The Seventeenth International Symposium on Artificial Life and Robotics 2012 (AROB 17th '12),
B-Con Plaza, Beppu, Oita, Japan, January 19-21, 2012

Optimizing the Thermoforming of Polypropylene Foam by an Artificial Neural Network

Shih-Jung Liu*, Ting-Ting Wen, and Yau-Zen Chang

Department of Mechanical Engineering
Chang Gung University, Tao-Yuan 333, Taiwan
(Tel: 886-3-2118166, Fax: 886-3-2118558)

*shihjung@mail.cgu.edu.tw

Abstract: In this study, the optimal processing parameters of polypropylene foam thermoforming are obtained by the use of an artificial neural network. Data from tests carried out on a lab-scale thermoforming machine were used to train an artificial neural network, which serves as an inverse model of the process. The inverse model has the desired product dimensions as inputs and the corresponding processing parameters as outputs. The structure, together with the training methods, of the artificial neural network is also investigated. The feasibility of the proposed method is demonstrated by experimental manufacturing of cups with optimal geometry derived from the finite element method. Except the dimension deviation at one location, which amounts to 17.14 %, deviations of the other locations are all below 3.5 %.

Keywords: Polypropylene Foam, Thermoforming, Artificial Neural Network, Optimal Processing Parameters.

1 INTRODUCTION

Thermoforming has become one of the most important polymer processing methods [1]. However, persistent problems in the selection of processing parameters due to the complex interconnected nature of the involved thermal, chemical, friction and visco-elastic phenomena have confounded the overall success of the process. One of the most serious problems is the lack of an adequate method to derive processing parameters with which prescribed product geometry can be obtained. The parameter-selection problem is of great concern, because it is usually linked with high cost and long start-up time. In practice, the trial-and-error method is relied upon with repeated processing runs to achieve required product geometry. This restriction renders uneconomical a small-batch type manufacturing with accurate product shape.

In this work, an effort was made to develop a technique based on ANN to find a set of thermoforming parameters that will produce foamed plastic products in a prescribed geometry. The technique inverts the relation between processing parameters and product dimensions. The inverse model has the desired product dimensions as inputs and the corresponding processing parameters as outputs. The effects of the design factors of the ANNs on the performance of mapping between product dimensions and processing parameters have also been investigated. Finally, experimental manufacturing of cups was used to test the adequacy of the method developed.

2 ARTIFICIAL NEURAL NETWORKS AND THE BACKPROPAGATION LEARNING RULE

ANN (artificial neural network) is one of the proven general-purpose architectures that are able to generate a nonlinear mapping between two sets of data. In this paper, the inverse model is treated as a mapping between the desired product geometry and the corresponding processing parameters, where the former are inputs and the latter are outputs.

The network architecture used for the inverse model problem is the multilayered feedforward neural network (MFNN)[2]. Figure 1 shows an MFNN with two hidden layers, each layer has a synaptic weight matrix W_i, i = 1, 2, 3, defining the connections between the previous layer and the next layer [3-9]. In Figure 1, the connection between the input patterns and the first hidden layer is defined by the weight matrix $W_1 = [w_{ij}^{(1)}] \in \Re^{N_1 \times N_{in}}$, where N_1 is the number of neurons in the first hidden layer and N_{in} is the number of inputs. The connection between the two hidden layers is defined by the weight matrix $W_2 = [w_{ij}^{(2)}] \in \Re^{N_2 \times N_1}$, where N_2 is the number of neurons in the second hidden layer. The last weight matrix $W_3 = [w_{ij}^{(3)}] \in \Re^{N_3 \times N_2}$ defines the connection between output patterns and the second hidden layer, where N_3 is the dimension of output patterns. Given the network input vector $u \in \Re^{N_{in} \times 1}$, the output of the first hidden layer $X_1 \in \Re^{N_1 \times 1}$, which is the input to the second hidden layer, can be written as

$$X_1 = f_1(W_1 . \ u + B_1)$$

where $f_1(\ )$ is the nonlinear activation function at each neuron in the first hidden layer and $B_1 \in \Re^{N_1 \times 1}$ is the threshold or bias vector of the same layer. The output of the second hidden layer $X_2 \in \Re^{N_2 \times 1}$ can be written as

$$X_2 = f_2 (W_2 . \ X_1 + B_2)$$

where f_2 ( ) is the nonlinear activation function and B_2 is the bias vector of the second hidden layer. The output of the last layer, which is the response of the network $Y = X_3 \in \Re^{N_3 \times 1}$, can be written as

$$Y = X_3 = f_3(W_3.\ X_2 + B_3)$$

where f_3 ( ) is the activation function and B_3 is the bias vector of the output layer.

In the following descriptions, the hidden layers of the ANN under study are composed of neurons with a hyperbolic tangent sigmoid activation function defined as:

$$y = f(v) = \tanh(\alpha v) = \frac{e^{\alpha v} - e^{-\alpha v}}{e^{\alpha v} + e^{-\alpha v}} = \frac{1 - e^{-2\alpha v}}{1 + e^{-2\alpha v}}$$

where the value of α is usually set to be 1. Furthermore, the activation function of the output layer is the linear function. That is,

$$Y = W_3.\ X_2 + B_3$$

The backpropagation learning algorithm is the most widely used training process for MFNN with differentiable activation functions today. The algorithm is based on the steepest descent gradient principle aiming at the minimization of deviation between the desired network output and the actual output, defined as the scalar positive function

$$E = \sum_{i=1}^{N_3} (d_i - y_i)^2$$

where d_i represents the desired network output and y_i is the actual output of the MLP corresponding to an input pattern. To emphasize relative deviation, the value of E is further tailored to be in a root of mean squared error form in the subsequent discussion:

$$E_{RMS} = \sqrt{\frac{1}{N_3} \sum_{i=1}^{N_3} (d_i - y_i)^2} \qquad \textbf{(1)}$$

On using the backpropagation algorithm, also called the generalized delta rule, we need to find the local error or delta, $\delta_j^{(s)}$, recursively first:

$$\delta_j^{(s)} = [\sum_{h=1}^{N_{s+1}} \delta_h^{(s+1)} \cdot w_{hj}^{(s+1)}] \cdot g_s(\sum_{i=1}^{N_{s-1}} w_{ji}^{(s)} \cdot x_i^{(s-1)} + b_j^{(s)}) \qquad \textbf{(2)}$$

where s = 1, 2 designates the appropriate network layer, g_s( ) represents the first derivation of the activation function f_s( ) in the s-th layer. And $\delta_j(3)$ for the output layer is evaluated as

$$\delta_j^{(3)} = (d_j - y_j) \cdot g_3(\sum_{h=1}^{N_2} w_{jh}{}^{(3)} x_h^{(2)} + b_j)$$

The learning rule for the weight matrix and bias vector is then given by

$$w_{ji}{}^{(s)}(k+1) = w_{ji}{}^{(s)}(k) + \mu \cdot \delta_j^{(s)} \cdot x_i{}^{(s-1)} \qquad (3)$$

$$b_j^{(s)}(k+1) = b_j^{(s)}(k) + \mu \cdot \delta_j^{(s)} \qquad (4)$$

where k denotes time index, s = 1, 2, 3 designates the appropriate network layer. Furthermore, μ in *Eqs 3* and *4* is the corresponding learning rate parameter, which is an important design factor of an MFNN [2].

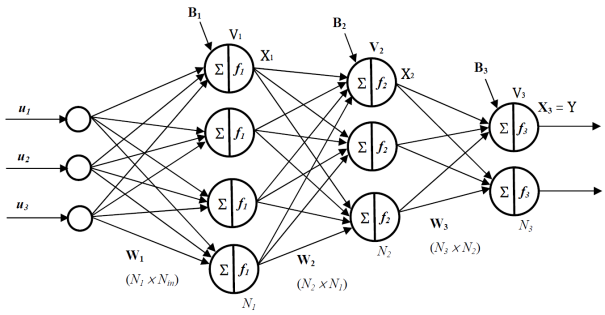

Figure 1. Architecture of a multilayered feedforward neural network (MFNN) with two hidden layers [2].

3 RESULTS AND DISCUSSION

With the network topology and learning rate paramete r being set at optimal values obtained above, Figure 2 shows the convergence history of the RMS prediction error values of the various network outputs. The average training time of a typical run was 12 seconds when executed using a 2.0 GHz Pentium 4 Personal Computer equipped with 1 GB RAM.

We can see that the average RMS error values are all below 0.2. Furthermore, the RMS errors of *E* and *F*, that is, the dropping velocity of the assisting plug and the heat transfer coefficient of the assisting plug, can be perfectly reduced below 0.01. We have that the network can fit to the 35 experimental data sets with RMS prediction errors well below 0.02.

Using the training data to assess the network performance can lead to over-fitting. The generalization properties of a neural network cannot be based on the training data alone. Untrained data sets must be used instead to evaluate its generalization capability. We randomly selected 5 experimental data sets, which were not used in the training, as test data sets.

To test the effectiveness of the trained ANN and study fundamental problems associated with inverse model of the thermo forming process, evaluation processes were conducted. The desired cup geometry, which serves as inputs to the ANN, is designed with ANSYS using the Finite Element Method (FEM).

The desired product dimensions were normalized and sent to the trained MFNN to generate outputs. The processing parameters corresponding to the optimal thickness distribution are: A = 187 ^{o}C, B = 91 mm, C = 0.3042 sec, D = 0.02 MPa, E = 222 mm/s, and F = 0.23 W/m·^{o}K (plastic).

Ten cups were manufactured and the average thicknesses of the six sites were obtained. These results are shown graphically in Figure 3. Except the deviation at point 3, which amounts to 17.14 %, dimension deviations of the other points are all below 3.5 % [2].

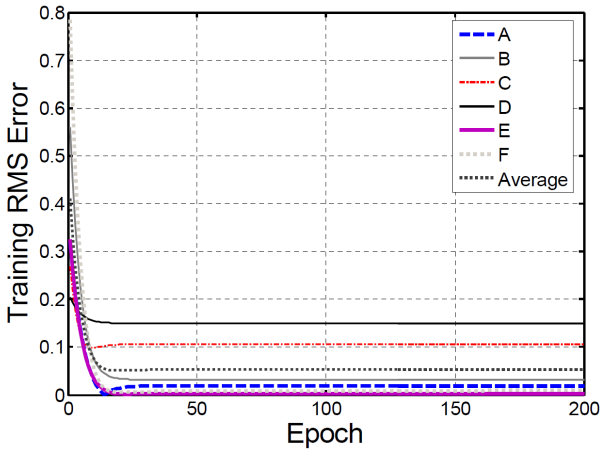

Figure 2. Convergence history of the RMS error values of the training data sets [2].

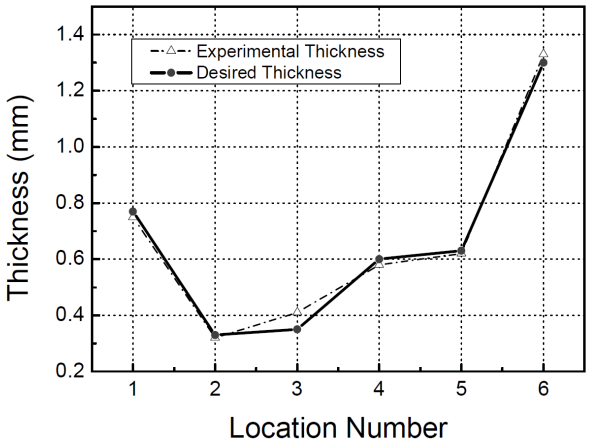

Figure 3. Experimental thickness versus desired thickness [2].

4 CONCLUSION

In this study, inverse model of thermoforming, which is composed of an artificial neural network, has been carried out. Taking thermoforming of the polypropylene foam cups as a specific case, we show in this paper that inverse modeling of a process by a neural network to derive processing parameters is both feasible and practical. This conclusion is justified by experimental results. In addition, inverse model can alleviate the prolonged trial processes required to derive a set of processing parameters corresponding to specified product geometry.

REFERENCES

1. J. L. Throne, Thermoforming, Hanser Publishers, New York (1986).
2. Y.Z. Chang, T.T. Wen, S.J. Liu, Polym. Eng. Sci. 45, 375 (2005)
3. F. M. Ham and I. Kostanic, Principles of Neurocomputing for Science and Engineering, McGraw-Hill (2001).
4. M. M. Gupta, L. Jin, and N. Homma, Static and Dynamic Neural Networks, John Wiley & Sons, Inc (2003).
5. V. Maniezzo, IEEE Trans. on Neural Networks, 51, 39 (1994).
6. D. E. Moriarty and R. Miikkulainen, Evol. Comput., 5, 373 (1997).
7. B. Yang, X.-H. Su, and Y.-D. Wang, the First Int.Conf. on Mach. Learn. and Cybernet. Conf. Proceed., 64 (2002).
8. J. A. Nelder and R. Mead, Comput. J., 7, 308 (1965).
9. W. H. Press, Numerical Recipes in C++: The Art of Scientific Computing, 2nd ed., Cambridge Univ. Press, New York (2002).