Enhancement of multilayer perceptron model training accuracy through the optimization of hyperparameters: a case study of the quality prediction of injection-molded parts

Kun-Cheng Ke1 · Ming-Shyan Huang2

Received: 6 July 2021 / Accepted: 20 September 2021 / Published online: 26 September 2021
© The Author(s), under exclusive licence to Springer-Verlag London Ltd., part of Springer Nature 2021

Abstract
Injection molding has been broadly used in the mass production of plastic parts and must meet the requirements of efficiency and quality consistency. Machine learning can effectively predict the quality of injection-molded part. However, the performance of machine learning models largely depends on the accuracy of the training. Hyperparameters such as activation functions, momentum, and learning rate are crucial to the accuracy and efficiency of model training. This research aims to analyze the influence of hyperparameters on testing accuracy, explore the corresponding optimal learning rate, and provide the optimal training model for predicting the quality of injection-molded parts. In this study, stochastic gradient descent (SGD) and stochastic gradient descent with momentum (SGDM) are used to optimize the artificial neural network model. Through optimization of these training model hyperparameters, the width testing accuracy of the injection product is improved. The experimental results indicate that in the absence of momentum effects, all five activation functions can achieve more than 90% of the training accuracy with a learning rate of 0.1. Moreover, when optimized with the SGD, the learning rate of the Sigmoid activation function is 0.1, and the testing accuracy reaches 95.8%. Although momentum has the least influence on accuracy, it affects the convergence speed of the Sigmoid function, which reduces the number of required learning iterations (82.4% reduction rate). In summary, optimizing hyperparameter settings can improve the accuracy of model testing and markedly reduce training time.

Keywords Activation function · Hyperparameter · Injection molding · Learning rate · Machine learning · Momentum · Quality prediction · Sigmoid function · Stochastic gradient descent · SDM · SGDM

1 Introduction
Injection molding is a key step in polymer processing and comprises the five stages of clamping, filling, packing, cooling and plasticizing, and demolding. When polymer materials are molded, the melt and mold temperatures, filling speed, packing pressure, and time are the primary factors that affect the quality of the parts [1]. In particular, polymers are temperature sensitive; different temperatures exhibit distinct rheological properties that affect the flow properties of the melt. Melt filling is driven by pressure, and the required pressure is related to the setting of the forward screw speed. A filling speed that is too low or high results in short shots and jetting problems, respectively. In addition, the holding pressure (also called postfilling) can compensate for the gap between the polymer melt after cooling and shrinking in the mold cavity, ensuring that the finished product meets the size requirements. Therefore, machine settings also influence the quality of the final product. However, under the same machine settings, because of the adverse effects of actual machine movement, material stability, and environmental factors, this quality cannot be guaranteed.

To determine the actual flow behavior of the polymer melt, pressure sensors installed on the surface of the mold cavity can measure the pressure changes of the melt during
molding [2–4]. The temperature distribution of the melt on
the mold surface directly affects the quality of the product.
The use of a composite sensor to track melt pressure and
temperature changes during the injection molding process
reveals the relationship between pressure and tempera-
ture, which can be monitored to further control the volume
change in injection-molded products [4]. Furthermore, some
researchers have employed nondestructive ultrasonic sens-
ing technology to measure changing melt pressure in the
mold during the injection process, through which the melt
pressure state can be monitored without damaging the mold
structure and the various stages of the molding process
can be identified [5]. In addition, with the rise of Industry
4.0-related technologies, a new wealth of data from in-mold
sensors and machine data is becoming available to the plas-
tic and composite industry. These data can be used for qual-
ity prediction and process variation detection [6, 7].

Following the capturing of the molding data, diverse
methods can be applied to predict quality, including domain
knowledge, statistical methods, and artificial intelligence
techniques. In regard to domain knowledge, the pvT theorem
is crucial to describing the specific volume state of the poly-
er melt corresponding to pressure and temperature changes
during the injection molding process. The optimum setting
of the pvT molding path (e.g., the use of scientific molding
methods [8–11]) assists in obtaining optimal product quality.

Statistics-based mathematical models can describe the
relationship between process parameters and part quality.
Among them, the Taguchi experimental method combined
with analysis of variance or correlation coefficient is widely
used to identify the ideal combination of injection molding
parameters. This method can be used to reduce the number
of experiments and determine the factors that reduce manu-
facturing costs and achieve stable quality goals [12–16].
The emergence of artificial intelligence has also offered highly
nonlinear fitting possibilities. Through the establishment
and testing of different training models, it can be effectively
applied to various scenarios. Artificial neural networks
(ANNs) enable users to rapidly create artificial networks and
quality prediction solutions through appropriate hyperpa-
rameter adjustments. These adjustments link a large amount
of process parameter information with resultant molding
quality [17–21]. ANN technology can be employed in the
learning process of different data types from large volumes
of data to achieve clustering, classification, prediction, and
regression functions. In particular, ANNs are suitable for
modeling tasks involving nonlinear relationships, such as
thermoplastic injection molding processes with complex
viscoelastic material behavior and nonlinear relationships
between quality, process, and machine parameters. Hyperpa-
rameters, including the hidden layer architecture, activation
functions, optimization solver, learning rate, and momen-
tum, play key roles in model learning. Common optimization
solvers include the stochastic gradient descent (SGD) [22,
23], stochastic gradient descent with momentum (SGDM)
[24, 25], and Adam [26, 27]. Learning rate and momentum
are both influential in the quality and speed of model learn-
ing but are rarely studied in the literature.

To explore the influence of hyperparameters in the actual
training process on model training speed, this research
examined the injection molding of integrated circuit (IC)
trays and extracted the pressure curve indicating the quality
of the part into the quality index. Through application of
the index and width of the part to the input and output lay-
ers of the neural model, respectively, the two optimization
solvers SGD and SGDM were explored. Among them, the
activation function, learning rate, and momentum varied,
and their influence on the accuracy, convergence speed, and
oscillation of the training model was analyzed to provide a
reference for adjustment.

## 2 Methods

### 2.1 Multilayer perceptron model

The multilayer perceptron (MLP) model is a supervised
ANN learning model with a forward propagation structure
that maps a set of input vectors to a set of output vectors.
An MLP can be regarded as a directed graph composed of
multiple node layers, with each layer fully connected to the
next layer. The MLP model contains three layers, namely
the input, hidden, and output layers. Except for the input
node, each node is a neuron with a summation function and
activation function. The activation function, expressed in
Eq. (1), is a nonlinear function used to map the summation
function \(xw + b\) to the output value \(y\). The terms \(x, w, b,\)
and \(y\) represent the input vector, weighting vector, bias, and
output value, respectively.

\[
y = \varphi(xw + b) \tag{1}
\]

The weighting values range between 0 and 1. These val-
ues change with the training data and represent the memory
of the neural network related to the input and output model
training.

#### 2.1.1 SGDM

SGDM is an optimized solution algorithm typically
employed in MLP model training. As described in Eqs. (2)
and (3), the SGD optimization algorithm, also known as
the steepest descent method, generates function solutions
in reference to the opposite direction of the gradient and
step distance (or learning rate, \(\alpha\)) to iteratively search the
weighting value \(w\). Although the weights can be updated
iteratively using the SGD algorithm, if multiple local minimums are present in the function, the local minimum or saddle point can be searched, but the global minimum cannot be obtained. This consequently halts the training iterations and leads to erroneous learning results.

The SGDM algorithm that combines SGD and momentum (β) adjustment has attracted considerable attention. The SGDM algorithm detailed in Eqs. (4) and (5) takes β into account, and thus the optimization algorithm can jump out of the local minimum during the model training process. This procedure enables the entire iteration to stably converge to the minimum value of the loss function.

\[ w_{t+1} = w_t - \alpha \times g_t \quad (2) \]

\[ g_t = -\frac{\partial \text{Loss}}{\partial w_t} \quad (3) \]

\[ w_{t+1} = w_t - \alpha \times m_t \quad (4) \]

\[ m_t = \beta \times m_{t-1} + (1 - \beta) \times g_t \quad (5) \]

**Learning rate** The learning rate (α) determines the iteration speed of weight adjustment in the neural network according to the gradient loss function. That is, the smaller the learning rate is, the slower the decline along the loss gradient is. A lower learning rate can prevent potentially optimal values from being overlooked, thus obtaining higher training accuracy, but this process requires a longer convergence time. Generally, the setting of the learning rate depends on experience, model size, and numerical complexity. Even for the same training model, when faced with input values of diverse dimensions, the convergence of the optimal learning rate must also change. Therefore, the learning rate must be adjusted for various data conditions. Depicts the optimization status for various learning rates. A low learning rate has a slow convergence speed but ensures that the minimum is identified at each step of training to obtain optimal training accuracy. By contrast, a high learning rate can accelerate the convergence speed but may fix on a suboptimal solution (Fig 1).

**Momentum** The conventional SGD algorithm calculates the steepest descent gradient to rapidly search for the minimum value of the objective function. However, such a simplistic strategy cannot prevent the failure of local minimums or saddle points, which results in false stops during the iterative search process. The introduction of momentum (β) in the SGD algorithm can overcome these shortcomings. Figure 2 illustrates the path of searching the minimum value, where the momentum represents the magnitude of the next movement, with the direction based on the current position.
Activation function The activation function, described in Eq. (1), in a neural network model transforms the inputs into outputs within a certain range and is added to the ANN to assist the network in learning complex patterns in the data. Specifically, these functions introduce nonlinear physical systems to ANNs. Without an activation function, a neural network acts as a linear regression with limited learning ability. To efficiently and accurately learn the nonlinear state of the quality of the injection-molded parts in correspondence to the process parameters, a suitable activation function is necessary. The derivatives also play vital roles in the weight update. Various activation functions perform this task differently. Figure 3 presents the various activation functions used in the ANN model, and Eq. (6) describes a commonly used Sigmoid activation function that converts the real value into the range of 0 to 1. When the input number is large, the result is close to 1, and when entering a small negative number, the result is close to 0. This function can accurately indicate whether the neuron is stimulated, with 0 and 1 referring to low and full activation, respectively. The Sigmoid function acts as the last layer of the machine learning model to convert the output into a probability score, which can be simpler to use and interpret. If the inputs are relatively large

Fig. 3 Activation functions: a Sigmoid, b Tanh, c ReLu, d Leaky ReLU, and e ELU
or small, the Sigmoid function induces a saturation effect in which the neuron resembles a dead state.

\[ f(x) = \frac{1}{1 + e^{-x}} \]  

(6)

The Tanh activation function detailed in Eq. (7) and Fig. 3b compresses the input value within a range of \([-1, 1\]. This function is similar to the Sigmoid function; thus, when the number of inputs is relatively large or small, the neuron resembles a dead state.

\[ f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \]  

(7)

When using the Sigmoid or Tanh activation functions, the gradient decreases as the number of layers increases. This reduces the gradient value of the initial layer, and these layers thus cannot be correctly learned; when the depth of the network moves the value to 0, their gradient tends to vanish, which is known as the vanishing gradient problem [28].

The rectified linear unit (ReLU) presented in Fig. 3c and Eq. (8) is characterized by the value of the first derivative located at 0 and 1. Compared with the Sigmoid and Tanh activation functions, the gradient of the ReLU is simple, does not face saturation, and converges more rapidly. However, when a large error gradient accumulates and leads to a large update of the neural network weights, an exploding gradient occurs, resulting in unstable learning. Moreover, a dying ReLU can become problematic when the gradient of all negative input values is 0; if a ReLU neuron is stuck on the negative side and constantly outputs 0, it is considered “dead.” Because the slope of the ReLU in the negative range is also 0, once the neuron becomes negative, it is unlikely to recover. Such neurons cannot play any role in distinguishing inputs and are essentially useless.

\[ f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \]  

(8)

The Leaky ReLU activation function, described in Eq. (9) and Fig. 3d, represents an improved version of the ReLU. This function primarily prevents the disappearance of the gradient generated when \( x \) is less than 0. When a neuron is in an inactive state, it allows a nonzero gradient, where \( C_1 \) is a small positive constant.

\[ f(x) = \begin{cases} C_1 x, & x < 0 \\ x, & x \geq 0 \end{cases} \]  

(9)

The exponential linear unit (ELU) activation function presented in Eq. (10) and Fig. 3e contains an adjustable positive constant \( C_2 \) to avoid the possible saturation of the ELU.

\[ f(x) = \begin{cases} x, & x < 0 \\ C_2(e^x - 1), & x \geq 0 \end{cases} \]  

(10)

### 2.2 Quality indices

To predict the quality of parts in terms of molding condition changes and reduce the amount of calculation data in the experiment, this study combined domain knowledge and data mining technology to convert the injection molding data into quality indices. The correlation of these indices with the quality of the finished product was then calculated. Those that correlated highly were selected as input parameters to improve the prediction accuracy and effectively reduce the amount of calculation required during model training. The following four main quality indices were introduced in this study [18, 19]:

1. **First-stage holding pressure index (\( Ph_{index} \))**: This represents the average holding pressure in the first stage, as expressed in Eq. (11). The function of the holding pressure is to compensate for the volumetric shrinkage of the part caused by the cooling of the polymer melt. The holding pressure affects the geometric dimensions of injection-molded parts and is crucial to precision injection molding.

\[ Ph_{index} = \frac{1}{t_1 - t_0} \int_{t_0}^{t_1} g \, dt \]  

(11)

2. **Peak pressure index (\( Pp_{index} \))**: This represents the maximum pressure during filling and compression, as described in Eq. (12). Injection molding constitutes a series of pressure-driven (\( f \)) melt flow processes. The pressure not only determines the speed of the melt flow and its flow inertia but also affects the quality of the melt flowing into the mold cavity. Therefore, the maximum pressure index affects the quality and geometry of injection-molded parts.

\[ Pp_{index} = \text{Max}(f) \]  

(12)

3. **Residual pressure drop index (\( Pr_{index} \))**: This indicates the average residual pressure drop in the cavity during cooling, as detailed in Eq. (13). The average residual pressure drop is related to the residual stress of the polymer. A residual pressure that is excessively high or low may cause geometric deformation or size shrinkage of the injection-molded parts, respectively.

\[ Pr_{index} = \frac{1}{t_3 - t_2} \int_{t_2}^{t_3} f \, dt \]  

(13)

4. **Pressure integral index (\( PI_{index} \))**: As expressed in Eq. (14), this indicates the integral value of the pressure curve with respect to time during the molding cycle, including the pressure changes during the melt filling process. This index is related to the total pressure
characteristics of the polymer melt during the molding process. Variation in the index can reflect changes in part quality, particularly weight-related changes.

\[ P_{I_{index}} = \int_{0}^{t_1} f dt \]  

(14)

These four indices have different ranges of values and are normalized between 0 and 1. Using these indices as inputs for model training can support rapid convergence and high accuracy.

3 Experimental setup

3.1 Mold, machine, materials, and measurements

In this experiment, an IC tray of \( 76 \times 76 \times 4.4 \) \( \text{mm}^3 \) and flow length-to-thickness ratio of 124 was fabricated as the research carrier for method verification (Fig. 4). The experimental operation was conducted on an all-electric injection machine (CT100; Fu-Chun Shin, Tainan, Taiwan) with a maximum clamping force of 100 tons. The processed material was acrylonitrile butadiene styrene (PA-756; Chi-Mei, Tainan, Taiwan). To collect the physical information on the polymer melt flow behavior within the mold cavity, two types of in-mold cavity sensors were installed (SSB-01KN08X06H and SSB01KN08X08H; Futaba, Mobara, Japan).

Figure 4 illustrates the locations of each sensor. To study the overall flow state of the melt and its response to the quality indices, seven pressure sensors were installed in the mold, with one at the sprue, one in front of the gate, one near the gate, two at the center of the cavity, and two far from the gate. The polymer melt was assumed to advance in laminar flow because of its typical fountain flow behavior; the ratio of the flow direction distance to the average thickness was also high (124), and pressure change along the thickness direction was ignored.

Table 1 lists the process parameter settings in the injection molding experiment. For this experiment, a two-factor full factorial method was adopted, in which the injection speed range was adjusted from 40 to 120 mm/s and the first-stage holding pressure varied from 50 to 100 MPa; data were collected four to eight times with the same parameters. Table 2

![](Fig. 4 Positions of the pressure sensors installed in the cavity [18])

| Item                                  | Unit | Parameters                  |
|----------------------------------------|------|------------------------------|
| Melt temperature                       | °C   | 205                          |
| Mold temperature                       | °C   | 60                           |
| Backpressure                           | MPa  | 4.5                          |
| Clamping force                         | Tons | 70                           |
| Decompression on stroke                | mm   | 10                           |
| Holding speed limit                    | mm/s | 80                           |
| V–P switchover position                | mm   | 12.45                        |
| Cooling time                           | s    | 16                           |
| Holding pressure 1st stage             | MPa  | 40, 50, 60, 70, 80, 90, 100  |
| Holding pressure 2nd stage             | MPa  | 5                            |
| Holding pressure 3rd stage             | MPa  | 15                           |
| Holding time 1st stage                 | s    | 1                            |
| Holding time 2nd stage                 | s    | 4                            |
| Holding time 3rd stage                 | s    | 5                            |
| Injection speed 1st stage              | mm/s | 40, 50, 60, 70, 80, 90, 100  |
| Injection speed 2nd stage              | mm/s | 5                            |
| Injection speed 3rd stage              | mm/s | 5                            |

Table 1 Molding parameter setting
lists the sensors used to extract quality indices, which records the system pressure curve and seven cavity pressure curves (SN1–SN7) for each shot. The system pressure curve was used to obtain two quality indices, namely the $P_h$ and $P_I$ indices. The seven cavity pressure curves (SN1–SN7) were used to obtain the $P_p$ index, and four curves (SN4–SN7) produced by sensors installed far from the gate were used to obtain the $P_r$ index. Figure 5a displays the physical meaning of the peak pressure index ($P_p$ index) and residual pressure drop index ($P_r$ index) in the cavity pressure profiles measured by sensors 1–7. The system pressure provides the driving force for the polymer melt to overcome the resistance during mold filling and compression. The parameter $P_r$ index represents the average pressure drop from sensors 4–7 to sensor 3 in the cooling stage, and this pressure drop indicated local shrinkage. Figure 5b shows the physical meaning of the first-stage holding pressure index ($P_h$ index) and system pressure integral index ($P_I$ index). The parameter $P_I$ index represents the momentum required for mold filling and compression, which indicates the quality of the injection-molded parts. The parameter $P_h$ index represents the main packing capacity in the holding stage, which is related to the molding weight and part geometry. In this study, a total of 445 subexperiments were conducted, each comprising 11 quality indices. These quality indices were used for the input data of the MLP model.

Usually, the weight of the part is used as a good indicator that can quickly provide a result of process stability. However, even if the weight of the parts is the same, the influence of the holding conditions may cause dimensional changes. In this study, rather than the weight of the part, the quality is represented by the width of the IC tray geometry. Figure 6 depicts the width measurement position, measured using a three-coordinate measurement machine (CRYSTA–Apex S700; Mitutoyo, Kawasaki, Japan). To classify the quality of injection-molded parts, we converted the measured geometrical width into multiple grades. We subsequently aggregated the data into 5 grades evenly spaced between the minimum and maximum values in width, which were used as the output data of the MLP model. Notably, the number of grades typically determines the precision of the predicted width. Consider when a $z$ value of 1.78 was used for outlier filtering, the maximum and minimum of experimental datapoints were 76.036 and 75.869 mm, respectively. The bandwidth of each grade was defined as the width range (i.e., the maximum width minus the minimum width) divided by the number of grades, which was 33.4 μm when the number of grades was 5. That is the precision of the prediction model for the width is 33.4 μm. Accordingly, the width ranges were divided into 5 grades, as displayed in Fig. 7.

### 3.2 Experimental procedure

A flowchart of the entire process operation is presented in Fig. 8, including data preprocessing, hyperparameter...
Fig. 6 IC tray and its width quality measurement location

Fig. 7 Classification of width
adjustment, and data training. A detailed description is outlined as follows:

1. Pressure signal extraction and quality measurement: The pressure curve of the molding process was obtained through the in-mold sensors, and a three-dimensional measuring instrument was used to measure its width value, thereby establishing a pressure history and width value database.

2. Outlier filtering: The standard score was used as the basis for judgment, with the z value of 1.78 selected as the standard to eliminate outlier values under the same molding parameters. Among them, the 1.78 standard score is equivalent to retaining 92.5% of the data volume [19].

3. Quality indices transformation: Based on Eqs. (11–14), the extracted in-mold pressure curve was converted into 11 quality indices (Table 2) for subsequent machine learning.

4. Data normalization: The size and dimension of the input values have distinct effects on data convergence. Processing prior to data normalization reduces the adverse effects of the different dimension values on the convergence results [29]. Therefore, max–min normalization was used in this experiment.

5. MLP model training: This experiment uses the SGD and SGDM as optimizers. These were applied to a fixed-architecture MLP for data training. The nodes of the input, first hidden, second hidden, and output layers were 11, 50, 25, and 5, respectively.

6. Hyperparameter adjustment: This experiment focused on the changes of five different activation functions, momentum, and learning rate to assess the results of learning accuracy and convergence behavior. The range of momentum was 0.1 to 0.9, and the range of learning rate was $10^{-5}$ to $10^{-1}$ and 0.1 to 0.9. A full factorial experimental design was employed in this study.

7. Training and testing results: The effect of momentum and learning rate on accuracy was assessed by recording the training and verification learning accuracy and efficiency, respectively.

8. Performance index calculation: To generate statistical trends and enhance physical interpretability, five trainings were conducted under the same hyperparameter settings; the learning rate and efficiency were recorded and the average calculated for subsequent evaluation.

### 3.3 MLP model and hyperparameter settings

Table 3 describes the MLP model used in this experiment and the corresponding hyperparameter settings. The number of nodes in the input layer, two hidden layers, and output layer of the MLP model was 11, 50/25, and 5, respectively. The number of training and testing datasets was 337 and 84, respectively, and the two optimized solvers SGD and SGDM were used for training with 7200 iterations. Both solvers set the learning rate value, and the SGDM adjusted the momentum value. The adjustment range of momentum was 0 to 0.9. They are classified as two groups, group 1 and group 2, to explore the influence of learning rate dimension on training accuracy. The training hardware specifications used in this experiment are detailed in Table 4.
4 Results and discussion

To explore the difference between conventional SGD and SGDM, the model was trained through adjustment of the activation function, learning rate ($\alpha$), and momentum ($\beta$) to capture its training and test accuracy. The effects of five activation functions (Sigmoid, Tanh, ReLU, Leaky ReLU, and ELU) on the learning rate were also analyzed. The $\alpha$ was divided into group 1, ranging from $10^{-5}$ to $10^{-1}$, and group 2, ranging from 0.1 to 0.9, for experimental adjustment. Finally, the $\beta$ was set in a range of 0.1 to 0.9.

4.1 Group 1 with zero momentum

The goal was to compare the effect of learning rate adjustment on the training accuracy of the MLP by using five activation functions in the SGD. The training rate set was evaluated for various activation functions following 7200 iterations. Figure 9 a and b illustrate the training and testing accuracy when the learning rate was set to $10^{-5}$ and $10^{-4}$, respectively, both of which proved inefficient (14–65%) and are not recommended for training MLP models. However, when the learning rate was set to $10^{-3}$, the ReLU and ELU activation functions exhibited high training and testing accuracy (over 90%) and rapid convergence. When the learning rate was increased to $10^{-1}$, the accuracy of all activation functions reached over 90%. Among them, the Tanh and ELU functions had the most rapid convergence speed at a learning rate of $10^{-1}$, although the accuracy of these activation functions exceeded 90% in only 151 iterations (Table 5). At a low learning rate of $10^{-3}$, the ELU and ReLU reached a learning accuracy of over 90% in 6193 and 6332 iterations, respectively, indicating effective convergence in model training. A learning rate of $10^{-1}$ provided effective training and convergence results in relation to learning accuracy (Table 6). For the Sigmoid function, the highest training accuracy was observed at a learning rate of $10^{-1}$, but 1177 iterations were required before its training accuracy exceeded 90%, which was the slowest rate of increase of all activation functions.

4.2 Group 2 with zero momentum

Different activation functions require an appropriate learning rate to achieve high learning accuracy. A learning rate of $10^{-1}$ enabled all five activation functions to obtain the optimal model learning performance. The learning rate was thus increased beyond $10^{-1}$ to investigate its performance in terms of accuracy, convergence, and stability.

Table 3 MLP model and hyperparameter settings

| Hyperparameter       | Values                                      |
|----------------------|---------------------------------------------|
| Number neuron of     |                                             |
| input layer          | 11                                          |
| 1st hidden layers    | 50                                          |
| 2nd hidden layers    | 25                                          |
| output layer         | 5                                           |
| Mini batch size      | 20                                          |
| Learning rate ($\alpha$) | Group 1: $10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}$, Group 2: $0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9$ |
| Momentum ($\beta$)   | 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 |
| Iteration            | 7200                                        |
| Activation function  | Sigmoid, ReLU, Tanh, Leaky ReLU, ELU        |
| No. training dataset | 337                                         |
| No. testing dataset  | 84                                          |

Table 4 Hardware specifications

| Hardware               | Specification       |
|------------------------|---------------------|
| Central processing unit| AMD R9 3900×        |
| Graphic processing unit| GIGABITE GeForce RTX3090 |
| Memory                 | 32 GB               |
| Training software      | MATLAB r2020b       |
In addition to the stability and quality of the learning accuracy, learning efficiency is critical to reducing computation time. Figure 11 and Table 7 describe the number of iterations required for group 2 with zero momentum to exceed a training rate of 90%. To further analyze the relationship between the learning rate and number of iterations required to exceed 90%, the correlation coefficients are listed in Table 8. As described in Fig. 11, the activation functions Tanh, ReLU, Leaky ReLU, and ELU and number of iterations required to exceed a 90% learning rate were positively correlated with the learning rate, with correlation coefficients of 0.82, 0.74, 0.84, and 0.88, respectively. These strong positive correlations indicated that an increase in the learning rate may slow convergence speed, thus requiring more iterations. Moreover, even at the highest learning rate, the ReLU and ELU functions could not achieve 90% training accuracy. Conversely, the Sigmoid function had a strong negative correlation coefficient (−0.84), demonstrating that effective convergence can be obtained with an increased learning rate. Notably, with a
learning rate of 0.8, only 271 iterations were required to obtain a learning accuracy of over 90%.

Although a high learning rate generally promotes rapid convergence, the stability of the combined learning rate and activation function must be monitored in model learning.

This study evaluated the stability through calculation of training accuracy deviation when the learning rate exceeded 90% in model training. Figure 12 illustrates that a learning rate in the range of 0.1 to 0.5 resulted in a training accuracy deviation of approximately 7% for group 2 with zero momentum.
momentum. By contrast, when the learning rate was in the range of 0.6 to 0.9, the training accuracy deviation of the ReLU and ELU functions increased considerably, indicating model instability and a high correlation with the learning rate. The Sigmoid activation function maintained a deviation of less than 5% under various learning rates, demonstrating a low oscillation phenomenon at a high learning rate and the ability to effectively maintain convergence (Table 9).

### 4.3 Effect of momentum on convergence

This experiment employed momentum acceleration in the SGDM method to explore the influence of momentum on modeling accuracy. Figure 13 and Table 10 present the testing accuracy of the SGDM after 7200 iterations at a learning rate of 0.1 ($\alpha = 0.1$). The introduction of momentum did not substantially improve the testing accuracy of the

| Activation Function | Learning rate |
|---------------------|---------------|
|                     | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | Avg |
| Sigmoid             | 3.9 | 2.9 | 3.9 | 4.1 | 2.7 | 3.3 | 4.9 | 4.7 | 4.8 | 3.9 |
| Tanh                | 4.6 | 5.0 | 5.3 | 5.7 | 7.0 | 8.9 | 6.6 | 5.7 | 10.6| 6.6 |
| ReLU                | 3.1 | 4.1 | 4.8 | 3.5 | 6.8 | 16.8| 13.7| -   | -   | 7.6 |
| Leaky ReLU          | 5.0 | 5.6 | 5.2 | 4.7 | 5.6 | 9.2 | 6.9 | 6.8 | 6.4 | 6.2 |
| ELU                 | 4.9 | 5.0 | 6.7 | 5.2 | 5.5 | 7.9 | 17.9| 61.8| -   | 14.4|

Unit of accuracy: %

- : less than 90% at 7200 iterations
five activation functions; the testing accuracy of the ELU and ReLU functions actually decreased with the addition of momentum. Table 11 lists the correlation coefficients of momentum and the number of iterations required to achieve 90% testing accuracy. The correlation coefficients of the Sigmoid, Tanh, ReLU, Leaky ReLU, and ELU functions were $-0.14$, $-0.3$, $0.62$, $0.23$, and $0.41$, respectively, indicating that momentum was not strongly correlated with the number of iterations.

Figure 14 and Table 12 detail the number of iterations required for model learning to achieve a 90% training rate, and Table 13 lists the correlation coefficients of the number of iterations and momentum. The Sigmoid function, with a correlation coefficient of $-0.99$, was the most effective in relation to changes in momentum. For momentum change from 0.1 to 0.9, the number of iterations required to reach a 90% training rate was markedly reduced from 1193 to 210 (improvement rate: 82.4%). The Tanh function had a strong negative correlation with momentum (correlation coefficient: $-0.79$), and the number of iterations required to achieve a 90% training rate was significantly reduced from 151 to 98 (improvement rate: 35.1%). By contrast, the ReLU, Leaky ReLU, and ELU functions exhibited minimal correlation with momentum; thus, momentum changes did not affect the learning efficiency.

### 4.4 Hyperparameters optimization

This analysis revealed that hyperparameter settings affect the accuracy and efficiency of model training. Through optimization of hyperparameter settings, model performance can be improved. Table 14 presents the two sets of hyperparameter settings, which were initially based on experience, with the later optimization process based on the recommendations in Sects. 4.1 to 4.3. The testing accuracy of the initial and optimized settings was 87.4% and 97.2%, respectively. Therefore, adjustment of the learning rate and momentum can vastly improve the accuracy of model testing (up to 9.8% improvement rate). In regard to the training performance efficiency, the number of iterations required for the initial and optimized settings to achieve 80% training accuracy was

---

**Table 10** Testing accuracy of the SGDM ($\alpha=0.1$) under 7200 iterations

| Activation Function | Momentum | 0.0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---------------------|----------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| Sigmoid             | 95.8     | 95.8| 93.9| 94.4| 95.1| 94.1| 93.4| 91.8| 94.4| 97.2|
| Tanh                | 93.2     | 92.7| 95.5| 95.1| 92.7| 93.4| 93.9| 92.9| 92.7|
| ReLU                | 95.3     | 94.8| 93.7| 92.7| 93.4| 94.1| 93.7| 92.5| 94.6| 88.2|
| Leaky ReLU          | 93.4     | 93.2| 92.5| 92.7| 90.1| 93.9| 93.9| 92.2| 96.0| 96.7|
| ELU                 | 94.6     | 86.8| 89.4| 91.1| 92.2| 92.5| 91.3| 92.0| 92.9| 93.9|

Unit of accuracy: %
The experimental results demonstrated that optimizing hyperparameter settings not only improves the accuracy of model testing but also considerably reduces training time. Overall, the experimental results demonstrated that optimizing hyperparameter settings not only improves the accuracy of model testing but also considerably reduces training time.

5680 and 210, respectively; therefore, the optimized hyperparameter settings markedly reduced training time. Overall, the experimental results demonstrated that optimizing hyperparameter settings not only improves the accuracy of model testing but also considerably reduces training time.

### Table 12

Number of iterations required for the SGDM for group 2 with momentum to achieve 90% training accuracy ($\alpha = 0.1$)

| Activation Function | Momentum | 0.0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---------------------|----------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| Sigmoid             |          | 1177| 1193| 1027| 905 | 809 | 675 | 516 | 423 | 291 | 210 |
| Tanh                |          | 151 | 133 | 117 | 126 | 119 | 119 | 101 | 73  | 114 | 98  |
| ReLU                |          | 417 | 214 | 229 | 219 | 148 | 342 | 196 | 353 | 177 | 225 |
| Leaky ReLU          |          | 239 | 238 | 226 | 250 | 322 | 192 | 212 | 241 | 269 | 295 |
| ELU                 |          | 151 | 129 | 208 | 146 | 122 | 106 | 135 | 213 | 127 | 448 |

### Table 13

Correlation coefficients of the number of iterations and momentum ($\alpha = 0.1$)

| Activation function | Sigmoid | Tanh | ReLU | Leaky ReLU | ELU | Pearson’s correlation coefficient |
|---------------------|---------|------|------|------------|-----|----------------------------------|
|                     | −0.99   | −0.79| −0.26| 0.26       | 0.48|                                  |

### Table 14

Comparison of various hyperparameter settings

| Hyperparameter         | Initial | Optimized |
|------------------------|---------|-----------|
| Learning rate          | 0.01    | 0.1       |
| Momentum               | 0.1     | 0.9       |
| Max. iteration         | 7200    | 7200      |
| Activation function    | Sigmoid | Sigmoid   |
| Testing accuracy       | 87.4%   | 97.2%     |
| Iterations to reach 80% training accuracy | 5,680 | 210 |

5680 and 210, respectively; therefore, the optimized hyperparameter settings markedly reduced training time. Overall, the experimental results demonstrated that optimizing hyperparameter settings not only improves the accuracy of model testing but also considerably reduces training time.

### 5 Conclusion

This study examined the influence of hyperparameters on the accuracy and convergence of MLP model training. In this investigation, IC tray injection molding cavity pressure curves were measured using sensors and converted to normalized quality indices to serve as the input data for model training; the part size was used as the output data. The MLP architecture comprised 11 nodes in the input layer, 75 and 50 nodes in the first and second hidden layer, respectively, and 5 nodes in the output layer. In addition, this study provided a comparison of the SGD and SGDM optimizers, a comparison of five activation functions (Sigmoid, ReLU, Tanh, Leaky ReLU, and ELU), and an evaluation of learning rate and momentum adjustment and its effects on the accuracy and efficiency of model training. The results are summarized as follows:

- Regarding the change of learning rate in the SGD algorithm without momentum, functions with a learning rate of 0.1 performed most effectively, with the learning rate exceeding 90%. The Tanh and ELU excitation functions combined with the SGD can obtain rapid convergence.
- With the introduction of momentum ranging from 0.1 to 0.9, the Sigmoid function achieved an average...
training accuracy of 93.8%, outperforming the other five functions. In addition, its strong correlation with momentum was reflected in the rapid convergence rate.

- Regarding the stability of the activation function, when the learning rate was between 0.1 and 0.5, the training accuracy ranges of the five functions were all within 7%. When the learning rate was between 0.6 and 0.9, only the training accuracy of the Sigmoid activation function remained below 5%, indicating its stability in MLP model training.

- This case study demonstrated the effectiveness of hyperparameter settings in MLP modeling. Under the proper hyperparameter settings (in this case, a learning rate of 0.1 and momentum of 0.9), the Sigmoid function performs effectively in terms of training accuracy and efficiency.

- Results from the two sets of hyperparameter settings, the initial set based on experience and the optimized set based on the recommendations in Sects. 4.1 to 4.3, demonstrated a testing accuracy of 87.4% and 97.2%, respectively. In regard to the training efficiency performance, the number of iterations required for the initial and optimized settings to achieve 80% training accuracy was 5680 and 210, respectively. Therefore, optimization of the hyperparameter settings improved the accuracy of model testing and reduced training time.

This research has shown the significant impact of hyperparameter settings on the accuracy and efficiency model testing through an actual case study. Establishing a fast and systematic method to find the best hyperparameter combination suitable for various injection molding applications is considered in the future work.

Acknowledgements This research was supported in part by the Featured Areas Research Center Program within the framework of the Higher Education Sprout Project by the Ministry of Education (MOE), Taiwan.

Author contribution K.-C. Ke and M.-S. Huang were responsible for deriving formulas. K.-C. Ke was responsible for simulation. K.-C. Ke and M.-S. Huang were involved in the discussion and significantly contributed to making the final draft of the article. All the authors read and approved the final manuscript.

Data availability The authors confirm that the data supporting the findings of this study are available within the article.

Declarations

Consent to participate Not applicable.

Consent to publish Not applicable.

Competing interests The authors declare no competing interests.

Ethical approval Not applicable.

References

1. Huang MS, Ke KC, Liu CY (2021) Cavity-pressure-based holding pressure adjustment for enhancing the consistency of injection molding quality. J Appl Polym Sci 138(50357):1–10
2. Zhang J, Zhao P, Zhao Y, Huang J, Xia N, Fu J (2019) Online measurement of cavity pressure during injection molding via ultrasonic investigation of tie bar. Sens Actuators Phys 285:118–126
3. Lin CC, Wang WT, Kuo CC, Wu CL (2014) Experimental and theoretical study of melt viscosity in injection process. Int J Mech Meca Eng 8:1–5
4. Wang J, Peng J, Yang W (2011) Filling-to-packaging switchover mode based on cavity temperature for injection molding. Polym-Plast Technol Eng 50:1273–1280
5. Zhao P, Xia N, Zhang J, Xie J, Zhang C, Fu J (2020) Measurement of molecular orientation using longitudinal ultrasound and its first application in in-situ characterization. Polymer 187(122092):1–11
6. Farahani S, Brown N, Loftis J, Krick C, Pichl F, Vaculik R, Pilli S (2019) Evaluation of in-mold sensors and machine data towards enhancing product quality and process monitoring via Industry 4.0. Int J Adv Manuf Technol 105:1371–1389
7. Loftis J, Farahani S, Pilli S (2020) Online quality monitoring of plastic parts using real-time data from an injection molding machine. Inter Manuf Sci Eng Conf, ASME 84256:V001T02A001
8. Chang YH, Wei TH, Chen SC, Lou YF (2020) The investigation on PVT control method establishment for scientific injection molding parameter setting and its quality control. Polym Eng Sci 60:2895–2907
9. Wang J (2012) PVT properties of polymers for injection molding. Some Critical Issues for Injection Molding 1–30
10. Hopmann C, Kahve C, Schmitz M (2020) Development of a novel control strategy for a highly segmented injection mold tempering for inline part warpage control. Polym Eng Sci 60:2428–2438
11. Wang J, Mao Q (2013) A novel process control methodology based on the PVT behavior of polymer for injection molding. Adv Polym Technol 32:E474–E485
12. Kamaruddin S, Khan ZA, Foong SH (2010) Application of Taguchi method in the optimization of injection moulding parameters for manufacturing products from plastic blend. Int J Eng Technol 2:574–580
13. Yizont T, Ariff ZM, Khalil AM (2017) Influence of processing parameters on injection molded polystyrene using Taguchi method as design of experiment. Procedia Eng 184:350–359
14. Kiatcharoenpol T, Vichirapraset T (2018) Optimizing and modeling for plastic injection molding process using Taguchi method. J Phys Conf Ser 1026:012018
15. Wang Q, Yang C, Du K, Wu Z (2019) Effect of micro injection molding parameters on cavity pressure and temperature assisted by Taguchi method. Mechanika 25:261–268
16. Feng Q, Liu L, Zhou X (2020) Automated multi-objective optimization for thin-walled plastic products using Taguchi, ANOVA, and hybrid ANN-MOGA. Int J Adv Manuf Technol 106:559–575
17. Lockner Y, Hopmann C (2021) Induced network-based transfer learning in injection molding for process modelling and optimisation with artificial neural networks. Int J Adv Manuf Technol 112:3501–3513
18. Ke KC, Huang MS (2020) Quality prediction for injection molding by using a multilayer perceptron neural network. Polymers 12:1812
19. Ke KC, Huang MS (2021) Quality classification of injection-molded components by using quality indices, grading, and machine learning. Polymers 13:353
20. Hwang S, Kim J (2019) Injection mold design of reverse engineering using injection molding analysis and machine learning. J Mech Sci Technol 33:3803–3812
21. Ogorodnyk O, Lyngstad OV, Larsen M, Wang K, Martinsen K (2019) Application of machine learning methods for prediction of parts quality in thermoplastics injection molding. Advanced Manufacturing and Automation VIII. Springer, Singapore, pp 237–244
22. Lei Y, Tang K (2021) Learning rates for stochastic gradient descent with nonconvex objectives. IEEE Trans Pattern Ana Mach Intell. https://doi.org/10.1109/TPAMI.2021.3068154
23. Cheridito P, Jentzen A, Rossmannek F (2021) Non-convergence of stochastic gradient descent in the training of deep neural networks. J Complex 64:101540
24. Jin R, He X (2020) Convergence of momentum-based stochastic gradient descent. 16th IEEE Int Conf Control Automation, Sapporo, Hokkaido, Japan, pp 779–784
25. Sharma A (2018) Guided stochastic gradient descent algorithm for inconsistent datasets. Appl Soft Comput J 73:1068–1080
26. Gupta P, Garg S (2019) Breast cancer prediction using varying parameters of machine learning models. 3rd Int Conf Computing Network Communications, Trivandrum, Kerala, India, pp 593–601
27. Bock S, Weis M (2019) A proof of local convergence for the Adam optimizer. Int Joint Conf Neural Networks, Institute of Electrical and Electronics Engineers Inc., Budapest, Hungary
28. Hochreiter S (1998) The vanishing gradient problem during learning recurrent neural nets and problem solutions”. Int J Uncertain Fuzziness Knowledge-Based Syst 6:107–116
29. Björck J, Gomes C, Selman B, Weinberger KQ (2018) Understanding batch normalization. ArXiv180602375 Cs Stat

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.