the model and may introduce great risk of convergence in a local minima. Therefore, a process that requires two hidden layers is rarely encountered. The use of more than two hidden layers has no theoretical reason. In this work, an FFNN with one hidden layer is used. One hidden layer is sufficient to capture the intricacy of ring opening polymerization process, and the risk of converging to local minima becomes almost nil.

### Hidden neuron selection

Determining the number of neurons in a hidden layer is an important characteristic of an FFNN or perceptron network. Neuron collection determines the overall neural network architect and greatly influences the final output. If inadequate number of neurons are used, then the network is unable to model complex data, resulting in underfitting. If excessive unnecessary neurons are present in the network, then the training time may become excessively long and cause overfitting. When overfitting occurs, the network starts to model random noise in the data. The outcome may cause a model to fit the training data extremely well but generalizes poorly to new data. Usually, rule-of-thumb methods are used to determine a suitable number of neurons to be used in the hidden layers, such as the following (Karsoliya 2012):

1. The number of hidden neurons should be between the number of neurons in the input and output layers.
2. The number of hidden neurons should be 2/3 the number of neurons in the input layer. If the number is insufficient, then add the number of neurons in the output layer.
3. The number of hidden neurons should be less than twice the number of neurons in the input layer.

This work implemented a cross-validation technique where the value of hidden nodes for training and testing varied based on trial and error. The number of hidden nodes with

the lowest mean sum squared error and high correlation coefficient “r” is selected.

### Training algorithms

As mentioned previously, ANN is programmed rather than trained because of its capability of learning. ANN can self-adjust the weighted connection of the links found among neurons in the network to produce the desired output through algorithm training. Training is the core process that enables the network to learn from previous data. After sufficient training, the network can process and respond to the new set of inputs. Optimal weights are then determined. This process is normally performed by identifying the accuracy between the desired target and the network output and then minimizing them according to the weights. In this research, three different performance functions were used for the evaluation of target and output accuracy. These functions, namely (1) MAPE, (2) MAE, and (3) root MSE (RMSE), were used to evaluate the measure of accuracy.

$$\text{MAPE} = \frac{1}{n} \sum_{t=1}^n \left| \frac{A_t - F_t}{A_t} \right| \times 100, \quad (2)$$

$$\text{MAE} = \frac{1}{n} \sum_{t=1}^n |F_t - A_t|, \quad (3)$$

$$\text{RMSE} = \frac{1}{n} \sum_{t=1}^n (F_t - A_t)^{0.5}, \quad (4)$$

where  $F_t$  = forecasted value,  $A_t$  = actual value,  $n$  = number of samples.

Backpropagation is a technique used by calculating the gradient and minimizing the error through batch or incremental styles. In this research, the main consideration is that the training algorithms for backpropagation can be categorized into six different classes, namely (1) additive momentum (gradient descent with momentum); (2) self-adaptive learning rate (LR) (variable LR gradient descent and gradient descent with momentum and adaptive LR);

**Table 2** Determining the number of hidden layers

Number of hidden layers	Results
None	Only able to represent linear separable functions or decisions
1	Can approximate any function that contains a continuous mapping from one finite space to another
2	Can represent an arbitrary decision boundary to arbitrary accuracy with rational activation functions and can approximate any smooth mapping to any accuracy
>2	Additional layers can learn complex representations (sort of automatic feature engineering) for layer layers

(3) resilient backpropagation; (4) conjugate gradient backpropagation (scaled conjugate gradient (SCG), conjugate gradient with Powell–Beale restarts, Fletcher–Powell conjugate gradient, and Polak–Ribière conjugate gradient); (5) quasi-Newton (Levenberg–Marquardt (LM), BFGS quasi-Newton, and one-step secant); and (6) Bayesian regularization propagation (Bayesian regularization). The ANN presented was conducted by using MATLAB™ R2014b, and the training algorithms used were as follows:

(a) Additive momentum (training function: traingdm)

In this algorithm, the variation tendency of the error curve that tolerates small changes in the neural network and the error of the gradient are considered during the backpropagation of the network. This algorithm can prevent the network from falling into local minimum point in the training process. An additive value that is proportional to the value of the previous weights is added together with the thresholds of change to each change of the weight and threshold. Threshold is referring to  $\frac{\partial E}{\partial X}$ . Weight is initiated by the system. A new change of the weights and threshold is generated based on the backpropagation algorithm. The adjust function  $\Delta X$  is as follows:

$$\Delta X(t + 1) = mc \times \Delta X(t) + Ir \times mc \times \frac{\partial E}{\partial X} \tag{5}$$

where  $t$  = training times,  $mc$  = momentum factor (normally 0.95),  $Ir$  = LR (a constant number that ranges from  $0 < Ir \leq 1$ ),  $E$  = error function.

The downsides of this training algorithm are that it is time consuming and its parameters can only be determined through experiments.

(b) Self-adaptive LR (training function: traingda, traingdx)

Slow speed for convergence during training is mainly caused by the improper selection of LR. An adaptive LR attempts to keep the learning step size as large as possible and ensures that the learning is stable. The function expression is the same as the additive momentum, but the LR,  $Ir$ , is not a constant variable:

$$Ir(t + 1) = \begin{cases} 1.05Ir(t) \\ 0.7Ir(t) \\ Ir(t) \end{cases}, \tag{6}$$

$$\begin{aligned} MSE(t + 1) &< MSE(t) \\ MSE(t + 1) &> 1.04MSE(t) \end{aligned}$$

(c) Resilient backpropagation (training function: trainrp)

A sigmoid transfer function is usually used in the hidden layers of a multilayer network. Therefore, the slope of the transfer function must approach zero as the input increases. Sigmoid function may have a small magnitude of gradient. Therefore, when the weights and biases are far from its optimal values, only small changes in the weights and biases occur. Thus, this process becomes time consuming. Resilient backpropagation training algorithm is usually used for eliminating the harmful effects of partial derivate magnitudes, and only the plus or minus symbol of the derivative,  $\Delta_{ij}^{(t)}$ , is considered for the direction of updating the correct values. This approach is a clear and simple learning rule with rapid speed and reaches the best convergence without selecting parameters.

$$\Delta X_{ij}^{(t)} = \begin{cases} -\Delta_{ij}^{(t)}, & \text{if } \frac{\partial E}{\partial X} > 0 \\ +\Delta_{ij}^{(t)}, & \text{if } \frac{\partial E}{\partial X} < 0, \\ 0, & \text{else} \end{cases}, \tag{7}$$

where  $\frac{\partial E}{\partial X}$  = Sum of gradient information for all the patterns,  $t$  = at the time ( $t$ ).

(d) Conjugate gradient backpropagation (training function: traingcb, traingcp, traingcg)

Conjugate gradient backpropagation is an algorithm that combines conjugate gradient and line search strategies to converge faster and minimize the performance function. Most conjugate gradient backpropagations have processes for updating weight and threshold. The cost of calculation is low because second-order function method eliminates the calculation and storage of second-order derivatives. It starts by searching  $P_0$  in the steepest descent direction  $-g_0$  (negative of the gradient) on the first iteration.

$$P_0 = -g_0. \tag{8}$$

Next, the weight and threshold value ( $X$ ) are determined by using the line search method

$$X_{t+1} = X_t + \alpha_t P_t, \tag{9}$$

where  $P$  = search direction,  $\alpha$  = parameter used to decrease the gradient.

(e) Quasi-Newton (training function: trainbfg, trainlm, trainoss)

Newton’s method provides optimization. Newton’s method often converges faster than conjugate gradient method. However, the computation of Hessian matrix (second derivative) is complex and expensive for FFNN. An algorithm that does not compute the second derivative, known

as quasi-Newton method, is proposed to overcome this issue. The basic step of Newton's method is

$$X_{t+1} = X_t + A_t^{-1} g_t, \quad (10)$$

where  $A_t$  = Hessian matrix.

(f) Bayesian regulation backpropagation (training function: trainbr)

In the Bayesian regulation framework, weights and biases are assumed to be random variables with specified distributions on the basis of LM optimization. The combination of squared errors and weights are minimized for the generalization of a remarkable network. The Bayesian regulation occurs within the LM algorithm. Jacobian performance,  $jX$ , is determined on the basis of the weight and bias of variable  $X$  through backpropagation. The variables are adjusted according to the LM algorithm.

$$\begin{aligned} jj &= jX * jX, \\ je &= jX * E, \\ dX &= -(jj + I * mu)/je, \end{aligned} \quad (11)$$

where  $E$  = all errors,  $I$  = identity matrix.

## Results and discussion

### Effect of reactor impeller speed on PCL molecular weight

Figures 5, 6 and 7 explain the relationship between biopolymerization time and molecular weight at 60, 70, and 80 °C with a  $\epsilon$ -CL-to-toluene ratio of 1:2 (v/v). Each plot exhibits the molecular weights of polymers obtained

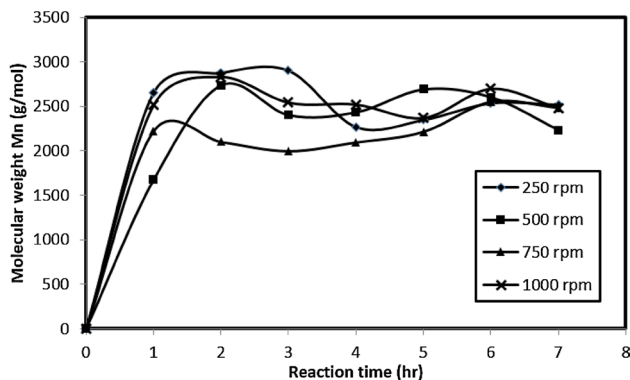


Fig. 5 Effect of reactor impeller speed on molecular weight with reaction time at 60 °C

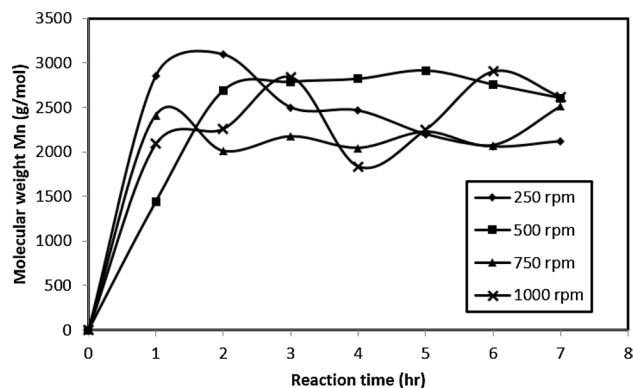


Fig. 6 Effect of reactor impeller speed on molecular weight with reaction time at 70 °C

at varied impeller speeds, namely 250, 500, 750, and 1000 rpm. The trend of the plot is uniform at all impeller speeds.

Figure 5 shows the behavior of the impeller at 60 °C. An  $M_n$  value of 2902 g/mol is obtained at the third hour at rotation speed of 250 rpm. An  $M_n$  value of 1677 g/mol is obtained at the first hour and at rotational speed of 500 rpm. Figure 6 shows the variation of molecular weight with reaction time at a temperature of 70 °C. Figure 6 indicates the trend for 70 °C at rotational speeds of 250, 500, 750, and 1000 rpm. An  $M_n$  value of 3095 g/mol is obtained at the second hour at rotational speed of 250 rpm. An  $M_n$  value of 1444 g/mol is obtained at the first hour at rotational speed of 500 rpm. Figure 7 shows the effect of 80 °C on the rotational speed of the impeller. An  $M_n$  values of 3091 and 2194 g/mol are obtained at rotational speed of 500 and 1000 rpm, respectively, in the third hour. This non-uniform behavior can be due to the mixing effect, which may have affected enzyme behavior, which in turn may have affected the polymer molecular weight. Possible explanations may lie in the differences among the heat transfer rates.

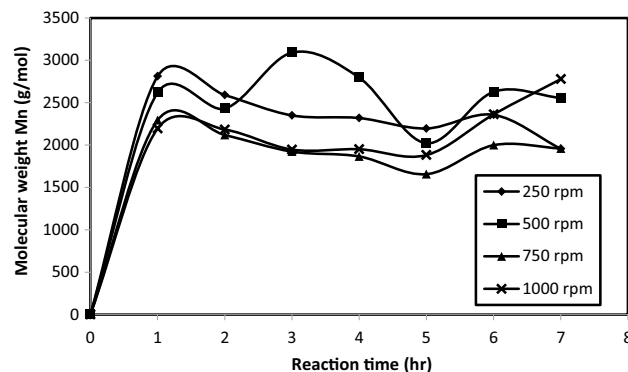


Fig. 7 Effect of reactor impeller speed on molecular weight with reaction time at 80 °C

The overall conclusion based on Figs. 5, 6 and 7 is that the maximum molecular weight of PCL obtained is 3095 g/mol at 70 °C when 250 rpm for 2 h. Furthermore, at high impeller speed, the molecular weight produced by PCL is consistently varying. This instability can be due to the formation of vortices and dead zones in the reactor, leading to poor mixing effect. This phenomenon contributes to the collisions of radicals and the termination of polymerization. The increase in reaction temperature, from 70 to 80 °C, causes a slight reduction in the molecular weight of the sample (Fig. 7). At high temperatures, the propagation rate is enhanced. However, the increased movement of chains inside the reactor causes a thermal effect that resulted in the depletion of the monomer. The results obtained at low impeller speeds are better than those obtained at high values. The molecular weight obtained from this analysis clearly indicates that the molecular weight lies within the range of 1500–3000 g/mol. Therefore, uniformity in the molecular weight values is maintained.

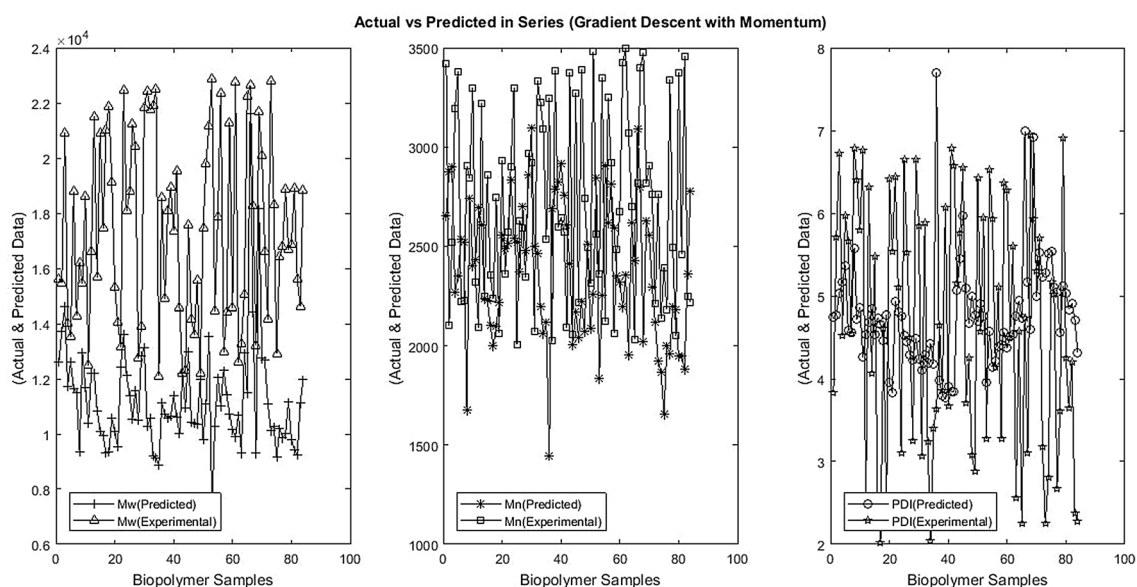
### Comparison of neural network algorithm results

The objective of this research is to compare different types of backpropagation training algorithms used in building FFNN models for predicting the  $M_n$ ,  $M_w$ , and PDI at the end of PCL biopolymerization in a batch mode. The best training algorithm is selected on the basis of the fitting of the graphs with least MAPE, MAE, and RMSE values for the training of the model for PCL molecular weight prediction. The datasets used in the research are data collected in

a scaled-up PCL production study. The data contain 84 samples that exhibit six features, namely polymerization time, reaction temperature, impeller speed,  $M_n$ ,  $M_w$ , and PDI. A total of 58 samples are randomly selected as training dataset (70%). Another 13 samples are randomly selected as validation dataset (15%) used for the construction of neural network models using different training algorithms, and the last 13 samples are used as testing dataset (15%) to access the models. For this purpose, three different performance functions: MAE, MAPE and MSE, are adopted.

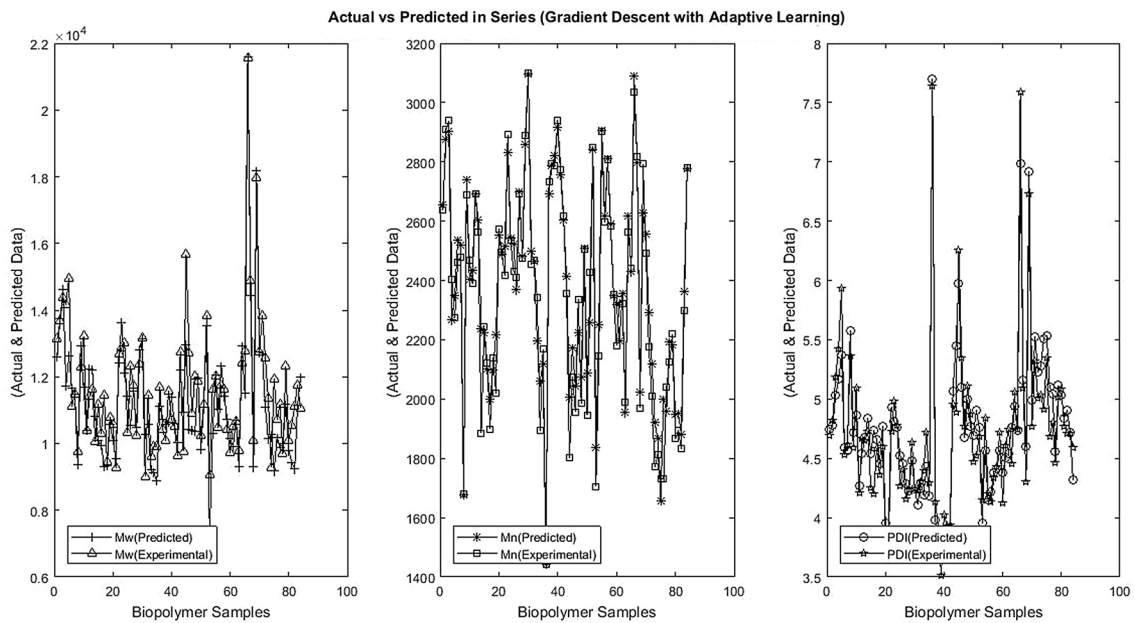
The parameters of the network are as follows: LR, 0.05; maximum amount of epoch, 1000; and level of error, 0.001. During the data processing, the number of neurons in the hidden layer is adjusted in numerous iterations, ranging from 1 to 20, and different weightages for training, testing, and validation are performed. The best performance for each algorithm is tabulated as follows:

Figures 8, 9, 10, 11, 12, 13, 14, 15, 16, 17 and 18 show the comparison between the actual and predicted data, that is,  $M_n$ ,  $M_w$ , and PDI for various training algorithms. The gradient descent with momentum backpropagation training algorithm has one hidden neuron as the optimal value because it provides the lowest MSE value (1.326; Table 3). Gradient descent with adaptive LR backpropagation training algorithm has five hidden neuron as optimum because the lowest MSE value is 0.395. Gradient descent with momentum and adaptive LR backpropagation training algorithm has 10 hidden neurons because the lowest MSE value is 0.333. Resilient backpropagation training algorithm has three hidden neuron because the lowest MSE value is 0.318. Conjugate

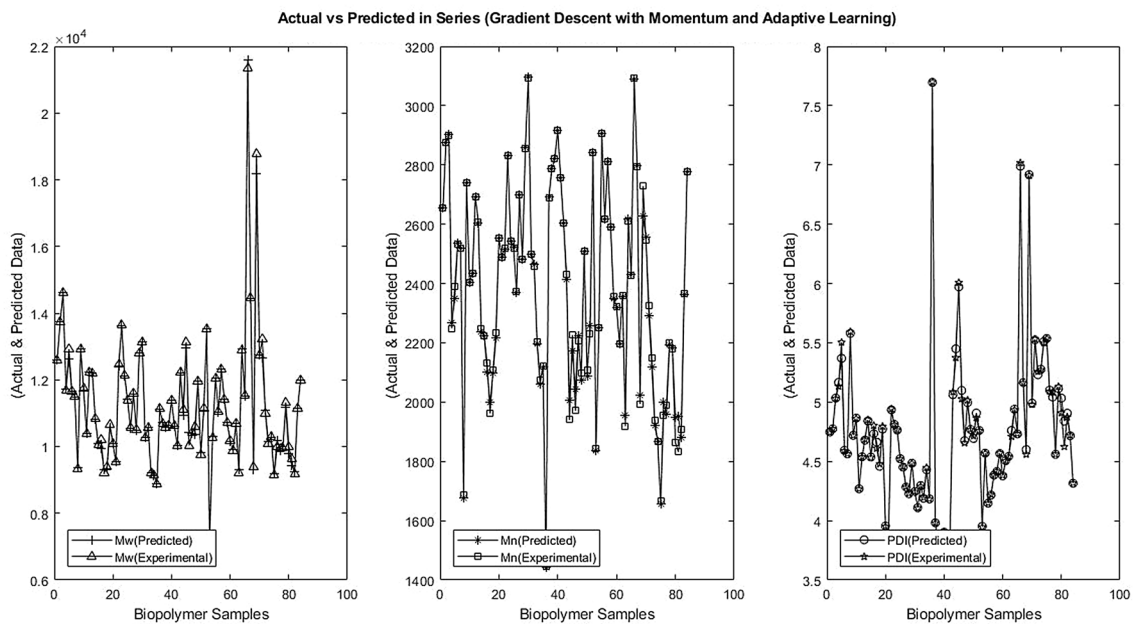


**Fig. 8** Graph of actual data versus predicted data in series under gradient descent with momentum backpropagation training algorithm for combined data of training, testing and validation





**Fig. 9** Graph of actual data versus predicted data in series under gradient descent with adaptive learning rate backpropagation training algorithm for combined data of training, testing and validation

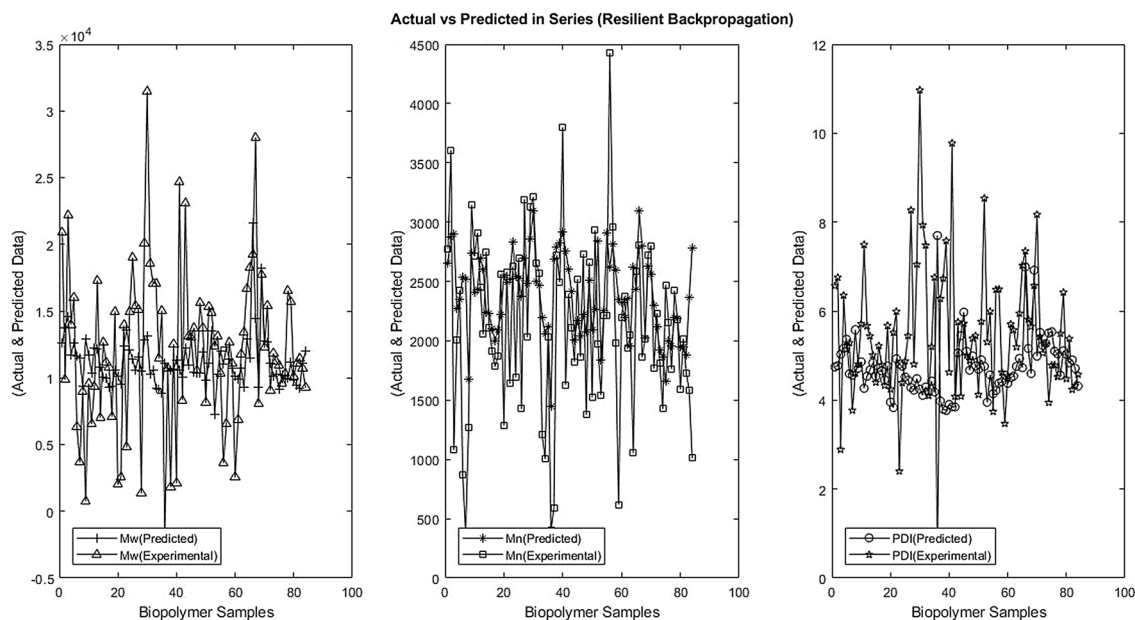


**Fig. 10** Graph of actual data versus predicted data in series under gradient descent with momentum and adaptive learning rate backpropagation training algorithm for combined data of training, testing and validation

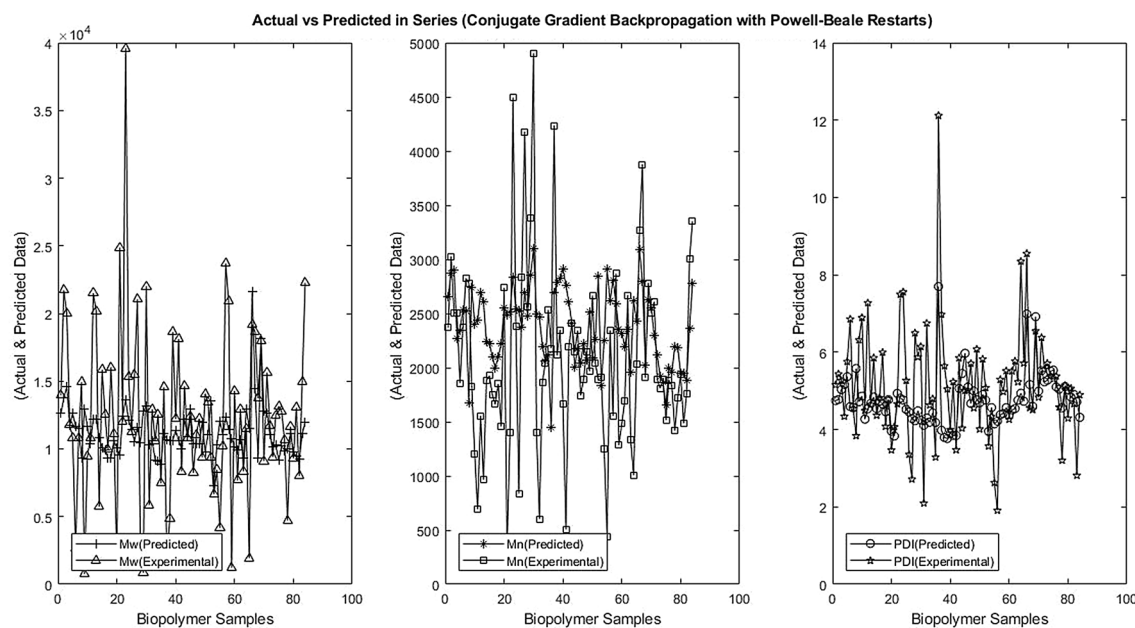
gradient backpropagation with Powell–Beale restart training algorithm shows two hidden neuron because the lowest MSE value is 0.335. Conjugate gradient backpropagation with Polak–Ribière updates 12 hidden neuron as optimum because the lowest MSE value is 0.315. SCG

backpropagation training algorithm has 10 hidden neurons because the lowest MSE value is 0.206.

LM backpropagation has 10 hidden neurons as optimum because the lowest MSE value is 0.095. One-step secant backpropagation has seven hidden neurons as optimum because the lowest MSE value is 0.315. BFGS



**Fig. 11** Graph of actual data versus predicted data in series under resilient backpropagation training algorithm for combined data of training, testing and validation

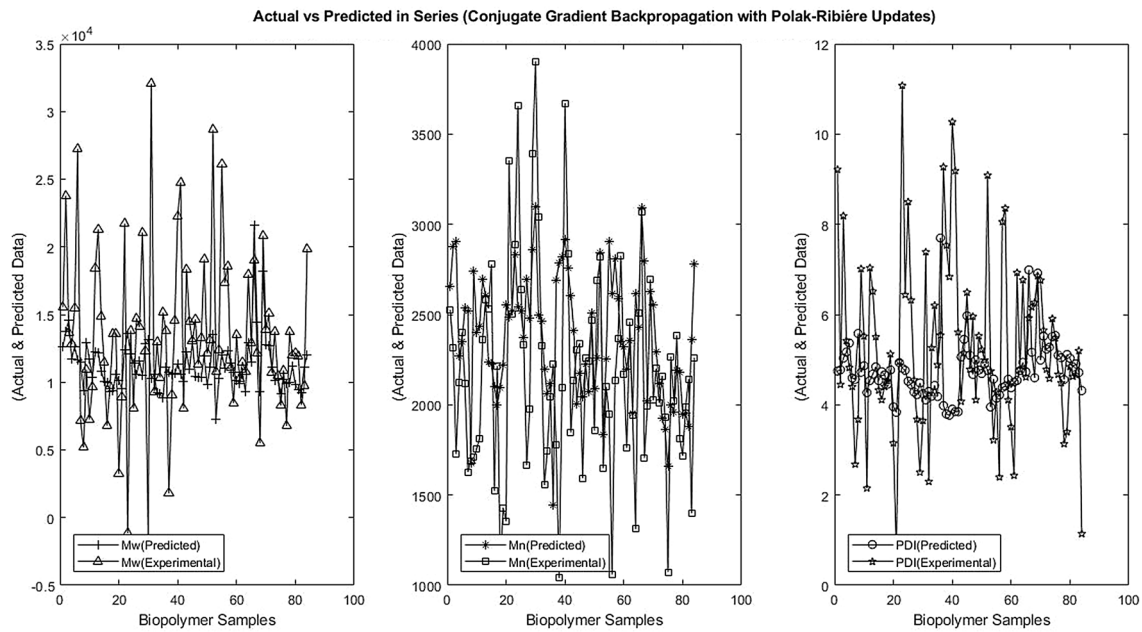


**Fig. 12** Graph of actual data versus predicted data in series under conjugate gradient backpropagation with Powell–Beale restarts training algorithm for combined data of training, testing and validation

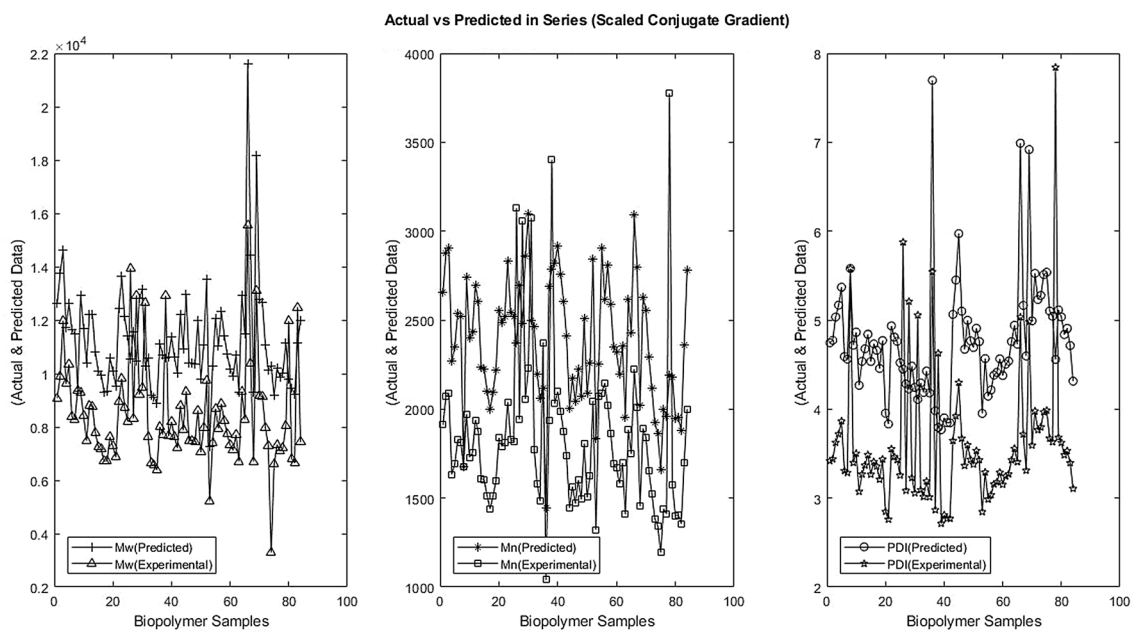
quasi-Newton backpropagation has eight hidden neurons as optimum because the lowest MSE value is 0.309. Bayesian regulation backpropagation has 11 hidden neurons as optimum because the lowest MSE value is 0.100.

The estimation errors in terms of MAPE, MAE and RMSE for the FFNN using different training algorithms are

represented in Tables 4, 5 and 6 for training, testing, and validation data, respectively. The models with low MAPE, MAE, and MAPE values are considered remarkable fit models. Two differences are observed in each training algorithm, namely calculation method and convergence quality, which



**Fig. 13** Graph of actual data versus predicted data in series under conjugate gradient backpropagation with Polak–Ribière updates training algorithm for combined data of training, testing and validation

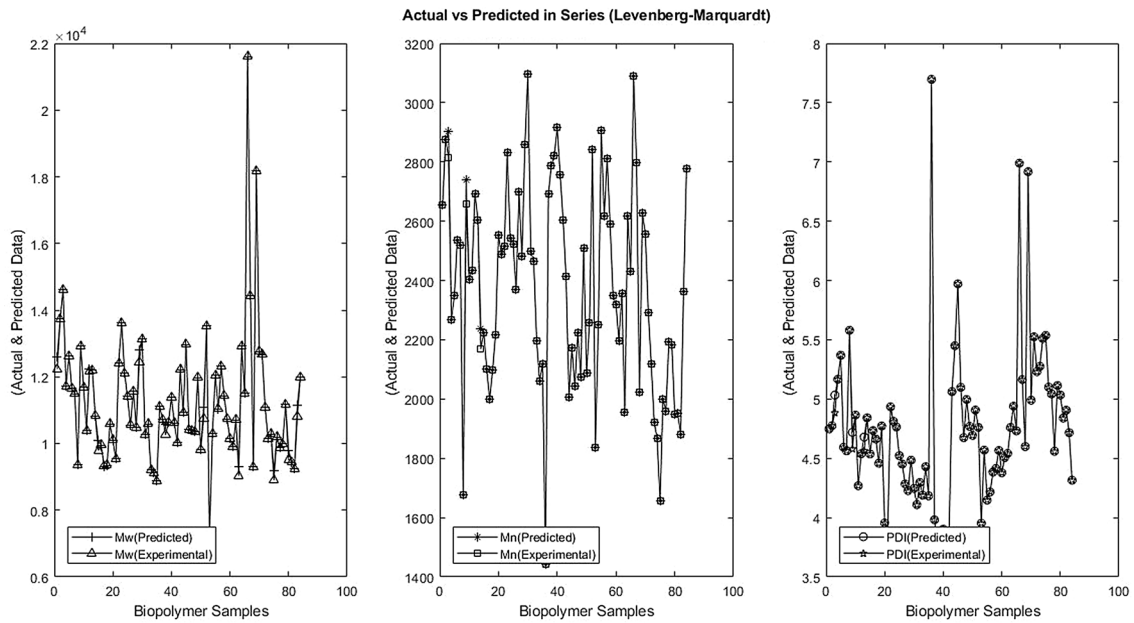


**Fig. 14** Graph of actual data versus predicted data in series under scaled conjugate gradient backpropagation training algorithm for combined data of training, testing and validation

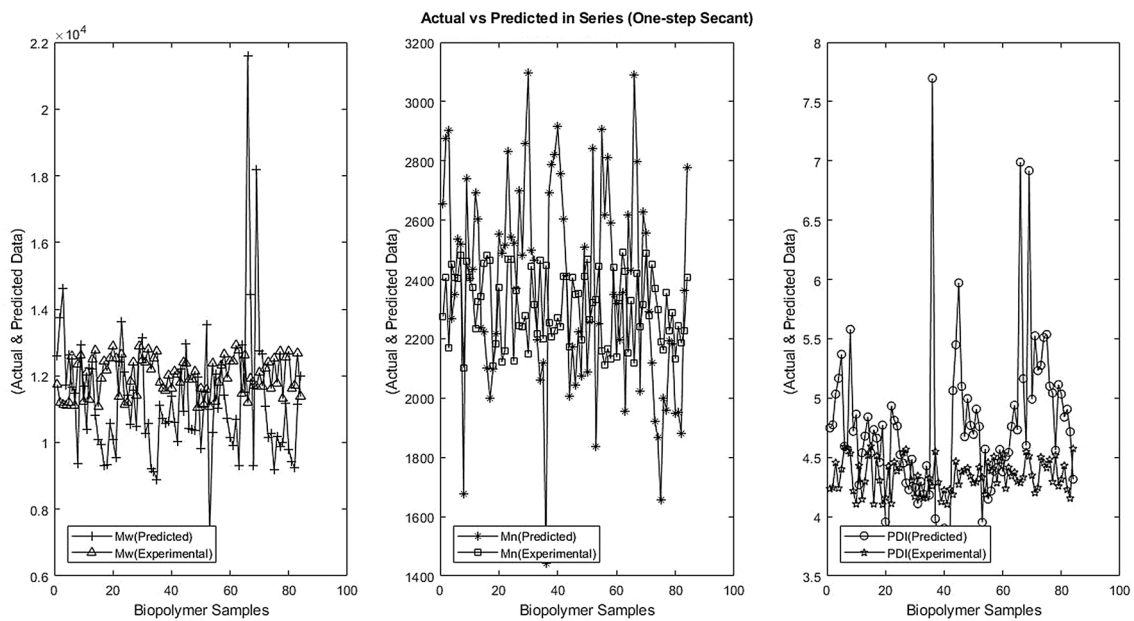
can also be known as the mathematical equation behind each training algorithm and prediction ability.

Figures 8, 9, 10, 11, 12, 13, 14, 15, 16, 17 and 18 show the comparison of the accuracy of all the different types of training algorithms using the predicted and real data. The

X axis shows the biopolymer samples, whereas the Y axis shows the value for  $M_n$ ,  $M_w$ , and PDI. Three subplots are provided for each training algorithm, one subplot for each modeled variable. Furthermore, training, testing, and validation data are distinguished in the plot. The plots show



**Fig. 15** Graph of actual data versus predicted data in series under Levenberg–Marquardt backpropagation training algorithm for combined data of training, testing and validation

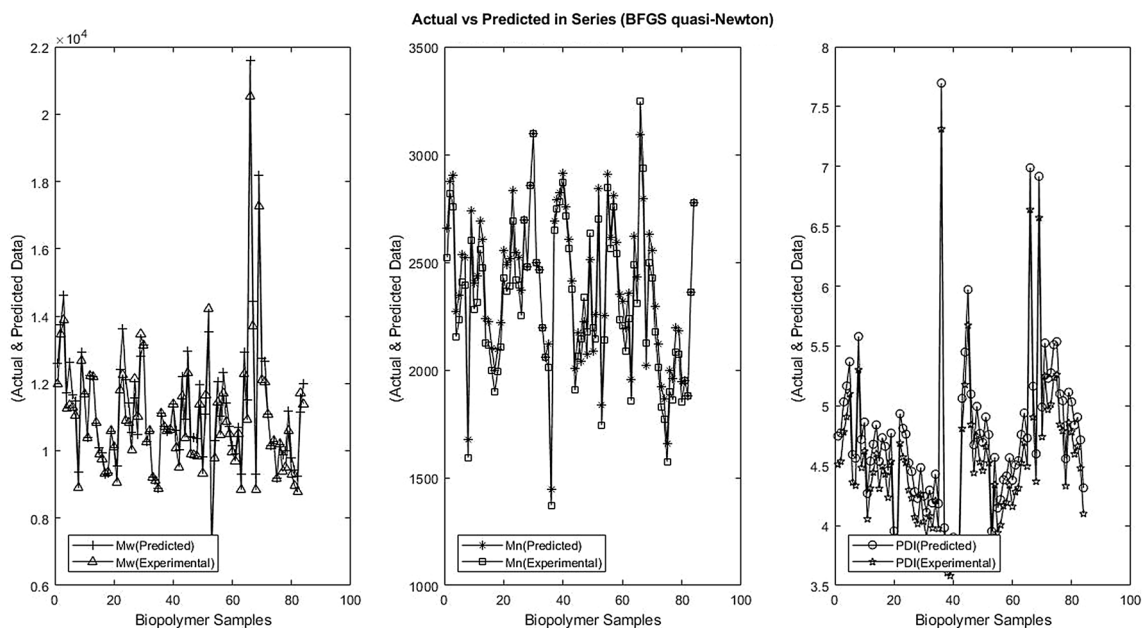


**Fig. 16** Graph of actual data versus predicted data in series under one-step secant backpropagation training algorithm for combined data of training, testing and validation

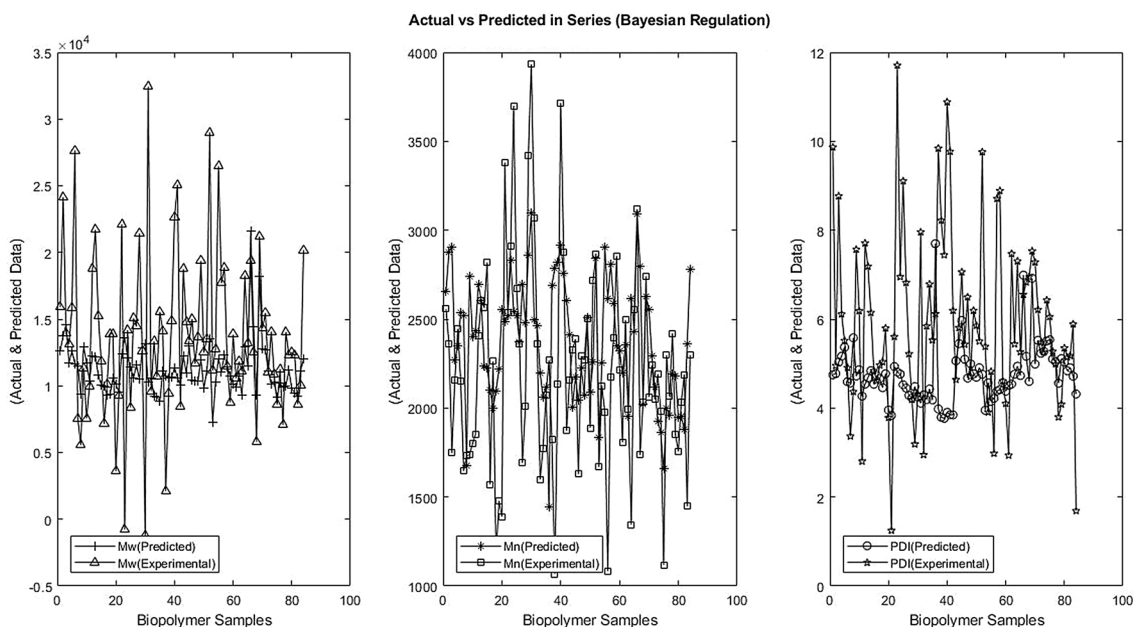
that the closer the value of the predicted value is to the real data, the more suitable the training algorithm is for the process, indicating higher accuracy of the model.

The results in Tables 4, 5 and 6 show that LM algorithm has the best training algorithm with minimum error in most of the cases. LM algorithm is an effective training algorithm

with the fastest convergence. Additive momentum algorithm has the worst prediction for biopolymerization process because the errors are greatly higher than those of other training algorithms. Additive momentum algorithm can smoothen the training by considering previously executed weight changes. However, the momentum coefficient is held



**Fig. 17** Graph of actual data versus predicted data in series under BFGS quasi-Newton backpropagation training algorithm for combined data of training, testing and validation



**Fig. 18** Graph of actual data versus predicted data in series under Bayesian regulation backpropagation training algorithm for combined data of training, testing and validation

constant, and no concrete reason assumes that such a strategy is optimum. Thus, the wrong coefficient for additive momentum algorithm might have led to the worst performance among all training algorithms.

Among the self-adaptive LR training algorithms, the GDA has a large estimation error with the worst training performance. The GDx has slightly considerable training performance and estimation error. In GDA, the training algorithm adapts the LR that corresponds to the current

**Table 3** Mean-squared error in test data for different number of hidden neuron in different training algorithms

Training algorithm	Mean squared error for different number of hidden neuron									
	1	2	3	4	5	6	7	8	9	10
trainlm	0.423	0.289	0.317	0.144	0.144	0.154	0.148	0.137	0.198	0.095
trainbfg	0.426	0.395	0.419	0.864	0.391	0.355	0.405	0.309	0.491	0.375
trainbr	0.397	0.237	0.138	0.141	0.137	0.131	0.109	0.125	0.081	0.119
traincgb	0.436	0.335	0.391	0.402	0.399	0.397	0.354	0.413	0.390	0.394
traincgp	0.433	0.468	0.405	0.362	0.399	0.434	0.393	0.401	0.373	0.396
traingda	0.442	0.420	0.442	0.437	0.395	0.396	0.481	0.437	0.397	0.456
traingdm	1.326	1.467	9.347	7.572	7.134	4.114	23.084	7.367	11.071	7.138
traingdx	0.425	0.404	0.851	0.698	0.381	0.505	0.440	0.604	0.856	0.333
trainoss	0.405	0.403	0.396	0.340	0.395	0.421	0.315	0.384	0.407	0.387
trainrp	0.436	0.411	0.318	0.332	0.325	0.369	0.403	0.387	0.348	0.325
trainscg	0.424	0.446	0.418	0.400	0.442	0.416	0.381	0.410	0.372	0.206
	11	12	13	14	15	16	17	18	19	20
trainlm	0.138	0.125	0.135	0.135	0.361	0.102	0.130	0.133	0.123	0.180
trainbfg	0.335	0.370	0.327	0.315	0.343	0.333	0.383	0.357	0.330	0.392
trainbr	0.100	0.385	0.125	0.172	0.394	0.101	0.395	0.394	0.395	0.174
traincgb	0.416	0.381	0.426	0.411	0.374	0.576	0.371	0.378	0.386	0.354
traincgp	0.463	0.315	0.400	0.424	0.437	0.340	0.323	0.359	0.422	0.387
traingda	0.466	0.459	0.523	0.484	0.523	0.458	0.401	0.473	0.467	0.522
traingdm	7.013	9.967	4.693	13.469	24.657	13.769	20.388	19.132	12.498	13.520
traingdx	0.428	0.353	0.583	1.331	0.385	0.802	0.336	0.348	0.353	0.745
trainoss	0.370	0.377	0.435	0.394	0.357	0.381	0.394	0.338	0.383	0.402
trainrp	0.372	0.393	0.342	0.414	0.368	0.319	0.449	0.418	0.421	0.453
trainscg	0.401	0.398	0.393	0.360	0.390	0.368	0.350	0.378	0.404	0.362

**Table 4** Comparison of NN-based approach using different training algorithms in terms of estimation error in training data (MAE, MAPE and MSE)

		Additive momentum	Self-adaptive learning rate		Resilient	Conjugate gradient			Quasi-Newton			Bayesian regulation
		GDM	GDA	GDX	RP	CGB	CGP	SCG	LM	BFG	OSS	BR
$M_n$	MAPE(%)	12.410	9.782	8.431	6.314	7.582	7.891	8.126	4.512	6.210	5.982	7.891
	MAE	62.050	48.910	42.155	31.570	37.910	39.455	40.630	20.760	31.050	29.210	39.455
	RMSE	27.063	19.630	16.341	16.213	16.079	16.520	17.610	15.844	15.044	17.745	15.978
$M_w$	MAPE(%)	14.231	8.782	9.412	7.643	8.563	7.562	7.983	5.310	6.431	6.331	8.278
	MAE	71.155	43.910	47.060	38.215	42.815	37.810	39.915	26.55	32.155	31.655	41.390
	RMSE	70.279	32.507	31.229	31.232	34.667	33.632	31.662	33.952	31.711	34.978	33.519
PDI	MAPE(%)	10.782	7.391	7.783	5.213	7.323	8.012	9.123	3.210	8.289	4.241	7.829
	MAE	1.775	0.884	1.6163	0.471	2.467	0.655	3.574	0.450	2.303	2.169	1.089
	RMSE	1.923	0.839	1.913	1.783	1.877	1.758	1.887	1.485	1.561	1.487	1.683

error surface encountered during the training. However, the LR is not subject to specific process during training but it is increased or decreased by a constant factor,  $\beta_{\Delta+}$ , and  $\beta_{\Delta}$  (Patricia et al. 2010). GDX, which combines momentum and adaptive LR and presents the advantages of GDA and GDX, is the latest training algorithm proposed.

As discussed earlier, resilient training algorithm is a remarkable training algorithm with relatively high accuracy, convergence speed, and robustness. Therefore, the performance of resilient algorithm is relatively better than that of other training algorithms. Among the conjugate gradient training algorithms, conjugate gradient with Powell–Beale restarts

**Table 5** Comparison of NN-based approach using different training algorithms in terms of estimation error in testing data (MAE, MAPE, and MSE)

	Additive momentum GDM	Self-adaptive learning rate			Resilient			Conjugate gradient			Quasi-Newton			Bayesian regulation	
		GDA	GDX	RP	CGB	CGP	SCG	LM	BFG	OSS	BR				
$M_n$	MAPE (%)	12.26	11.41	8.98	9.87	10.74	11.30	7.51	9.81	8.38	10.72				
	MAE	61.31	57.06	44.92	49.36	53.71	56.48	34.56	49.05	40.93	53.61				
	RMSE	24.61	22.12	23.07	20.94	22.49	24.48	26.38	23.77	24.86	21.71				
$M_w$	MAPE(%)	12.67	11.75	12.65	12.17	10.80	11.01	10.60	10.87	11.54	12.00				
	MAE	63.36	58.76	63.27	60.87	54.01	55.07	53.00	54.36	57.71	59.99				
	RMSE	46.91	45.99	51.70	49.28	48.04	43.68	46.78	53.60	63.76	48.58				
PDI	MAPE(%)	10.13	11.30	9.05	10.87	11.59	12.35	6.90	10.91	7.57	11.19				
	MAE	1.21	2.35	1.13	3.66	1.00	4.84	0.97	3.03	3.87	1.56				
	RMSE	1.15	2.78	3.10	2.79	2.54	2.56	3.19	2.05	2.65	2.41				

**Table 6** Comparison of NN-based approach using different training algorithms in terms of estimation error in validation data (MAE, MAPE and MSE)

	Additive momentum GDM	Self-adaptive learning rate			Resilient			Conjugate gradient			Quasi-Newton			Bayesian regulation	
		GDA	GDX	RP	CGB	CGP	SCG	LM	BFG	OSS	BR				
$M_n$	MAPE (%)	8.21688	7.5879	5.74574	6.4447	7.57536	6.41954	3.97056	5.6511	5.539332	6.991426				
	MAE	41.0844	37.9395	28.7287	32.2235	37.8768	32.0977	18.2688	28.2555	27.04846	34.95713				
	RMSE	16.4892	14.7069	14.75383	13.66715	15.8592	13.9119	13.94272	13.69004	16.43187	14.156508				
$M_w$	MAPE(%)	7.37688	8.4708	6.95513	7.27855	7.25952	6.30657	4.6728	5.85221	5.862506	7.334308				
	MAE	36.8844	42.354	34.77565	36.39275	36.2976	31.53285	23.364	29.26105	29.31253	36.67154				
	RMSE	27.30588	28.1061	28.42112	29.46695	32.28672	25.01298	29.87776	28.85701	32.389628	29.697834				
PDI	MAPE(%)	6.20844	7.0047	4.74383	6.22455	7.69152	7.20717	2.8248	7.54299	3.927166	6.936494				
	MAE	0.74256	1.45467	0.42861	2.09695	0.6288	2.82346	0.396	2.09573	2.008494	0.964854				
	RMSE	0.70476	1.7217	1.62253	1.59545	1.68768	1.49073	1.3068	1.42051	1.376962	1.491138				

and Polak–Ribière conjugate gradient provide similar results in error estimation and training performance, whereas SCG provides relatively poor results. The conjugate-based training algorithms have the same estimation errors as other training algorithms. Among the quasi-Newton training algorithms, LM has the least estimation errors and the best training performance. Bayesian regulation training algorithm can perform early stopping during training. However, this property is unsuitable for this application because biopolymerization is an extremely sensitive process and often results in large estimation errors.

## Conclusion

Biopolymerization was investigated in a bioreactor setup. Polymerization temperature and impeller speed were optimized. An  $M_n$  of 3095 Da was obtained at 70 °C at impeller speed of 250 rpm and time of 2 h. A total of 11 different training algorithms in the neural network, including (1) additive momentum (gradient descent with momentum), (2) self-adaptive LR (variable LR gradient descent and gradient descent with momentum and adaptive LR), (3) resilient backpropagation, (4) conjugate gradient backpropagation (SCG, conjugate gradient with Powell–Beale restarts, Fletcher–Powell conjugate gradient and Polak–Ribière conjugate gradient), and (5) quasi-Newton (LM, BFGS quasi-Newton, and one-step secant); and (6) Bayesian regulation propagation (Bayesian regularization) are tested. The network parameters include maximum epoch of 1000, LR of 0.05, and target error of 0.001. LM is the most suitable training algorithm for biopolymerization. The MAPE values for  $M_n$ ,  $M_w$ , and PDI are 4.512, 5.31, and 3.21%, respectively.

**Acknowledgements** This work was supported by RU Geran-Faculti Program Grant RF008A-2018 by University of Malaya.

## References

- Ajay K, Gross RA (2000) *Candida antarctica* lipase B catalyzed polycaprolactone synthesis: effects of organic media and temperature. *Biomacromology* 1:133–138. <https://doi.org/10.1021/cr9901211>
- Arumugasamy SK, Ahmad Z (2010) *Candida antarctica* as catalyst for polycaprolactone synthesis: effect of temperature and solvents. *Asia-Pac J Chem Eng* 6:398–405. <https://doi.org/10.1002/apj.583>
- Arumugasamy SK, Uzir MH, Ahmad Z (2012) Modeling of polycaprolactone production from  $\epsilon$ -caprolactone using neural network. In: *Neural information processing: 19th international conference, ICONIP 2012, Doha, Qatar, November 12–15, 2012, proceedings, part II*. Huang T, Zeng Z, Li C, Leung CS. Springer, Berlin, Heidelberg, pp 444–451. [https://doi.org/10.1007/978-3-642-34481-7\\_54](https://doi.org/10.1007/978-3-642-34481-7_54)
- Bassi AK, Gough JE, Zakikhani M, Downes S (2011) The chemical and physical properties of poly( $\epsilon$ -caprolactone) Scaffolds functionalised with poly(vinyl phosphonic acid-co-acrylic acid). *J Tissue Eng* 2011:615328. <https://doi.org/10.4061/2011/615328>
- Chaibakhsh N, Abdul Rahman MB, Basri M, Salleh AB, Abdul Rahman RNZ (2012) Response surface modeling and optimization of immobilized *Candida antarctica* lipase-catalyzed production of dicarboxylic acid ester. *Chem Prod Process Model* 7:1–13. <https://doi.org/10.1515/1934-2659.1483>
- Forouzanfar M, Dajani HR, Groza VZ, Bolic M, Rajan S (2010) Comparison of Feed-Forward Neural Network training algorithms for oscillometric blood pressure estimation. 4th International workshop on soft computing applications. <https://doi.org/10.1109/sofa.2010.5565614>
- Ghaffari A, Abdollahi H, Khoshayand MR, Bozchalooi IS, Dadgar A, Rafiee-Tehrani M (2006) Performance comparison of neural network training algorithms in modeling of bimodal drug delivery. *Int J Pharm* 327:126–138. <https://doi.org/10.1016/j.ijpharm.2006.07.056>
- Heaton J (2008). *Introduction to neural networks with Java*, Heaton Research
- Hornik K, Stinchcombe M, White H (1989) Multilayer feedforward networks are universal approximators. *Neural Netw* 2(5):359–366
- Hunsen M, Azim A, Mang H, Wallner SR, Ronkvist A, Xie W, Gross RA (2007) A cutinase with polyester synthesis activity. *Macromolecules* 40:148–150. <https://doi.org/10.1021/ma062095g>
- Karsoliya S (2012) Approximating number of hidden layer neurons in multiple hidden layer BPNN architecture. *Int J Eng Trends Technol* 3(6):4
- Khatti T, Naderi H, Kalantar SM (2017) Application of ANN and RSM techniques for modeling electrospinning process of polycaprolactone. *Neural Comput Appl*. <https://doi.org/10.1007/s00521-017-2996-6>
- Kobayashi S, Makino A (2009) Enzymatic polymer synthesis: an opportunity for green polymer chemistry. *Chem Rev* 109:5288–5353. <https://doi.org/10.1021/cr900165z>
- Labet M, Thielemans W (2009) Synthesis of polycaprolactone: a review. *Chem Soc Rev* 38:3484–3504. <https://doi.org/10.1039/b820162p>
- Lee-Cosio BM, Delgado-Mata C, Ibanez J (2012) ANN for gesture recognition using accelerometer data. *Procedia Technol* 3:109–120. <https://doi.org/10.1016/j.protcy.2012.03.012>
- Patricia M, Janusz K, Witold P (2010) *Soft computing for recognition based biometrics*. Springer, Berlin
- Sulaiman J, Wahab SH (2018) Heavy rainfall forecasting model using artificial neural network for flood prone area. In: Kim K, Kim H, Baek N (eds) *IT convergence and security 2017. Lecture notes in electrical engineering*, vol 449. Springer, Singapore
- Torres DPM, Gonçalves MDPF, Teixeira JA, Rodrigues LR (2010) Galacto-oligosaccharides: production, properties, applications, and significance as prebiotics. *Comp Rev Food Sci Food Saf* 9(5):438–454. <https://doi.org/10.1111/j.1541-4337.2010.00119.x>
- Uyama H, Kikuchi H, Takeya K, Kobayashi S (1996) Lipase-catalyzed ring-opening polymerization and copolymerization of 15-pentadecanolide. *Acta Polym* 47(8):357–360. <https://doi.org/10.1002/actp.1996.010470807>
- Varma IK, Albertsson AC, Rajkhowa R, Srivastava RK (2005) Enzyme catalyzed synthesis of polyesters. *Prog Polym Sci* 30(10):949–981. <https://doi.org/10.1016/j.progpolymsci.2005.06.010>
- Xinxing P, Lee B, Chunrong Z (2013) A comparison of neural network backpropagation algorithms for electricity load forecasting. 2013 IEEE international workshop on intelligent energy systems (IWIES). <https://doi.org/10.1109/iwies.2013.6698556>