Original Article

Kaiwen Zheng, Jiaoxue Shi and Shichang Chen*

Utilizing ResNet for enhanced quality prediction in PET production: an AI-driven approach

https://doi.org/10.1515/polyeng-2024-0048
Received February 28, 2024; accepted May 19, 2024;
published online July 1, 2024

Abstract: To promote theoretical understanding for optimizing the entire process parameters (temperature, pressure, flow rate, etc.) and quality indicators (molar fraction, end-group concentration, and number-average molecular weight) in the industrial production of polyethylene terephthalate (PET), a dataset construction for production parameters and product quality indicators was accomplished in conjunction with industrial process simulation software. A complete deep learning workflow including data collection, dataset construction, model training, and evaluation was established. A prediction method for process-product quality of PET production based on the residual neural network (ResNet) network was proposed to reduce the complexity of quality control in polyester production. The results show that compared to traditional convolutional neural network (CNN), ResNet has higher accuracy ($R^2 \geq 0.9998$) in predicting the PET production process and product quality. It can accurately establish the mapping relationship between production parameters and product quality indicators, providing theoretical guidance for intelligent production.

Keywords: PET production; artificial neural networks; deep learning; quality prediction

***Corresponding author: Shichang Chen**, National & Local United Engineering Laboratory on Textile Fiber Materials and Processing Technology, Zhejiang Sci-Tech University, Hangzhou 310018, China; and Zhejiang Modern Textile Technology Innovation Center, Shaoxing 312030, China, E-mail: scchen@zstu.edu.cn. https://orcid.org/0000-0002-3974-6913
Kaiwen Zheng, National & Local United Engineering Laboratory on Textile Fiber Materials and Processing Technology, Zhejiang Sci-Tech University, Hangzhou 310018, China; and Zhejiang Modern Textile Technology Innovation Center, Shaoxing 312030, China
Jiaoxue Shi, Zhejiang Guxiandao Polyester Dope Dyed Yarn Co., Ltd., Shaoxing 312000, China

1 Introduction

Polyethylene terephthalate (PET) is a semi-crystalline polymer, and due to its excellent mechanical properties, corrosion resistance, and fiber-forming ability, it is one of the most widely used fiber materials at present. With the development of modern industry, the single-line production capacity of PET often reaches tens of thousands or even millions of tons. Given the complexity of large-scale production process control and the limited referential value of experimental data, the prediction of polymer product quality determined by raw materials and process control remains a challenging issue.[1–3]

The quality of polymer products is determined by their microscopic molecular structure and fundamental molecular parameters, such as the average molecular weight of homopolymers, molecular weight distribution (MWD), molecular weight distribution, and composition distribution of copolymers. These microscopic molecular structures and basic molecular parameters are directly influenced by production parameters.[4–6] However, the parameters involved in the PET production process are numerous and highly interdependent, such as temperature, pressure, residence time, stirring and mixing, feed rate, pH value, devolatilization, etc. At the same time, some parameters are difficult to measure in real-time, making the process of parameter control and optimization face many challenges and difficulties.[7,8]

To solve the problems of multi-parameter optimization and high production testing costs in chemical production, computer production simulation has emerged. Mechanism-based modeling is a key part of modern industrial production, helping engineers and decision-makers predict, analyze, and optimize production processes. Mechanism-based modeling of PET mainly revolves around various stages of the PET production process, using mathematical models and software tools such as Aspen, ChemCAD, Pro/II, etc. Although mechanism-based modeling remains the mainstream method of chemical process modeling, the presence of uncertainties and interference factors in actual industrial processes, as well as various industrial scaling

effects difficult for mechanism analysis, pose significant challenges to the application of strict mechanism models in industrial processes, especially since strict mechanism models are computationally complex and time-consuming, making their application in industrial online calculations very difficult.[9,10]

In recent years, the successful application of machine learning (ML) algorithms or artificial intelligence (AI) in areas such as voice recognition, text classification, and medical research has attracted widespread attention. Meanwhile, ML has also achieved significant breakthroughs in the field of materials research. With the emergence and rapid development of machine learning, data-driven production assistance has become one of the hot research directions. Machine learning models, trained with large amounts of data, discover correlations between complex phenomena, overcoming the limitations of human thought. This provides a new approach for chemical production simulation. For example, Patrick Taylor and Gareth Conduit successfully used ML to establish the mapping relationship between the microchemical changes and creep in alloys.[11] Deep learning (DL), as an algorithm in ML, has also received a high degree of attention. Soowan Park and Karuppasamy Pandian Marimuthu proposed a DL-based nanoindentation method to reduce the complexity of evaluating the mechanical properties of polymers. By conducting nanoindentation tests on polycarbonate and polymethyl methacrylate, the performance of the trained deep neural network (DNN) models was validated experimentally. The trained DNN models were able to accurately predict material parameters, aligning well with literature findings.[12] Fei Lu[13] proposed two models, artificial neural networks (ANN) and convolutional neural networks (CNN), to predict the glass-forming ability of various amorphous alloys. The prediction accuracy of these two models reached 0.77623 and 0.71693, respectively, both over 19 % higher than that reported in standard predictions. This indicates that ANN can model complex nonlinear relationships between input and output variables in the manufacturing process, which traditional statistical methods may fail to capture. By integrating ANN with real-time monitoring systems, they can provide immediate feedback on product quality, allowing for instantaneous adjustment of process parameters during production.[14] Therefore, ANN can be used to explore the relationship between polyester production quality and processing parameters. Based on this, this paper uses the residual neural network (ResNet) model to predict polyester production results and compares it with traditional CNN models in prediction accuracy.

2 Materials and methods

2.1 Data collection

Deep learning requires a dataset to enable the neural network to undergo multiple iterations of learning, finding the mapping relationship between feature values and output values. This dataset describes the hidden correspondence between processing parameters and product performance.[15] The dataset for this paper was generated by simulated production in Aspen.

Aspen Plus is a chemical process simulation software used for designing, simulating, optimizing, and operating chemical processes. The Aspen software allows engineers to design and optimize processes without actual experiments, thus saving time and resources. This paper builds the deep learning dataset by collecting data from PET production simulations in Aspen under different production parameters.

The production process adopts the traditional three-kettle technique for Aspen process construction, as shown in Figure 1.[16,17] In the feature selection part, this study mainly focuses on the pressure and temperature of each kettle. Parameters such as feed ratio, flow rate, stirring, and mixing as constants. The performance indicators of the product are represented by the molar fraction of small molecules in the product, end-group number, and molecular weight. These small molecules include EG, DEG, TPA, and H_2O. To improve the generalization ability of the model, the dataset construction process needs to include some outlier values. The values should be larger than the designed production range to ensure that outliers can be captured during data collection. The temperature and pressure values for each reaction kettle are shown in Table 1. To enhance sample coverage, Latin hypercube sampling (LHS) is used within the value range as input for Aspen. LHS is a statistical method for

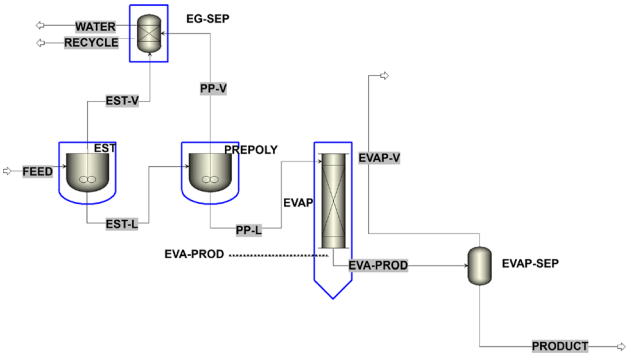

Figure 1: Aspen PET three-kettle production model.

Table 1: Input parameters value range.

Blocks	Temperature (°C)	Pressure (Pa)
EST	208–312	6.5×10^5–9.7×10^5
PREPOLY	230–280	5.3×10^3–8.0×10^3
EVAP	240–300	1.0×10^2–1.6×10^2

generating random samples of variables, particularly useful in computer simulations and complex models. The core idea of LHS is to ensure that the entire range of each input variable is represented, thus providing better sample coverage than traditional random sampling. This method is especially suitable for simulations that need to consider multiple input parameters or variables, where traditional Monte Carlo methods might require a large number of samples to achieve the same level of accuracy.[18]

In the process of feature construction, there are differences in dimension and magnitude among different production parameters, which can increase the difficulty of training. It is necessary to normalize the feature values, as shown in the formula below.

$$X_{\text{norm}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{1}$$

Among them, x is the current value, $x_{\min}$ is the minimum value, $x_{\max}$ is the maximum value, and X_{norm} is the normalized value of x.

The dataset in this paper consists of 80,000 data points, with an input dimension of 6 and an output dimension of 8. In this paper, the dataset is split in an 8:2 ratio, meaning 80 % of the dataset is divided into a training set, used to train the model's parameters. This is the main data source for the model to learn the mapping relationship from input data to output results; 20 % is divided into a test set, used after the model development process to obtain an accurate estimate of the model's generalization ability on unknown data. 80 % of the training set is used for training, and 20 % for validation.[19,20]

2.2 Machine learning

This paper employs CNN and ResNet to establish the mapping relationship between processing parameters and product performance in the PET production process. Based on the study in literature,[13] which used fully connected ANN and CNN to predict the crystallization properties of materials, this study chose three different structures of CNN as the research subjects. The first type is the CNN with two convolutional layers as mentioned in the literature, the second type is a CNN with five convolutional layers, and the third type is a CNN with residual blocks, namely ResNet. The machine learning system, as shown in Figure 2, mainly consists of three parts: dataset construction, model training, and model evaluation.

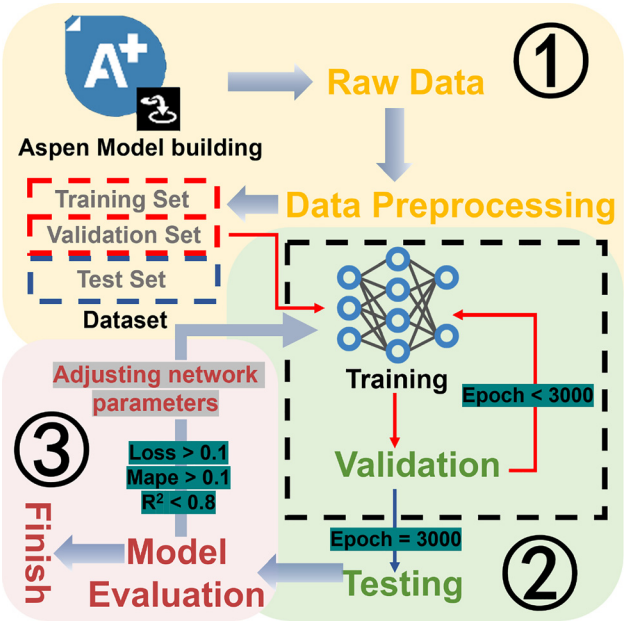

Figure 2: Machine learning system.

2.2.1 Convolutional neural network

A CNN is a type of neural network implemented by adding convolutional layers to the traditional ANN. An artificial neural network consists of three parts: input layer, hidden layers, and output layer. The input layer receives the input data, which in this study are the pressure and temperature of each kettle, serving as the feature values of the dataset, with each neuron corresponding to a feature of the input data. The hidden layers are located between the input and output layers and can be one or more. The output layer produces the network's output; for regression tasks, this is the predicted result.[21,22]

Neural network training is divided into two main parts: forward propagation of information and backward propagation of errors. The forward propagation of information in a single neuron is calculated as Figure 3.

Where x is the input, w is the weight, and f is the activation function. After obtaining the predicted value, the global error is calculated. If the global error exceeds a preset error, backpropagation is executed combined with the gradient descent algorithm, modifying the weights and biases of each layer. The calculations for the backward propagation of error are as follows:

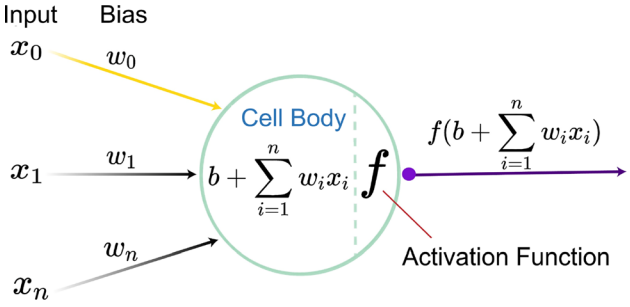

Figure 3: Forward propagation of information in a neuron.

$$E = \frac{1}{2}\sum_{i=1}^{n}\left(y_i^d - y_i^p\right)^2 \tag{2}$$

$$w_{ij}(k+1) = w_{ij}(k) - \eta\frac{\partial E(k)}{\partial w_{ij}(k)} \tag{3}$$

$$b_{ij}(k+1) = b_{ij}(k) - \eta\frac{\partial E(k)}{\partial b_{ij}(k)} \tag{4}$$

where y^d_i is the target value, y^p_i is the predicted value, E is the global error, η is the learning rate, $W_{ij}(k)$ and $B_{ij}(k)$ are the connection weights and biases between layers.[23]

A CNN consists of an input layer, convolutional layers, pooling layers, fully connected layers, and an output layer, among which the convolutional and pooling layers are the most important parts of the serial neural network.[24] The convolutional layer is used for convolution operations, aiming to extract input data through convolution kernels and the principle of convolution is shown in Figure 4. The pooling layer serves to reduce the dimensionality of the data. Typically, a flattening layer is added to CNNs to transform the data from the pooling layers into a one-dimensional vector, facilitating the connections of neurons in the fully connected layers.

The addition of convolutional layers addresses the issue of high-dimensional training in fully connected networks, where an excessive number of parameters increases the difficulty of training. In this layer, the input data is convoluted with a set of learnable filters (also known as convolutional kernels). Each filter slides over the input data, computing the dot product between the filter and the data it covers, thus generating new feature maps. This process captures local features within the input data.[25] In the convolutional layer, each neuron is connected only to a small region of the input data (known as the receptive field), rather than being fully connected. This means that each neuron only needs to focus on a small part of the input data, effectively reducing the number of parameters in the model. Neurons in the convolutional layer use the same weights and biases, a process known as parameter sharing. This significantly reduces the number of parameters in the model, making CNNs easier to train and less prone to overfitting compared to fully connected networks. By stacking multiple convolutional layers in a CNN, the network can learn and recognize patterns at different scales. However, when the number of layers in a CNN increases to a certain extent, the error rate tends to saturate or even increase. This is mainly due to problems like vanishing gradients and exploding gradients, which are more severe in deep neural networks.[26]

2.2.2 ResNet neural network

The ResNet network, building upon the foundation of CNN, incorporates residual blocks and skip connections (also known as skip connection), effectively solving the problem of training deep neural networks. These shortcut paths allow gradients to be directly back-propagated to earlier layers, enabling ResNet to train deeper networks more effectively.

As shown in Figure 5, (a) represents a traditional CNN network, (b) represents a ResNet network. $H(x)$ is the output function after convolution operations, x represents input parameters, and y represents output. ResNet refers to a stack of several layers as a block. For a given block, the function it can approximate is denoted as $F(x)$. Instead of having $F(x)$ directly learn the expected underlying mapping $H(x)$, it is more advantageous to learn the residual $H(x) - x$, thus defining $F(x) = H(x) - x$. Consequently, the original forward path becomes $F(x) + x$, using $F(x) + x$ to approximate $H(x)$. The authors believe this approach may be easier to optimize because, compared to making $F(x)$ learn an identity

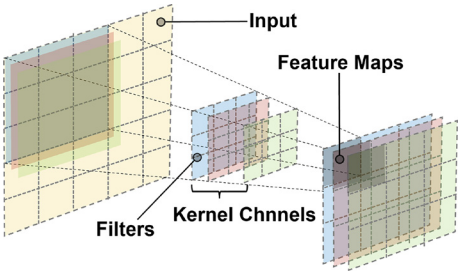

Figure 4: Principle of convolution.

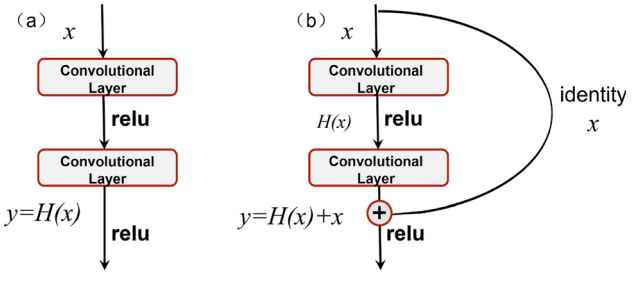

Figure 5: The difference between CNN and ResNet.

mapping, making $F(x)$ approximate zero is much simpler. Thus, for redundant blocks, simply having $F(x) \to 0$ achieves an identity mapping without reducing performance. In this way, each layer learns the residual or difference between the input and the desired output, rather than directly learning the output. This enables ResNet to better handle the vanishing gradient problem in deep networks.[27]

2.2.3 Hyperparameter settings

In the construction and training of neural networks, setting a large number of hyperparameters is involved. This includes the choice of normalization methods, loss functions, activation functions, and optimizers.

Batch normalization (BN) is a technique used to speed up the training of neural networks. It normalizes the inputs to each layer, making their mean 0 and variance 1, thereby addressing the so-called "internal covariate shift" problem, which refers to the inconsistency in data distribution between layers of the network.[28]

Batch normalization is typically performed after convolutional or fully connected operations and before the activation function. Specifically, for a given batch of data $B = \{x_1, x_2, \ldots, x_m\}$, the batch normalization operation can be divided into the following steps:

Compute the mean μ_b and variance σ^2 of the batch data B. Normalize the batch data B, as shown in equation (5):

$$\widehat{x}_i = \frac{(x_i - u_b)}{\sqrt{(\sigma^2 + \epsilon)}} \tag{5}$$

where ε is a very small number to prevent division by zero. Scale and shift the normalized data, as shown in equation (6):

$$y_i = \gamma \widehat{x}_i + \beta \tag{6}$$

where γ and β are learnable parameters used to restore the original distribution of the data.

In this paper, Huber loss is used for the calculation of the loss value. A loss function measures the difference between the predicted values of a neural network and the actual values, serving as a key guiding factor for the optimization algorithm to find the best parameters. It not only quantifies the prediction error but also guides the direction of network weight updates through gradient information. Huber loss is a commonly used loss function in regression problems, especially when data may contain outliers or noise. It combines the characteristics of mean squared error and absolute error, reducing the impact of outliers on the model while maintaining sufficient differentiability and mitigating the impact of outliers. The derivative of Huber loss is also relatively easy to calculate, making it convenient for use in optimization algorithms like gradient descent.[29]

$$\text{loss} = \begin{cases} \frac{1}{2}a^2 & \text{if } |a| \le \delta \\ \delta|a| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases} \tag{7}$$

a represents the difference between the predicted and actual values, and δ is a hyperparameter that determines the point at which the loss function transitions from quadratic loss to linear loss. In this paper, Huber loss with $\delta = 1$ is used to calculate the loss value.

Learning rate (LR) is a key hyperparameter in neural network training, controlling the speed at which model weights are updated during training. The learning rate is a positive real number, typically between 0 and 1. It is used to scale the magnitude of weight and bias updates, as in equations (3) and (4). If LR is too high, convergence can be difficult; if LR is too low, convergence can be slow.[30] In this study, a dynamic learning rate is chosen, decreasing as the number of epochs increases. In the early stages of training, a higher LR speeds up model convergence, while in the later stages, a lower LR helps stabilize convergence. The learning rate update strategy is as follows:

$$\text{LR} = \text{Floor}\,(\text{epoch}/\text{step})*\delta \tag{8}$$

Floor(x) is the floor function, *step* is the step length, and δ is the scaling coefficient, where *step* = 100 and δ = 0.8.

The optimizer (optimization function) is an algorithm used to adjust network parameters to minimize the loss function. Optimization algorithms iteratively update the weights (w) and biases (b) of the neural network, enabling the model to better fit the training data. In this study, Adam is chosen as the optimization algorithm for the model. The Adam optimization algorithm combines the advantages of many previous optimization algorithms and has shown fast, stable training performance in many practical applications. The Adam algorithm maintains estimates of the first moment (the moving average of the gradient) and the second moment (the moving average of the squared gradient) in each step of iteration. It combines the advantages of Momentum and RMSProp with bias correction, and its calculation process is as follows:[31]

Set the learning rate α, the decay rates of the first moment estimate β_1, the decay rates of the second moment estimate β_2, and a very small constant ε to maintain numerical stability.

Initialize the first and second moment estimates:

$$m = 0, v = 0 \tag{9}$$

Calculate the gradient:

θ_t is the parameter at time step t, f is the loss function.

$$g_t = \nabla f\,(\theta_t) \tag{10}$$

Update the first moment estimate (exponentially weighted moving average of the gradient):

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \tag{11}$$

Update the second moment estimate (exponentially weighted moving average of the squared gradient):

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \tag{12}$$

Bias correction:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{13}$$

Update parameters:

$$\theta_{t+1} = \theta_t - \alpha \cdot \frac{\hat{m}_t}{\sqrt{v_t} + \epsilon} \tag{14}$$

An activation function is a mathematical function in neural networks that transforms the input signal of a node (neuron). Its primary purpose is to convert the input signal into an output signal and to determine whether the neuron should be "activated". They introduce non-linear characteristics, enabling neural networks to fit complex, non-linear functions.[32] This study selects the LeakyReLu (α = 0.01) function as the activation function, which allows a small part of the negative input to pass through. Leaky ReLU addresses the "dying ReLU" problem in the ReLU (rectified linear unit) function by introducing a small positive slope α, ensuring that the output is not completely zero even if the input is negative. Like ReLU, it does not involve complex operations such as exponentials or logarithms, making the computation of Leaky ReLU highly efficient.[33]

$$\mathrm{Re}Lu(x) = \max(0, x) \tag{15}$$

$$\text{Leaky Re}Lu(x) = \begin{cases} \alpha x & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} \tag{16}$$

Additionally, the computational setup for this research included Windows 10 as the operating system, with the following hardware specifications:

(a) CPU: AMD Ryzen7 5800x
(b) GPU: RTX 3090 (24G)
(c) RAM: 32 GB.

3 Results and discussion

To evaluate model performance, this study utilizes three indicators: Loss, R^2, and MAPE, to characterize the model. Loss primarily represents the error between predicted values and actual values, and HuberLoss is used as the loss function in this study. R^2, or the coefficient of determination, is used to describe the degree of fit of the model to the data. Its value ranges between 0 and 1, mainly used in regression analysis. The closer the value of R^2 is to 1, the better the fit. MAPE is the mean absolute percentage error, used to measure the accuracy of the prediction model. The smaller the value of MAPE, the higher the accuracy of the prediction model. R^2 is primarily used to evaluate the goodness of fit of the model, while MAPE is used to evaluate the accuracy of the prediction model. Both have their limitations, so in practical applications, a combination of various evaluation indicators is often used. Their calculation expressions are as follows (y_i is the actual value, and $\hat{y}_i$ is the predicted value):

$$R^2 = 1 - \frac{\sum_{i=1}^{n} \left(y_i - \hat{y}_i \right)^2}{\sum_{i=1}^{n} \left(y_i - y^2 \right)^2} \tag{17}$$

$$\text{MAPE} = \frac{1}{n} \sum_{i=1}^{n} \frac{\left| y_i - \hat{y}_i \right|}{\left| y_i \right|} \tag{18}$$

Figure 6 shows the change in the degree of fit during the training process of models with different network structures. Graphs (a), (b), and (c) show the performance of different network structures in terms of loss, MAPE, and R^2 during the training process, respectively. The horizontal axis represents the epoch, i.e., the number of training rounds. From (a), it can be seen that around 1,000 iterations, the Loss value gradually stabilizes, indicating that the network has nearly converged. ConvNet1 and ConvNet2 show a smoother and more stable decline in Loss value during training compared to ResNet, but their converged Loss values are much higher than ResNet. This indicates that ConvNet1 and ConvNet2 have issues with gradient anishing and cannot fully learn the mapping relationship between processing parameters and product performance indicators in polyester production. In (b), it can be seen that the MAPE of the networks initially drops sharply. However, after a sharp decline, ConvNet1 and ConvNet2 tend to stabilize with minor fluctuations, suggesting that they cannot improve the accuracy of the model's predictions in subsequent learning and fail to accurately predict some dimensions of the output values. In contrast, ResNet, after a sharp decline, experiences significant fluctuations before stabilizing, suggesting that the high initial learning rate caused overly aggressive model parameter updates. With the decline in the learning rate, the model can steadily improve the accuracy of its predictions in subsequent iterations. In (c), the trend of R^2 is similar to MAPE. Both ConvNet1 and ConvNet2 rapidly rise to a high level at the beginning of training and then maintain

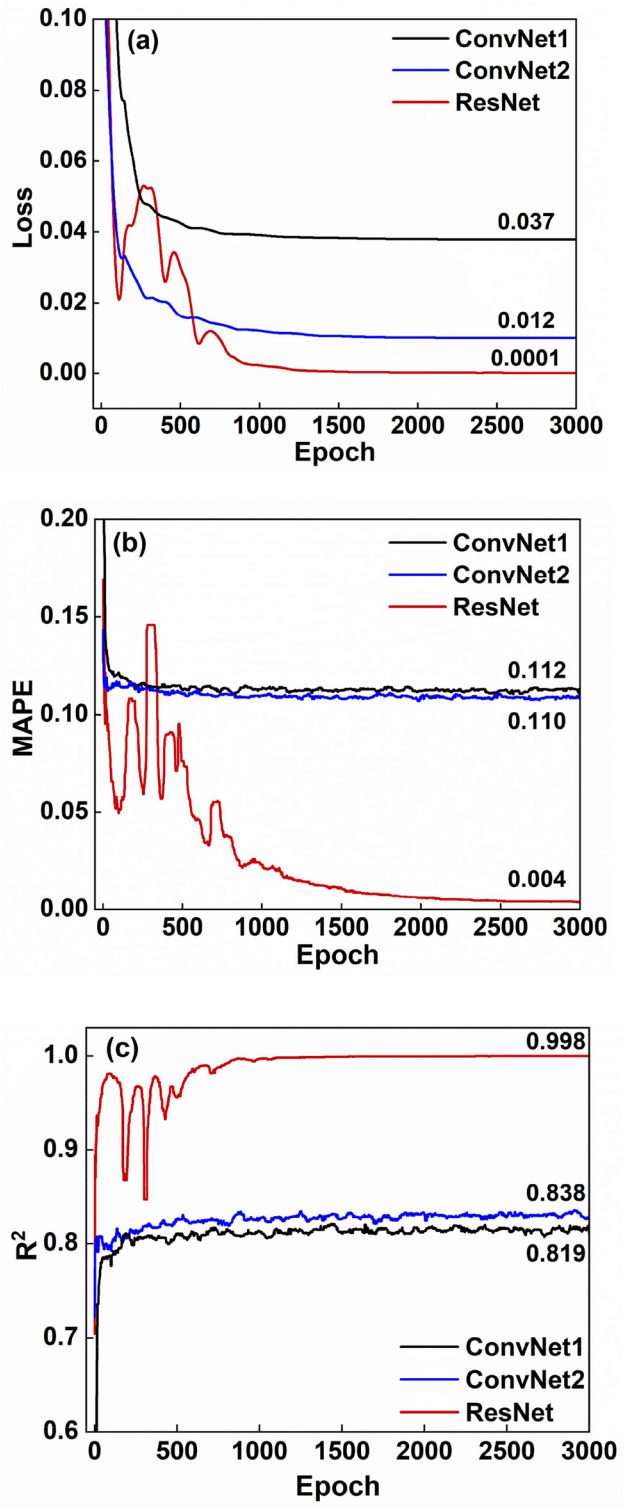

Figure 6: Learning curves of different networks during training.

minor fluctuations. ResNet also rapidly rises to a high level initially but, after several significant fluctuations, stabilizes. This indicates that the ResNet network structure is superior in fitting this task compared to ConvNet1 and ConvNet2. An overview of the training curves in Figure 6 shows that during the initial adaptation period of training, the loss of ConvNet1 and ConvNet2 sharply decreases, whereas ResNet experiences significant fluctuations and its loss remains higher than that of ConvNet1 and ConvNet2 at Epoch below 500. This occurs because, although residual connections are designed to help with gradient flow and alleviate the vanishing gradient problem in deep networks, the network still needs time to adapt to this type of connection. At this stage, the residual blocks may not yet have started effectively learning the 'residual', which ideally should represent the minimal necessary adjustments to transform the input into the output, leading to a large discrepancy between the initial outputs and the targets. In later training stages, the convergence time for ConvNet1 and ConvNet2 is earlier than that for ResNet. This is because ConvNet1 and ConvNet2 complete the learning of shallow mapping relationships early in training and fail to learn deep mapping relationships, whereas ResNet continues to learn complex mapping relationships in subsequent training, which is crucial for the overall network performance and generalization capability.

From the training process, it is evident that the ResNet network has stronger feature extraction capabilities and is more suitable for learning the mapping relationship between processing parameters and product performance indicators in polyester production. Meanwhile, the limited improvement of ConvNet2 compared to ConvNet1 also suggests that simply increasing the number of network layers does not effectively enhance network performance.

To verify the performance of different networks on the test set, scatter plots between predicted values and actual values were drawn, as shown in Figure 7. Table 2 is the output parameter table. The closer the scatter points are to the diagonal line, the closer the predicted values are to the actual values, indicating better predictive performance of the network for that output value. As can be seen from Figure 7, the scatter plots of predicted-actual values for each output show that the ResNet network maintains very good prediction accuracy ($R^2 \geq 0.999$) for every output value, while ConvNet1 and ConvNet2 networks have R^2 values ranging between 0.833–0.988 and 0.811–0.999 for different dimensions.

ConvNet1, for most parameters, has predicted values relatively close to the $y = x$ line, but there is a noticeable deviation in predictions for certain dimensions such as $[w_{H_2O}]$ $[w_{DEG}]$ and $[w_{EG}]$. ConvNet2 also shows a similar trend to ConvNet1 in predictions for most dimensions, but its predictions are closer to the actual values for $[M_{EG}]$ and $[M_{EA}]$. Overall, ResNet's predicted values are closer to the

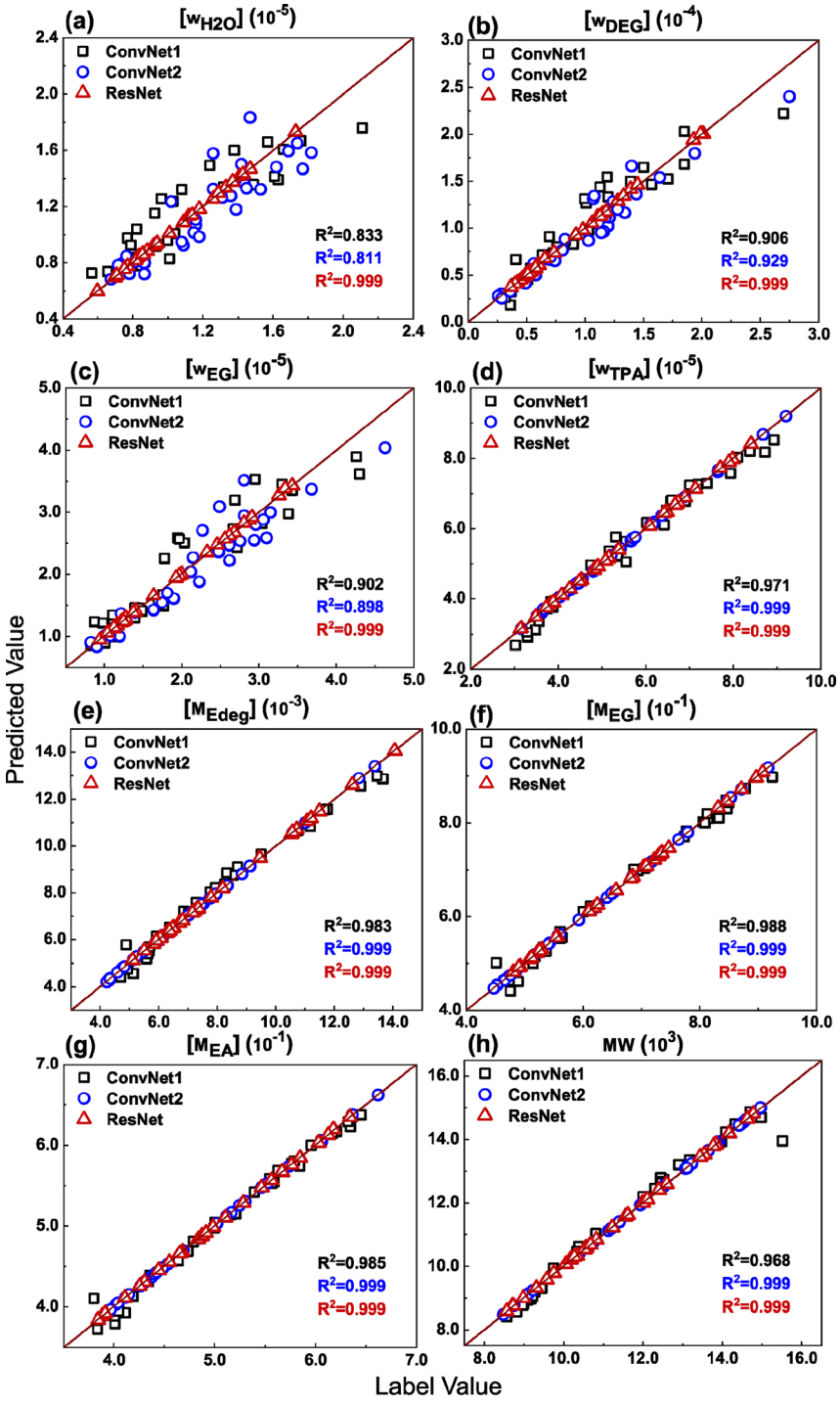

Figure 7: Performance of different network structures on the test set.

$y = x$ line, showing the best fit. This indicates that while ConvNet1 and ConvNet2 perform well in predicting certain dimensions, they cannot accurately predict all output dimensions simultaneously and exhibit significant accuracy differences across different outputs, especially in $[w_{H_20}][w_{DEG}]$ and $[w_{EG}]$ compared to ResNet. This suggests that ConvNet1 and ConvNet2 are unable to effectively learn the complex mapping relationships in the dataset during training. In contrast, ResNet provides accurate predictions across all dimensions, indicating that ResNet's network structure has stronger feature extraction and generalization capabilities.

To evaluate the model's robustness, 5 % Gaussian noise was added to the training set, and the model was trained under these noisy conditions. Adding Gaussian noise helps to simulate real-world conditions of data collection. Due to the

Table 2: Output parameters.

Parameter	Label	Unit
H_2O mole fraction	$[w_{H_20}]$	%
DEG mole fraction	$[w_{DEG}]$	%
EG mole fraction	$[w_{EG}]$	%
TPA mole fraction	$[w_{TPA}]$	%
T DEG	$[M_{Edeg}]$	kmol/h
T EG	$[M_{EG}]$	kmol/h
T TPA	$[M_{EA}]$	kmol/h
Molecular weight	MW	g/mol

complexity of the real production environment, measurements tend to be inaccurate, and the data obtained has certain errors. In a real production environment, the model needs to learn the mapping relationship between features and outputs from these error-containing data which can distort the training process. Therefore, training in a training set with added Gaussian noise can test whether the model can learn the corresponding mapping relationship from data with noise. Figure 8 shows the test results. The straight red line represents an equality of actual and predicted values,

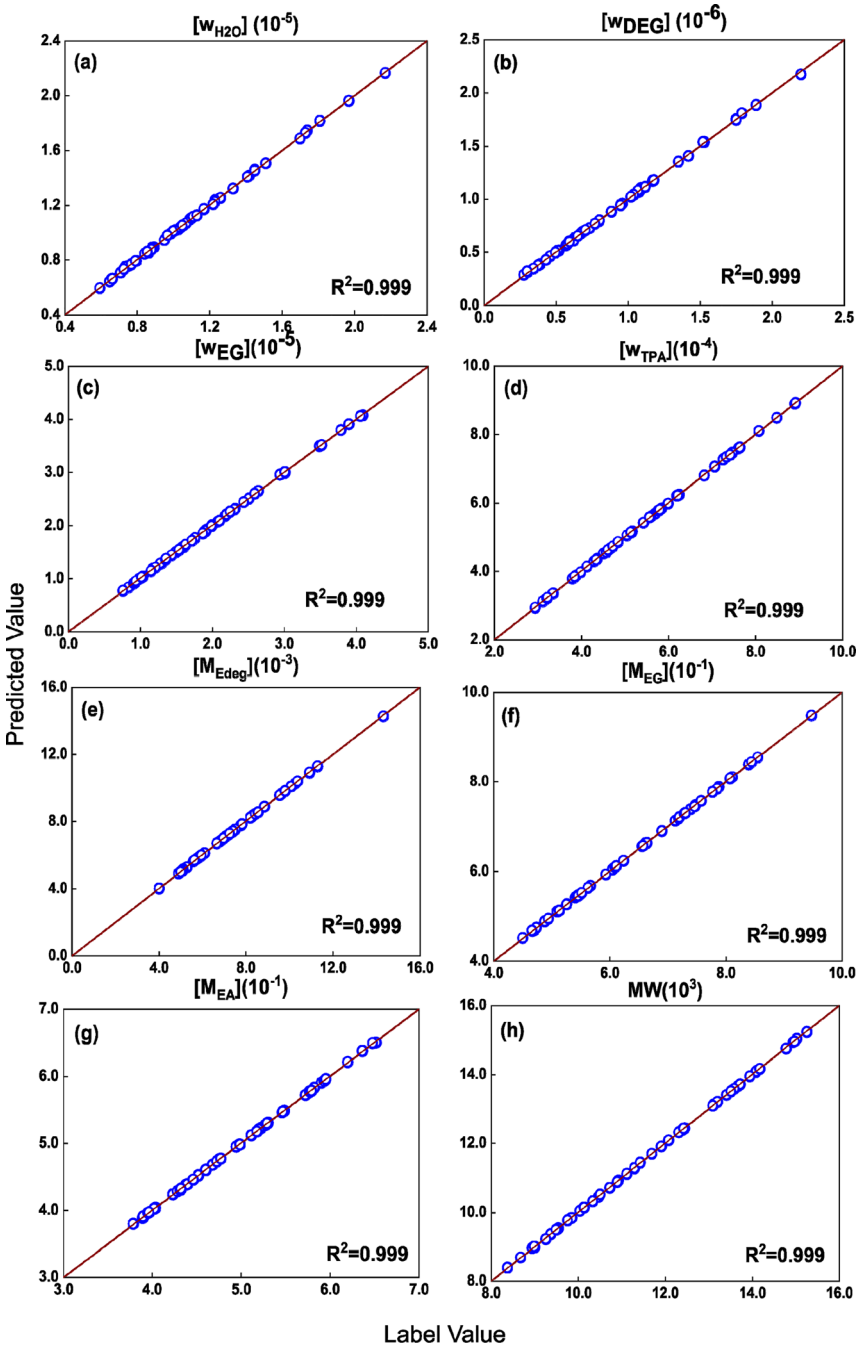

Figure 8: Robustness test of the ResNet network.

indicating 100 % accuracy. If the data points are primarily concentrated on the red line, it implies that the model can accurately predict the target values. As can be seen in Figure 8, the ResNet model, even when trained with the presence of noise, still maintains a high prediction accuracy ($R^2 \geq 0.999$), demonstrating good robustness. This indicates that the model can effectively handle noise in the input data and extract key features from it. This demonstrates the model's strong generalization ability, enabling it to cope with data variations that may be encountered in real-world applications.

4 Conclusions

This study utilized the ResNet network combined with Aspen engineering modeling technology. Using a PET production model built by Aspen and employing the LHS algorithm to obtain random combinations of input parameters, an input-output dataset was generated to validate the feasibility of predicting PET production results through neural networks.

The results show that the ResNet neural network, compared to two types of CNN networks, demonstrated higher prediction accuracy ($R^2 \geq 0.999$) in predicting PET production outcomes, with no significant accuracy differences across various output dimensions. While the two CNN networks (ConvNet1 and ConvNet2) approached the performance of ResNet in some dimensions, they could not accurately predict all output dimensions simultaneously. For instance, there were noticeable deviations in predictions for $[w_{H_20}]$ [wDEG] and [wEG], exhibiting significant accuracy differences across different outputs. Moreover, the ResNet, even after training with datasets containing Gaussian noise, maintained stable prediction accuracy, indicating that the ResNet network has good robustness in predicting PET production outcomes. We based on simulated data, explored the potential and feasibility of using artificial intelligence technology in polyester production. However, polyester production often involves other manufacturing stages such as thickening and spinning, which could be included in model training in future studies. Our research focused only on a forward model, which predicts product performance based on production process parameters. Future research could explore a reverse model, where product performance is the input and optimal process parameters are the output.

Research ethics: Not applicable.
Author contributions: The authors have accepted responsibility for the entire content of this manuscript and approved its submission.
Competing interests: The authors state no conflict of interest.
Research funding: This work was sponsored by the National Natural Science Foundation of China (52173047, 51803187) and the Key Research and Development Program of Zhejiang Province (2021C01020).
Data availability: Not applicable.

References

1. Kirshanov, K.; Toms, R.; Aliev, G.; Naumova, A.; Melnikov, P.; Gervald, A. Recent developments and perspectives of recycled poly(ethylene terephthalate)-based membranes: a review. *Membranes* **2022**, *12* (11), 1105.
2. Fischer Kerche, E.; Silveira Caldas, B. G.; Carvalho, R. F.; Amico, S. C. Mechanical response of sisal/glass fabrics reinforced polyester – polyethylene terephthalate foam core sandwich panels. *J. Sandw. Struct. Mater.* **2022**, *24* (6), 1993–2009.
3. Ji, H.; Song, M.; Wei, T.; Jiang, Q.; Ye, F.; Zhang, Y. Microstructure design of polyester industrial yarns with excellent flame retardancy and high strength. *J. Ind. Text.* **2022**, *52*, 152–168.
4. Gil, I. D.; Vargas, J. C.; Corriou, J. P. Optimal nonlinear control of an industrial emulsion polymerization reactor. *Chem. Eng. Res. Des.* **2016**, *111*, 63–82.
5. Dias, A. C. S. R.; da Silva, W. B.; Dutra, J. C. S. Propylene polymerization reactor control and estimation using a particle filter and neural network. *Macromol. React. Eng.* **2017**, *11* (6), 1700010.
6. Ghadipasha, N.; Zhu, W.; Romagnoli, J. A.; McAfee, T.; Zekoski, T.; Reed, W. F. Online Optimal feedback control of polymerization reactors: application to polymerization of acrylamide–water–potassium persulfate (KPS) system. *Ind. Eng. Chem. Res.* **2017**, *56* (25), 7322–7335.
7. Simitzis, J. Correlation between the production parameters of unsaturated polyesters with the aim to control the polyesterification process. *Eur. Polym. J.* **1988**, *24* (1), 87–92.
8. Carreau, P. J. Rheology of polymeric systems: principles and applications. *AIChE J.* **1999**, *45*, 1836.
9. Tsai, C. C.; Lin, S. T. Integration of modern computational chemistry and Aspen Plus for chemical process design. *AIChE J.* **2020**, *66* (10), 16987.
10. Bhaskar, V.; Gupta, S. K.; Ray, A. K. Multiobjective optimization of an industrial wiped film poly(ethylene terephthalate) reactor: some further insights. *Comput. Chem. Eng.* **2001**, *25* (2), 391–407.
11. Taylor, P. L.; Conduit, G. Machine learning superalloy microchemistry and creep strength from physical descriptors. *Comput. Mater. Sci.* **2023**, *227*, 112265.
12. Park, S.; Marimuthu, K. P.; Han, G.; Lee, H. Deep learning based nanoindentation method for evaluating mechanical properties of polymers. *Int. J. Mech. Sci.* **2023**, *246*, 108162.
13. Lu, F.; Liang, Y.; Wang, X.; Gao, T.; Chen, Q.; Liu, Y.; Zhou, Y.; Yuan, Y. Prediction of amorphous forming ability based on artificial neural network and convolutional neural network. *Comput. Mater. Sci.* **2022**, *210*, 111464.
14. Zhao, F.; Wu, J.; Zhao, Y.; Ji, X.; Zhou, L.; Sun, Z. A machine learning methodology for reliability evaluation of complex chemical production systems. *RSC Adv.* **2020**, *10* (34), 20374–20384.

15. Ye, H.; Du, Z.; Lu, H.; Tian, J.; Chen, L.; Lin, W. Using machine learning methods to predict VOC emissions in chemical production with hourly process parameters. *J. Clean. Prod.* **2022**, *369*, 133406.
16. Al-Malah, K. I. M. *Aspen Plus: chemical engineering applications*; John Wiley & Sons: Hoboken, 2022.
17. Seavey, K.; Liu, Y. A. *Step-growth polymerization process modeling and product design*; John Wiley & Sons: Hoboken, 2009.
18. Donovan, D.; Burrage, K.; Burrage, P.; McCourt, T. A.; Thompson, B.; Yazici, E. Ş. Estimates of the coverage of parameter space by latin hypercube and orthogonal array-based sampling. *Appl. Math. Model.* **2018**, *57*, 553–564.
19. Yuan, Z.; Niu, M. Q.; Ma, H.; Gao, T.; Zang, J.; Zhang, Y.; Chen, L. Q. Predicting mechanical behaviors of rubber materials with artificial neural networks. *Int. J. Mech. Sci.* **2023**, *249*, 108265.
20. Ye, S.; Li, B.; Li, Q.; Zhao, H. P.; Feng, X. Q. Deep neural network method for predicting the mechanical properties of composites. *Appl. Phys. Lett.* **2019**, *115* (16), 161901.
21. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521* (7553), 436–444.
22. Liu, X.; Tian, S.; Tao, F.; Yu, W. A review of artificial neural networks in the constitutive modeling of composite materials. *Composites, Part B* **2021**, *224*, 109152.
23. Rumelhart, D. E.; Hinton, G. E.; Williams, R. J. Learning representations by back-propagating errors. *Nature* **1986**, *323* (6088), 533–536.
24. Sun, M.; Song, Z.; Jiang, X.; Pan, J.; Pang, Y. Learning pooling for convolutional neural network. *Neurocomputing* **2017**, *224*, 96–104.
25. Derry, A.; Krzywinski, M.; Altman, N. Convolutional neural networks. *Nat. Methods* **2023**, *20* (9), 1269–1270.
26. Chegeni, M. K.; Rashno, A.; Fadaei, S. Convolution-layer parameters optimization in convolutional neural networks. *Knowledge-Based Syst.* **2023**, *261*, 110210.
27. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition, Las Vegas, US*, 2016.
28. Wang, J.; Li, S.; An, Z.; Jiang, X.; Qian, W.; Ji, S. Batch-normalized deep neural networks for achieving fast intelligent fault diagnosis of machines. *Neurocomputing* **2019**, *329*, 53–65.
29. Xie, J.; Liu, S.; Chen, J.; Jia, J. Huber loss based distributed robust learning algorithm for random vector functional-link network. *Artif. Intell. Rev.* **2023**, *56* (8), 8197–8218.
30. Jacobs, R. A. Increased rates of convergence through learning rate adaptation. *Neural Networks* **1988**, *1* (4), 295–307.
31. Reyad, M.; Sarhan, A. M.; Arafa, M. A modified Adam algorithm for deep neural network optimization. *Neural Comput. Appl.* **2023**, *35* (23), 17095–17112.
32. Krogh, A. What are artificial neural networks? *Nat. Biotechnol.* **2008**, *26* (2), 195–197.
33. Dubey, S. R.; Singh, S. K.; Chaudhuri, B. B. Activation functions in deep learning: a comprehensive survey and benchmark. *Neurocomputing* **2022**, *503*, 92–108.