Pergamon

Computers ind. Engng Vol. 34, No. 2, pp. 433–450, 1998

Printed in Great Britain
PII: S0360-8352(97)00141-1
0360-8352/98 $19.00 + 0.00

A NEURAL NETWORK APPROACH FOR EARLY COST ESTIMATION OF PACKAGING PRODUCTS

Y. F. ZHANG and J. Y. H. FUH

Department of Mechanical and Production Engineering, National University of Singapore, 10 Kent Ridge Crescent, Singapore, 119260 Singapore

(Received 1 June 1997)

Abstract—Product costs need to be identified early, i.e., during the design stage, where they can be controlled best. This implies the need to estimate the product's cost without full knowledge of the manufacturing process plans. In this paper, a feature-based cost estimation using a back-propagation neural network is proposed and a prototype system has been developed for estimating the costs of packaging products based on design information only. All the cost-related features of a product design were extracted and quantified according to their cost effects. The correlation between these cost-related features and the final cost of the product was established by training a back-propagation neural network using historical cost data. The extraction of cost-related features and the construction, training and validation of the neural network are described. The performance of the trained neural network based on a set of testing samples is also given. © 1998 Elsevier Science Ltd. All rights reserved.

INTRODUCTION

It has been widely reported that 70–80% of the product costs are committed during the design stage although only a small proportion of the total product costs has been actually incurred [1–3]. Therefore, there are significant savings in product costs, as well as improved productivity and product quality if necessary changes of the product are made in the early design stage. A prerequisite for making good product design is the availability of product cost information. The problem of cost estimation has traditionally been the province of manufacturing engineers. Typically, the cost of a product is derived by summation of the various cost components such as the material cost, machining cost, direct labour cost, administration and engineering cost, etc. However, this requires information not only on a detailed product design but also its process plan and indirect cost factors which are not available in the design stage. A product cost estimation method based on design information is therefore a necessary step to help the designer achieve good trade-off decisions. Generally, the implication of a product's design specifications on its manufacturing processes and cost is consistent given consistent manufacturing practices. Therefore, sufficiently accurate product cost estimates can be achieved even without full knowledge of the process steps and equipment to be used. In this context, product design specifications can be viewed as cost drivers. The explicit cost drivers, such as material cost, can be obtained directly, while the implicit ones, such as complexity-related cost, have to be derived through the analysis of historical cost data. Therefore, the key to product cost estimation is how to use the historical cost data to understand the effects of the implicit cost drivers.

In this study, a new approach, feature-based cost estimation, is proposed for packaging products based on the fact that all the design features of a product have contributed to the final cost of a product in one way or another. The approach involves extraction of cost-related features from typical packaging products and the identification of the cost estimation function using historical cost data. However, such a function cannot be obtained through conventional methods such as regression analysis without knowing the nature of nonlinearity of the correlation. A new method for function approximation is therefore required.

Neural network has been considered as a powerful tool for function approximation. One advantage of neural networks is that they are capable of learning by examples [4, 5]. This implies that they can be trained to perform tasks by presenting them with examples rather than specifying the procedure. A back-propagation neural network has been constructed and trained using

historical data to approximate the relationships between the cost-related features and the overall product cost. Once the training is completed, the user does not need any knowledge of manufacturing process, process parameters, etc. The input contains only information available in the early design stage, e.g., lot size, material, design features, etc. This paper describes how the neural network approach can be applied to product cost estimation.

PREVIOUS WORK IN PRODUCT COST ESTIMATION

Since the concept of concurrent engineering was introduced in early 1980, there have been a large amount of literature on cost estimation; some of them are aimed particularly at cost estimation during early design stage. Most approaches can be broadly classified into the following categories: 1) traditional detailed-breakdown cost estimation, 2) simplified breakdown cost estimation, 3) group technology (GT-)based cost estimation, 4) cost estimation based on cost functions or cost increase functions, and 5) activity-based cost (ABC) estimation.

In the traditional detailed-breakdown approach, the total cost during the production cycle of a product is typically the summation of all the cost incurred, such as material cost, manufacturing cost, labour cost, overheads, etc. The calculation, however, requires detailed information on design, process routing, and machining time. The accuracy of the cost estimation depends on the availability of the required information and the variance of the actual activities. A comprehensive description of this method was given by Ostwald [6].

The simplified breakdown methods are mainly developed for cost estimation in the early design stage when a process plan is not available. Such cost estimation is based on the assumed optimum manufacturing methods irrespective of the processes and equipment which will actually be used. Empirical equations are often required to be developed for breakdown elements. A shortcoming of this approach is that it is difficult to update the cost estimate based on actual cost information. The work of Dewhurst and Boothroyd [7] on early cost estimation models for machining and injection moulding components, Boothroyd and Reynolds [8] on a cost model for typical turned parts, and Boothroyd and Radovanovic [9] on a cost model for machined components are representative examples of this category.

The GT-based approach is based on the similarity principle. It typically uses a basic cost value while taking into account the effects of variable cost factors such as size, complexity, etc. Linear relationship between the final cost and the variable cost factors is assumed. Among the examples of this category, Poli *et al.* [10] developed a tool to estimate injection mould tooling cost using a six-digit code of a part design. Look-up tables were used to extract the cost drivers based on the code.

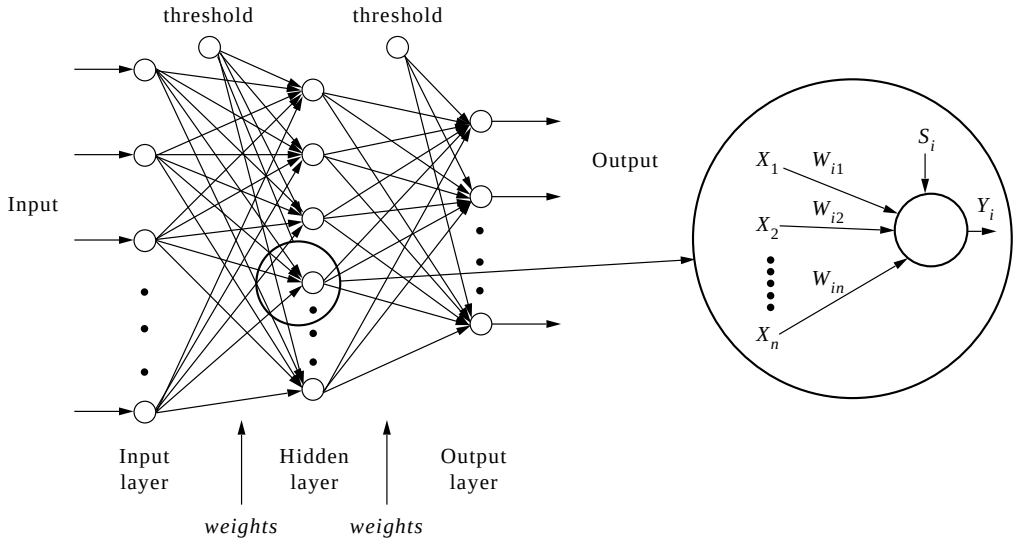


Fig. 1. Concept of back-propagation neural network.

The cost function method combines both technical and economical variables into one equation, yielding the costs as the function of product properties. The cost increase function method is used for a specified group of products. A product's cost can be represented as a function related to the cost of the basic design for that group. The coefficients and exponents are derived through the regression analysis of historical cost data. The problem of using cost functions lays in deducing them. With a large number of variables, the function tends to be complicated, therefore more difficult to present. Moreover, the cost function approach has its limited applicability: for each group of products a new function has to be derived. A more comprehensive review on this method was given by Wierda [11] and Hundal [12].

Activity-based costing (ABC) is a method for accumulating product costs by determining all cost drivers associated with the activities required to produce the product. Based on historical, observed, or estimated data, the cost per unit of the activity's output is calculated. The estimated cost for a new product can be obtained according to the product's consumption of these activities. However, it is difficult to update the system based on actual cost data. The work of Ong [13] on cost tables to support wire harness design in the assembly of electrical connections is an example of this category.

There has been very little work on product cost estimation using neural networks. Shtub and Zimerman [14] applied back-propagation neural networks to estimate the cost of assembly systems. The emphasis was given on the comparison of performance between the neural network model and the regression model. Our approach for product cost estimation can be considered as a complementary work for other areas in applying neural network. In this work, we demonstrate the potential of back-propagation neural networks in early cost estimation. Supervised learning based on historical cost data is used to obtain the relationship between the cost of packaging product and its cost-related features.

BACK-PROPAGATION NEURAL NETWORKS

A back-propagation neural network [4, 5, 15] is a multiple-layer network with an input layer, an output layer, and some hidden layers between the input and output layers. Each layer has a number of processing unit, called *neurons*. Figure 1 shows a three-layer back-propagation neural network. A neuron simply computes the sum of their weighted inputs, subtracts its threshold from the sum, and passes the results through its transfer function. This can be expressed mathematically as

$$y_{\mathrm{i}} = f_{\mathrm{i}}\left(\sum_{j=1}^{n} w_{\mathrm{ij}} x_{\mathrm{j}} - s_{\mathrm{i}}\right) \tag{1}$$

where y_i represents the output of the neuron, w_{ij} represents the weight associated with the input j, s_i represents the threshold value of the neuron, and f_i represents the transfer function. The three commonly used transfer functions for back-propagation neural networks are Log-Sigmoid, Tan-Sigmoid, and Linear function (see Fig. 2). The input to a neuron can be either the output from other neurons, or directly from the external input to the network. The output from a neuron can be used either as an input to the subsequent neurons or an output of the network. The value of w_{ij} determines how strongly the input influences the activity of a neuron. The magnitude of a weight can be changed during a training process. It is mainly through this mechanism that the neuron is made adaptive to new information presented to it and hence the learning process is accomplished. The total weights of a neural network reflect the knowledge and skills that it has learnt through previous training and determine how the network will react when an unknown input is presented to it.

The back-propagation neural networks refer to their training algorithm, known as error back-propagation or generalised delta rule. The training of such a network starts with assigning random values to all the weights. An input is then presented to the network and the output from each neuron in each layer is propagated forward through the entire network to reach an actual output. The error for each neuron in the output layer is computed as the difference between an actual output and its corresponding target output. This error is then propagated backwards

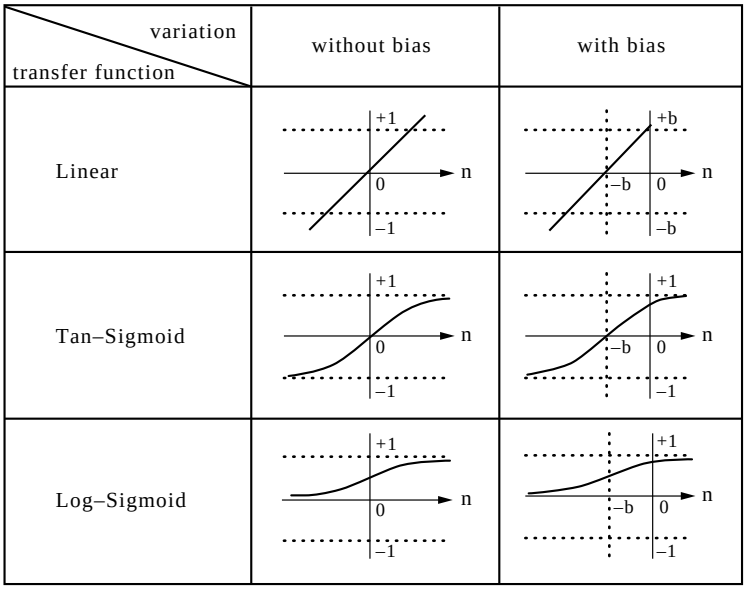


Fig. 2. Some commonly used transfer functions.

through the entire network and the weights are updated. The weights for a particular neuron are adjusted in direct proportion to the error in the units to which it is connected. In this way the error is reduced and the network learns. This training process is also called *supervised training* since the target output for each input is known.

One of the most important characteristics of back-propagation neural networks is their ability to learn by examples. With proper training, the network can memorise the knowledge in the problem solving of a particular domain. Since manufacturing problems are generally not well structured, they provide a potential ground for neural network applications. So far, attempts have been made in many manufacturing areas including group technology, feature recognition, tool condition monitoring, etc. [16–18].

THE OVERVIEW OF THE NEURAL NETWORK APPROACH

As shown in Fig. 3, the proposed neural network approach for product cost estimation has two operational stages: the *preparatory* stage and the *production* stage. The preparatory stage is to construct and train a neural network using existing products and their historical cost data. During this stage, the domain of the product spectrum is defined, all the cost-related features of the product are identified and quantified according to their influence to the final cost. Next, existing products are coded to form the cost-related feature matrices. A back-propagation neural network is then constructed by specifying the number of layers and the number of neurons in each layer. However, the weight for each interconnection is randomly assigned. The training can be started using the cost-related feature matrices as the inputs and the products' historical cost as the target outputs. The training will stop when a certain error value indicated at the end of training is reached. The final network configuration and weights for all the interconnections are then stored in a "knowledge base" and the neural network is ready for estimating the costs of new products.

The production stage occurs when the neural network is in use for cost estimation of a new product. During this stage, an incoming product is first coded to obtain its cost-related feature matrix according to its design specifications. The matrix is then input into the network. By using the existing weight matrix obtained from the training, an estimated cost of this product can be obtained.

It is worth to note that the preparatory stage can be repeated if the cost estimation for new product does not meet the expectation. This can be done by adding the new product to the training samples and retraining the network accordingly.

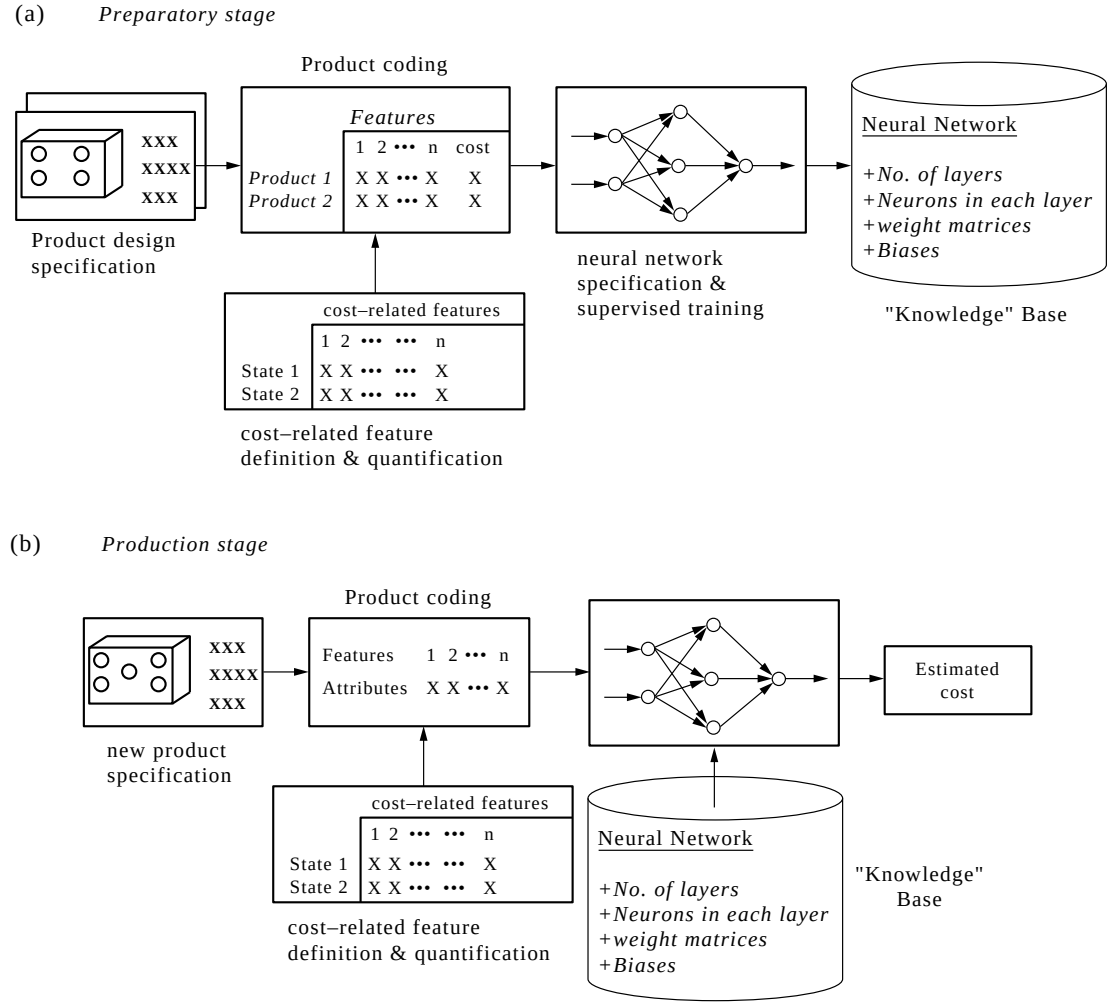


Fig. 3. Stages of product cost estimation using a back-propagation neural network approach.

COST-RELATED FEATURES IN PACKAGING PRODUCTS

A cost-related feature of a product can be defined as the characteristics associated with the product, which makes certain contributions to the total cost of the product. It can be either a design specification or a processing requirement. However, only those related with product design specifications are considered in this study since information on manufacturing processes is not readily available in the design stage. Also, cost-related features should be product-type dependent. It can be broadly classified into many categories, such as machined products, casting products, injection-moulded products, and packaging products, etc.

The type of packaging product we studied is the *regular slotted carton* (RSC). An example of an unfolded RSC design is shown in Fig. 4. The raw material for the packaging products is paper which has to be converted into corrugated boards before it is used for manufacturing the carton boxes. A corrugated board consists of a piece of corrugated paper called the fluting sandwich between two sheets of plain papers known as the liners. Most of the cartons also consist of some printing which may be wordings or logos.

Based on our study on the RSC design and its manufacturing processes, various cost-related features based on product designs are extracted and classified under the aspects of *material*, *printing plates*, *printing features*, and *production requirement* (lot size). As required, the cost-related features can be obtained easily by a user without much computation or reasoning based on product specification only.

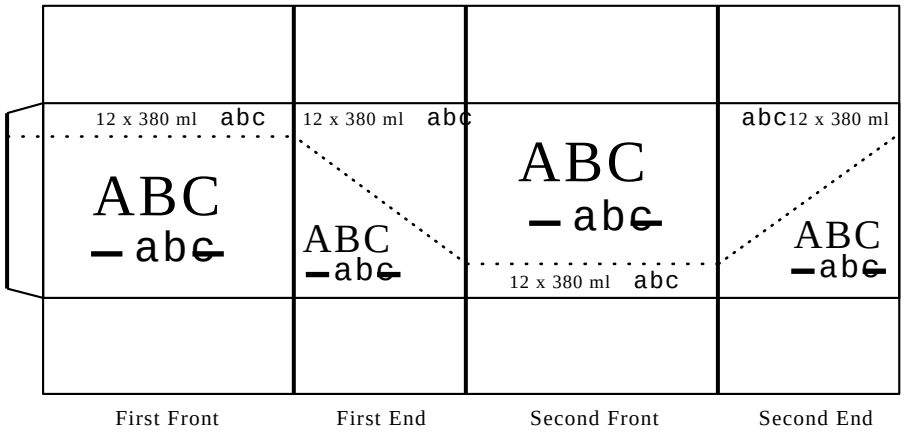


Fig. 4. The layout of an unfolded RSC design.

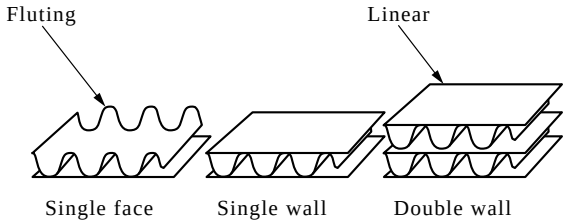


Fig. 5. Types of corrugated board construction.

Since a neural network can only perform arithmetic operations, each cost-related feature of a product is assigned a value between 0 and 1 called a *cost-related feature attribute* (CFA). This process is called the quantification of cost-related features. The magnitude of a CFA is an indication of the relative cost influence among the various possible states of each feature. A state which contributes to a higher cost will be assigned a higher value and vice versa. Although the values of CFAs are strongly related within the same feature, they are independent to the ones assigned to other features. Finally, all the cost-related features of a product can be converted to a CFA vector which can then be used as the input to the neural network for training or cost estimation.

The four possible cost-related features and their quantifications are described in the following subsections.

Cost-related features based on material

1. *Type of corrugated board construction* (CFA1): there are three types of corrugated board construction, namely, *single face*, single wall, and *double wall* (see Fig. 5). Generally, the thicker the corrugated board, the higher the cost will be. The quantification of CFA1 is given in Table 1.

2. *Quality of material* (CFA2, CFA3, CFA4, CFA5, CFA6): the quality of material is defined as the weight per unit area of the paper. Based on the type of corrugated board construction, CFA2, CFA3, and CFA4 refer to the quality of the first liner, the second liner, and the third liner, respectively. The quantification of liner quality is given in Table 2. Likewise, CFA5 and CFA6 refer to the quality for the first fluting and the second fluting respectively. Detailed quantification of fluting quality is given in Table 3.

Table 1. Quantification of corrugated construction type

	Type		
	single face	single wall	double wall
CFA1	0.3	0.4	0.6

Table 2. Quantification of liner quality

	Type										
	K4	K5	K6	K7	K8	K9	W5	W6	W7	W8	N.A.
CFAx	0.52	0.5	0.42	0.44	0.46	0.48	0.54	0.56	0.58	0.6	0
	T4	T5	T6	T7	M1	M2	M3	M4	M5	M6	N.A.
CFAx	0.34	0.36	0.38	0.4	0.18	0.2	0.22	0.24	0.26	0.28	0

x = 2,3,4.
N.A.: not applicable.

Table 3. Quantification of fluting quality

	Type												
	F5	F6	M1	M2	M3	M4	M5	M6	T4	T5	T6	T7	N.A.
CFAx	0.3	0.32	0.18	0.2	0.22	0.24	0.26	0.28	0.34	0.36	0.38	0.4	0

x = 5,6.

3. *Height of fluting* (CFA7, CFA8): the height of the fluting is defined as the distance between two liners as shown in Fig. 6. There are in total three types of fluting height, viz. 0.1, 0.15, and 0.2. The larger the height, the higher the cost is. CFA7 and CFA8 refer to the height of the first fluting and the second fluting respectively.

4. *Size of the carton* (CFA9): this concerns the amount of raw material used per carton and is expressed as the area required by the corrugated board,

$$\text{CFA9} = \frac{\text{area}(\text{m}^2)}{20} \tag{2}$$

where 20 m^2 is the largest possible size of the area of a product.

Cost-related features based on printing plates

1. *Availability of printing features* (CFA10): products with printing features cost more than the ones without printing features. If there are printing features, CFA10 is set to 0.5; otherwise, CFA10 is set to 0.

2. *Material of printing plates* (CFA11): the types of material used for printing plates are normally rubber and photopolymer. The cost of the latter material is approximately three times that of the rubber. Quantification of printing plate material is given in Table 4.

3. *Types of cut for the rubber plate* (CFA12): the cut for the rubber plate can be either *normal* or *reverse* (see Fig. 7). The *normal* type has wordings or logos printed with colours. A *reverse* cut on the other hand prints the surroundings with colours leaving the wordings or logos uncoloured. In general, *reverse* cutting is more difficult and hence costs more. Quantification for the type of cut is given in Table 5.

4. *Size of the printing plates* (CFA13, CFA14): the size of the printing plates is defined as the area of the rubber or photopolymer required. CFA13 refers to rubber and CFA14 to photopolymer.

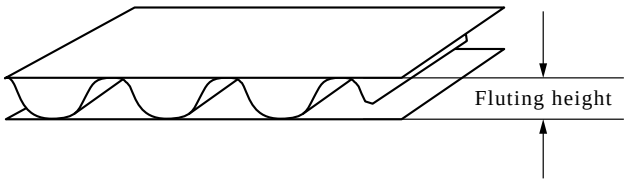


Fig. 6. The fluting height.

$$\text{CA13} = \frac{\text{rubber area (m}^2\text{)}}{2.5} \tag{3}$$

$$\text{CFA14} = \frac{\text{photopolymer area (m}^2\text{)}}{0.7} \tag{4}$$

where $2.5\,\text{m}^2$ and $0.7\,\text{m}^2$ are the largest size of printing plates which can be produced from rubber and photopolymer, respectively.

Cost-related features based on printing features

1. *Number of colours* (CFA15): the maximum number of colours that can be printed in one pass in the manufacturing environment studied is three. Any number greater than three will require more passes and thus higher cost. Even when the number of colours is no more than three, the larger the number, the more set-up time will be required. Quantification on number of colours is given in Table 6.
2. *The printed area* (CFA16): the printed area determines the amount of print required directly. Moreover, it affects the quality of the print as well. This is mainly due to the unevenness of the corrugated board, which in turn causes the ink to be unable to reach the valleys of the surface. The larger the printed area is, the higher the chance the print could not achieve the required quality and hence additional cost is incurred for amendments. Quantification on the printed area (based on the percentage of the total area) is given in Table 7.
3. *Effects of colour trapping* (CFA17, CFA18, CFA19): colour trapping refers to a situation where the printing consists of two or more different colours that are printed in contact with one another (see Fig. 8). The problem of colour trapping arises from the misalignment of the printing plates. Caution is hence to be taken to ensure that the various printing plates are positioned properly in order to minimize the effect of one colour overlapping another. As a result, more

reverse cut normal cut

Fig. 7. Types of cut for rubber plate.

Table 4. Quantification of printing plate material

	Type			
	Photopolymer	Rubber	Both	N.A.
CFA11	0.6	0.2	0.4	0

Table 5. Quantification on types of cut

Photopolymer	Rubber			
	Type			
	normal	reverse	both	N.A.

Table 6. Quantification on number of colours

	No. of colours				
	1	2	3	>3	N.A.
CFA15	0.2	0.3	0.4	0.7	0

Table 7. Quantification on printing area

	Printed area				
	< 30%	30–50%	50–70%	> 70%	N.A.
CFA16	0.2	0.4	0.6	0.8	0

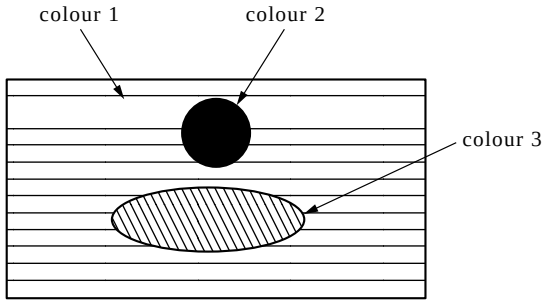


Fig. 8. An example of colour trapping.

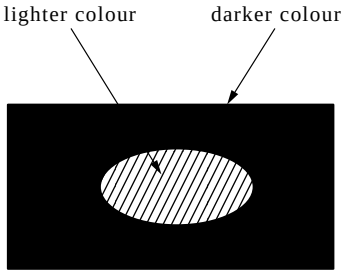


Fig. 9. Colour trapping between a darker and a lighter colour.

samples are usually required during the process of aligning the printing plates. In some cases, the print may involve colour trapping between a dark and a light colour. The lighter colour is often printed first and any misalignment can be later covered up by the darker colour (see Fig. 9). The effect of misalignment is hence reduced. CFA17 refers to the availability of colour trapping. A positive answer gives 0.5 to CFA17; otherwise, CFA17 is set to 0. CFA18 refers to the number of printing plates for different colour trapping and CFA19 refers to the type of colour trapping. Quantification of CFA18 and CFA19 is given in Table 8 and Table 9, respectively.

4. *Availability of printing inks* (CFA20): only the inks that are frequently used are stored in the shop floor. If an ink required is not available, it has to be specially purchased and often there will be some excess after the printing is done. Extra cost is therefore incurred. CFA20 is set to 0.2 if required printing inks are available; otherwise CFA20 is set to 0.4. If no printing is required, CFA20 is set to 0.

Table 8. Quantification on number of printing plates for different colour trappings

	Type			
	2	3	4	N.A.
CFA18	0.3	0.4	0.5	0

Table 9. Quantification on type of colour trapping

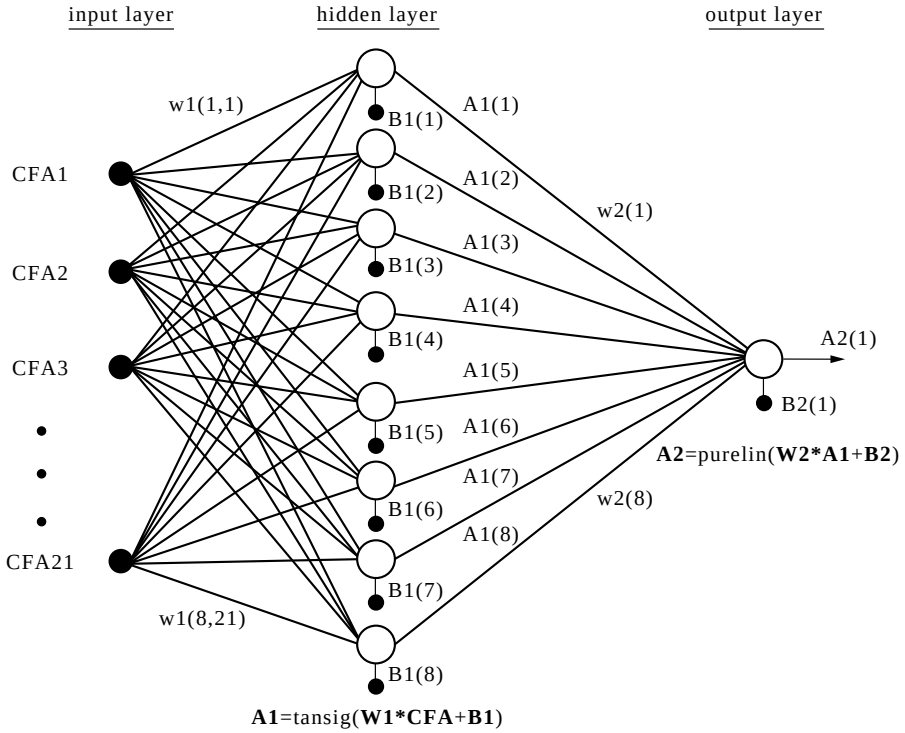
	Type			
	Between dark and light	Between light only	Both	N.A.
CFA19	0.3	0.6	0.5	0

Cost-related features based on production requirement

Lot size (CFA21): in general, the larger the lot size, the smaller the cost for each product will be, since the shared set-up costs and printing plate costs by each product is lower.

$$\mathrm{CFA21} = \frac{\text{lot size}}{50000} \tag{5}$$

where 50 000 is the largest lot size ever produced in the company.



Where,

CFA = [CFA1, CFA2, CFA3, ... CFA21]^{T}

B1 = [B(1), B1(2), B1(3), B1(4), B1(5), B1(6), B1(7), B1(8)]^{T}

B2 = [B2(1)]

A1 = [A1(1), A1(2), A1(3), A1(4), A1(5), A1(6), A1(7), A1(8)]^{T}

A2 = [A2(1)]

$$\mathbf{W1} = \begin{bmatrix} w1(1,1) & w1(1,2) & \cdots & \cdots & w1(1,21) \\ w1(2,1) & w1(2,2) & \cdots & \cdots & w1(2,21) \\ w1(3,1) & w1(3,2) & \cdots & \cdots & w1(3,21) \\ w1(4,1) & w1(4,2) & \cdots & \cdots & w1(4,21) \\ w1(5,1) & w1(5,2) & \cdots & \cdots & w1(5,21) \\ w1(6,1) & w1(6,2) & \cdots & \cdots & w1(6,21) \\ w1(7,1) & w1(7,2) & \cdots & \cdots & w1(7,21) \\ w1(8,1) & w1(8,2) & \cdots & \cdots & w1(8,21) \end{bmatrix}$$

W2= [w2(1) w2(2) w2(3) w2(4) w2(5) w2(6) w2(7) w2(8)]

Fig. 10. The back-propagation neural network architecture for cost estimation.

The quantification of the cost-related features is basically based on the cost index used by the company to estimate the product cost using a linear equation. The values assigned to various states of a cost-related feature are arbitrary in particular for those features with only two discrete states, e.g., Yes/No. During our study, it was observed that different quantification of these Yes/No features (e.g., 0.0/0.5 or 0.2/0.4) hardly affects the capability of the trained neural network as long as the qualitative cost relationship between the "Yes" and "No" states is correct. This is understandable since a neural network trained using the same set of samples but with a different quantification scheme will have different network parameters (weights and biases). This also demonstrates the learning ability and robustness of the neural networks approach.

THE ARCHITECTURE OF THE NEURAL NETWORK

A feed-forward back-propagation neural network is adopted to be trained for cost estimation. The neural network has three layers: an input layer, a hidden layer, and an output layer (see Fig. 10). The number of neurons in the input layer is the same as the dimension of the CFA matrix (**CFA**), i.e., 21 in this application. The output layer has only one neuron which represents the cost. The hidden layer has eight neurons. In fact, both 1-hidden layer and 2-hidden layer networks have been tried, and the results seemed very close while training with 2-hidden layers takes much longer. The number of neurons in the hidden layer is also determined through a trial-and-error. **CFA** is the input vector, **W1** and **W2** are the weight matrices for the interconnections between the input layer and the hidden layer and that between the hidden layer and the output layer, respectively. **B1** and **B2** are bias vectors for the neurons in the hidden layer and the output layer. Biases give a network more freedom to learn the desired function. Using them tends to increase the chances that a back-propagation network will find an acceptable solution and also tends to decrease the number of training epochs required. **A1** represents the output vector from the hidden layer, while **A2** represents the output vector from the output layer, which is the cost. The transfer function for the hidden layer is the Tan-Sigmoid, while the linear function is used for the output layer since the final cost value can be larger than 1.0.

As for implementation, a program was written using the MATLAB Neural Network Toolbox [19] to access the performance of the neural network on the training, validation, and testing samples. The performance measure is based on the cost percentage error (CPE) which is expressed as follows:

$$\mathrm{CPE}(i) = \frac{E(i) - T(i)}{T(i)} \times 100\% \tag{6}$$

where $E(i)$ is the estimated cost of sample i from the network and $T(i)$ is the targeted cost.

TRAINING OF THE BACK-PROPAGATION NETWORK

The network described above has been trained using the *back-propagation training rule*. The weights and biases of the networks are adjusted in order to minimize the sum squared errors of the network. This is done by continually changing the values of the network weights and biases in the direction of steepest descent with respect to the error. The training process is described in detail in the following steps:

1. *Preparing input and output of the training samples:* training samples are collected from the products which have been produced. The training samples should be *representative* enough to cover most cost-related features of the whole product spectrum. The neural network can therefore be trained comprehensively by exposing itself to various situations. Sufficient knowledge on the product variation can then be gained. As a result, 66 existing products were selected as training, validation, and testing samples. These include RSCs with and without colour trapping,

as well as RSCs with no printing features. Their **CFA**s extracted from the product design specifications and the actual cost data **T**s provided by the company are listed in Tables 10–12.

2. *Initialising the training session:* before the training starts, the parameters of the network for the training must be initialised. The initial weights and biases are generated by random functions with values between −1 and +1.

3. *Training:*

(a) The training data **CFA**s and **T**s are presented to the network in the form of vectors $[\mathbf{CFA}^1, \mathbf{CFA}^2, \ldots, \mathbf{CFA}^1, \mathbf{CFA}^m]^T$, and $[\mathbf{T}^1, \mathbf{T}^2, \ldots, \mathbf{T}^i, \ldots, \mathbf{T}^m]^T$, where m is the number of training samples.

(b) The input vector of each sample is pre-multiplied by a set of weights, and summed up with the biases through the transfer function to generate an output **A2**.

Table 10. The CFAs, actual costs (CostT) and estimated costs (CostE) of training samples

	Sample #															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
CFA1	0.4	0.4	0.6	0.6	0.4	0.6	0.6	0.4	0.4	0.4	0.4	0.6	0.4	0.4	0.6	0.4
CFA2	0.5	0.5	0.5	0.52	0.5	0.42	0.52	0.52	0.54	0.5	0.54	0.5	0.42	0.5	0.42	0.42
CFA3	0.52	0.52	0.18	0.18	0.5	0.18	0.18	0.52	0.52	0.5	0.42	0.18	0.42	0.5	0.18	0.42
CFA4	0	0	0.5	0.52	0	0.42	0.52	0	0	0	0	0.5	0	0	0.42	0
CFA5	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
CFA6	0	0	0.2	0.2	0	0.2	0.2	0	0	0	0	0.2	0	0	0.2	0
CFA7	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.15	0.1	0.1	0.15
CFA8	0	0	0.15	0.15	0	0.15	0.15	0	0	0	0	0.15	0	0	0.15	0
CFA9	0.089	0.065	0.042	0.084	0.025	0.057	0.100	0.026	0.035	0.023	0.042	0.024	0.041	0.030	0.049	0.055
CFA10	0.5	0.5	0.5	0	0.5	0.5	0	0	0.5	0.5	0.5	0	0.5	0.5	0.5	0.5
CFA11	0.2	0.2	0.2	0	0.2	0.2	0	0	0.2	0.6	0.6	0	0.4	0.4	0.4	0.2
CFA12	0.2	0.2	0.25	0	0.2	0.2	0	0	0.25	0	0	0	0.25	0.2	0.25	0.2
CFA13	0.126	0.126	0.145	0	0.022	0.074	0	0	0.284	0	0	0	0.106	0.007	0.079	0.216
CFA14	0	0	0	0	0	0	0	0	0	0.44	0.288	0	0.072	0.016	0.212	0
CFA15	0.2	0.2	0.2	0	0.2	0.2	0	0	0.3	0.2	0.4	0	0.2	0.3	0.3	0.2
CFA16	0.4	0.4	0.4	0	0.2	0.2	0	0	0.8	0.6	0.2	0	0.6	0.2	0.4	0.6
CFA17	0	0	0	0	0	0	0	0	0.5	0	0.5	0	0	0	0.5	0
CFA18	0	0	0	0	0	0	0	0	0.3	0	0.3	0	0	0	0.3	0
CFA19	0	0	0	0	0	0	0	0	0.3	0	0.3	0	0	0	0.3	0
CFA20	0.2	0.2	0.2	0	0.2	0.2	0	0	0.4	0.4	0.4	0	0.2	0.2	0.2	0.2
CFA21	0.16	0.06	0.1	0.04	0.04	0.3	0.1	0.02	0.18	0.09	0.03	0.01	0.14	0.2	0.1	0.08
CostT	1.563	1.180	1.102	1.812	0.455	1.366	1.917	0.439	0.773	0.416	0.977	0.612	0.812	0.544	1.461	1.086
CostE	1.567	1.183	1.052	1.710	0.464	1.361	1.977	0.425	0.774	0.417	0.957	0.729	0.846	0.538	1.482	1.093

	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
CFA1	0.6	0.4	0.6	0.6	0.4	0.4	0.6	0.6	0.4	0.4	0.6	0.6	0.4	0.4	0.4	0.4
CFA2	0.44	0.54	0.54	0.42	0.5	0.5	0.5	0.5	0.5	0.52	0.5	0.5	0.42	0.42	0.5	0.5
CFA3	0.18	0.42	0.18	0.18	0.5	0.5	0.18	0.18	0.5	0.52	0.18	0.18	0.42	0.42	0.5	0.52
CFA4	0.44	0	0.42	0.42	0	0	0.5	0.5	0	0	0.4	0.5	0	0	0	0
CFA5	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.26	0.2	0.2	0.2	0.2	0.26	0.24
CFA6	0.2	0	0.2	0.2	0	0	0.2	0.2	0	0	0.2	0.2	0	0	0	0
CFA7	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.20	0.15	0.1	0.1
CFA8	0.15	0	0.15	0.15	0	0	0.15	0.15	0	0	0.15	0.15	0	0	0	0
CFA9	0.041	0.019	0.052	0.058	0.028	0.019	0.038	0.070	0.054	0.015	0.026	0.037	0.051	0.042	0.033	0.025
CFA10	0.5	0.5	0.5	0	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
CFA11	0.4	0.6	0.2	0	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.6	0.4	0.6	0.2	0.2
CFA12	0.25	0	0.2	0	0.2	0.2	0.2	0.2	0.2	0.2	0.25	0	0.25	0	0.2	0.25
CFA13	0.123	0	0.167	0	0.047	0.067	0.111	0.059	0.059	0.006	0.619	0	0.188	0	0.008	0.056
CFA14	0.081	0.12	0	0	0	0	0	0	0	0	0	0.494	0.072	0.221	0	0
CFA15	0.2	0.2	0.4	0	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.3	0.2	0.3	0.2
CFA16	0.6	0.6	0.6	0	0.4	0.6	0.6	0.2	0.2	0.2	0.4	0.6	0.6	0.2	0.4	0.4
CFA17	0	0	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0
CFA18	0	0	0.4	0	0	0	0	0	0	0	0	0	0	0	0	0
CFA19	0	0	0.3	0	0	0	0	0	0	0	0	0	0	0	0	0
CFA20	0.2	0.4	0.4	0	0.2	0.2	0.2	0.4	0.4	0.2	0.2	0.2	0.4	0.2	0.2	0.2
CFA21	0.07	0.06	0.09	0.08	0.12	0.03	0.2	0.15	0.1	0.08	0.03	0.1	0.15	0.08	0.02	0.03
CostT	1.174	0.429	1.649	1.438	0.487	0.447	0.882	1.591	0.986	0.346	0.656	0.918	1.159	0.805	0.625	0.479
CostE	1.158	0.434	1.650	1.379	0.505	0.385	0.912	1.593	1.022	0.286	0.664	0.916	1.122	0.820	0.673	0.500

Table 11. The CFAs, actual costs (CostT) and estimated costs (CostE) of validation samples

	Sample #																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
CFA1	0.6	0.6	0.6	0.6	0.6	0.4	0.4	0.6	0.6	0.4	0.6	0.4	0.6	0.4	0.6	0.4	0.4
CFA2	0.5	0.42	0.5	0.5	0.52	0.42	0.42	0.42	0.42	0.5	0.5	0.5	0.5	0.54	0.5	0.5	0.5
CFA3	0.18	0.18	0.18	0.18	0.18	0.42	0.42	0.18	0.18	0.5	0.18	0.5	0.18	0.42	0.18	0.5	0.5
CFA4	0.5	0.42	0.5	0.5	0.52	0	0	0.42	0.5	0	0.5	0	0.5	0	0.5	0	0
CFA5	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
CFA6	0.2	0.2	0.2	0.2	0.2	0	0	0.2	0.2	0	0.2	0	0.2	0	0.2	0	0
CFA7	0.1	0.1	0.1	0.1	0.1	0.1	0.15	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
CFA8	0.15	0.15	0.15	0.15	0.15	0	0	0.15	0.15	0	0.15	0	0.15	0	0.15	0	0
CFA9	0.043	0.060	0.119	0.057	0.040	0.031	0.051	0.060	0.061	0.080	0.043	0.056	0.052	0.035	0.108	0.041	0.036
CFA10	0.5	0.5	0	0.5	0.5	0.5	0.5	0.5	0	0.5	0.5	0.5	0	0.5	0.5	0.5	0
CFA11	0.2	0.2	0	0.2	0.2	0.2	0.4	0.2	0	0.2	0.2	0.2	0	0.6	0.2	0.2	0
CFA12	0.25	0.25	0	0.2	0.2	0.2	0.25	0.25	0	0.25	0.2	0.2	0	0	0.2	0.2	0
CFA13	0.027	0.049	0	0.003	0.062	0.037	0.263	0.072	0	0.069	0.108	0.039	0	0	0.059	0.059	0
CFA14	0	0	0	0	0	0	0.128	0	0	0	0	0	0	0.288	0	0	0
CFA15	0.2	0.3	0	0.2	0.2	0.2	0.3	0.3	0	0.3	0.2	0.3	0	0.4	0.2	0.2	0
CFA16	0.4	0.6	0	0.4	0.2	0.2	0.6	0.2	0	0.4	0.4	0.2	0	0.2	0.2	0.2	0
CFA17	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	0	0	0
CFA18	0	0	0	0	0	0	0	0	0	0	0	0	0	0.3	0	0	0
CFA19	0	0	0	0	0	0	0	0	0	0	0	0	0	0.3	0	0	0
CFA20	0.2	0.2	0	0.2	0.2	0.2	0.2	0.2	0	0.2	0.2	0.2	0	0.4	0.4	0.4	0
CFA21	0.06	0.10	0.13	0.1	0.2	0.2	0.11	0.02	0.05	0.04	0.04	0.05	0.06	0.08	0.07	0.1	0.02
CostT	0.975	1.623	2.685	1.323	0.908	0.601	1.057	1.630	1.414	1.612	1.094	0.997	1.146	0.843	2.401	0.769	0.616
CostE	1.096	1.494	2.334	1.322	0.962	0.609	1.037	1.531	1.416	1.453	1.102	1.017	1.152	0.836	2.246	0.794	0.624

(c) The error for each set of data is calculated by $E_i = \mathrm{T}^i - \mathrm{A2}^i$). The sum-squared error, E, is then obtained by

$$E = \sum_{i=1}^{m} (E_i)^2 \tag{7}$$

(d) Generally, the training will stop when either (i) the sum-squared error is smaller than the error goal specified by the user, or (ii) the maximum number of training epochs is

Table 12. The CFAs, actual costs (CostT) and estimated costs (CostE) of testing samples

	Sample #																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
CFA1	0.4	0.6	0.4	0.6	0.6	0.6	0.4	0.4	0.4	0.4	0.6	0.6	0.6	0.4	0.6	0.6	0.4
CFA2	0.42	0.52	0.5	0.54	0.54	0.54	0.42	0.5	0.5	0.44	0.42	0.44	0.52	0.5	0.5	0.5	0.5
CFA3	0.42	0.18	0.5	0.18	0.18	0.18	0.42	0.5	0.5	0.44	0.18	0.18	0.18	0.5	0.18	0.18	0.5
CFA4	0	0.52	0	0.42	0.42	0.42	0	0	0	0	0.42	0.5	0.52	0	0.5	0.5	0
CFA5	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.26
CFA6	0	0.2	0	0.2	0.2	0.2	0	0	0	0	0.2	0.2	0.2	0	0.2	0.2	0
CFA7	0.15	0.1	0.1	0.1	0.1	0.1	0.15	0.1	0.1	0.15	0.1	0.1	0.1	0.1	0.1	0.1	0.1
CFA8	0	0.15	0	0.15	0.15	0.15	0	0	0	0	0.15	0.15	0.15	0	0.15	0.15	0
CFA9	0.036	0.063	0.080	0.048	0.047	0.057	0.051	0.031	0.031	0.044	0.057	0.056	0.046	0.051	0.108	0.060	0.083
CFA10	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0	0.5	0.5	0.5	0.5
CFA11	0.6	0.2	0.2	0.4	0.2	0.6	0.4	0.2	0.2	0.2	0.2	0.2	0	0.2	0.2	0.4	0.2
CFA12	0	0.2	0.25	0.25	0.2	0	0.25	0.25	0.25	0.2	0.25	0.2	0	0.2	0.2	0.2	0.2
CFA13	0	0.075	0.069	0.276	0.191	0	0.166	0.121	0.062	0.057	0.003	0.062	0	0.049	0.059	0.098	0.104
CFA14	0.096	0	0	0.188	0	0.147	0.1	0	0	0	0	0	0	0	0	0.235	0
CFA15	0.2	0.2	0.3	0.4	0.4	0.4	0.3	0.2	0.2	0.2	0.2	0.2	0	0.2	0.2	0.2	0.2
CFA16	0.2	0.2	0.4	0.4	0.4	0.4	0.6	0.6	0.6	0.2	0.2	0.2	0	0.2	0.2	0.4	0.4
CFA17	0	0	0	0.5	0.5	0.5	0	0	0	0	0	0	0	0	0	0	0
CFA18	0	0	0	0.4	0.4	0.4	0	0	0	0	0	0	0	0	0	0	0
CFA19	0	0	0	0.3	0.3	0.3	0	0	0	0	0	0	0	0	0	0	0
CFA20	0.2	0.2	0.2	0.4	0.4	0.4	0.2	0.2	0.2	0.2	0.2	0.2	0	0.4	0.4	0.2	0.2
CFA21	0.05	0.06	0.1	0.09	0.2	0.05	0.08	0.15	0.12	0.05	0.08	0.1	0.14	0.02	0.08	0.2	0.25
CostT	0.701	1.470	1.422	1.489	1.475	1.750	1.016	0.566	0.565	0.955	1.327	1.488	1.011	0.878	2.374	1.384	1.564
CostE	0.779	1.411	1.435	1.628	1.547	1.848	1.059	0.552	0.562	0.881	1.431	1.370	0.980	0.992	2.243	1.294	1.464

reached. However, the target error goal was unknown. We also observed that the sum-squared error tended to converge to a certain value as the number of training epochs increased. Therefore, we used the relative changing rate (RCR) of the sum-squared error to determine when the training has reached a satisfying result. The RCR was calculated at any point where every 1000 training epochs has been completed,

$$\mathrm{RCR}_i = \frac{E_{i\times 1000}}{E_{(i-1)\times 1000}} \tag{8}$$

where:

$E_{i\times 1000}$ = sum-squared error at $(i\times 1000)$ epochs;

$E_{(i-1)\times 1000}$ = sum-squared error at $(i-1)\times 1000)$ epochs, $i = 1, 2, 3, \ldots$.

The target error goal was given a very small value, 0.001, that cannot be reached. The maximum number of epochs is given a very large value, 500 000. This was to ensure that it will go to step (e) to continue training in between the checking points. At the checking point, if $0.99<\mathrm{RCR}_i<1.01$, the training stops; otherwise, go to Step (e).

(e) The derivatives of the error for each neuron layer is calculated and back-propagated to its proceeding layers to alter the weights and biases and go back to Step (b).

In order to train the network quickly and effectively, two improvement measures, namely, *momentum* and *adaptive learning rate*, provided by the MATLAB Neural Network Toolbox [19] have been incorporated into the network. The use of *momentum* can reduce back-propagation sensitivity to small details in the error surface and help the network to avoid getting stuck in shallow minima which would prevent the network from finding a lower error solution. This can be achieved by making weight changes equal to the sum of a fraction of the last weight change and the new change suggested by the back-propagation rule. Large learning rate can make the network learn quickly. However, if the learning rate is too large, the solution search keeps jumping over the error minimum without converging. The use of *adaptive learning rate* is to make the learning rate as a variable which attempts to keep the learning step as large as possible while keeping learning stable. A simplified description of the principle of *adaptive learning rate* is that if the new error exceeds the old error by more than a predefined ratio, the learning rate is decreased; otherwise, the learning rate is increased.

4. *Validation:* the trained network is then validated using a set of samples that have not been used in training to check its generalisation ability. The estimated cost of each validating sample was obtained by inputting its **CFA** to the trained network. The estimated cost relative deviation for each sample was calculated as

$$\mathrm{RD}_i = \frac{\mathrm{CostE}_i - \mathrm{CostT}_i}{\mathrm{CostT}_i} \times 100\% \tag{9}$$

where:

CostT_i = actual cost of sample i;

CostE_i = estimated cost of sample i.

If the relative deviations of some validating samples are more than $\pm 15\%$, there must be important information in these validating samples that the network has not learnt. These samples are then added to the training set. The same number of new samples are added to the validation set. The network is re-trained using this augmented training set and the validating process is repeated.

Initially, 20 samples were used as training set and 17 other samples were used for validation. After a number of rounds of training and validating, the size of training set was increased to 32 when the trained network performs well on the validation set of 17. The final results of the training using those 32 samples and the validation on 17 other samples are summarised and shown in Figs 11 and 12, where the horizontal axis represents the sampling products, and the vertical axis represents the estimated cost error percentage vs. the actual cost of the sampling products. The weights and biases of the trained neural network are shown in Tables 13–15.

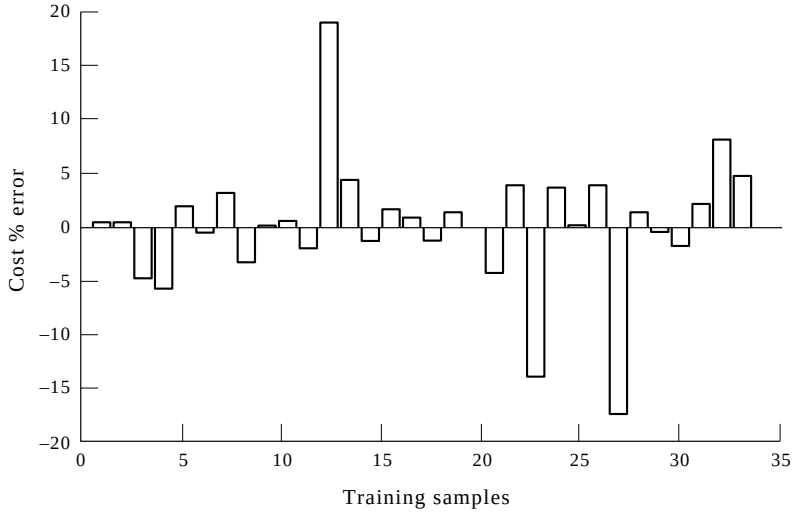


Fig. 11. Estimated costs vs. actual costs for the 32 training samples.

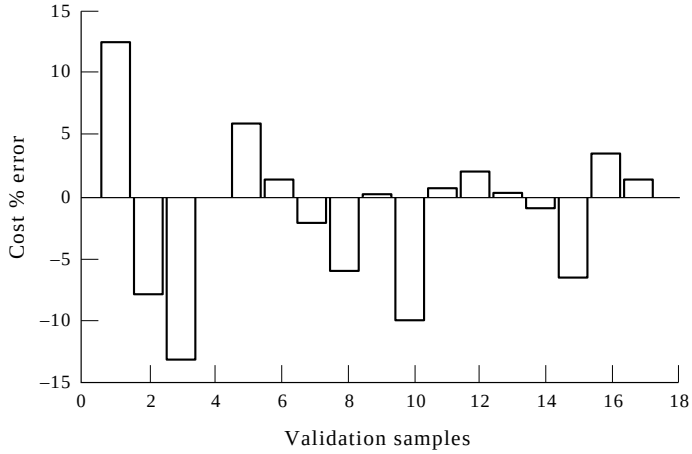


Fig. 12. Estimated costs vs. actual costs for the 17 validation samples.

Table 13. Weights for interconnections between input neurons and hidden neurons

$w1_{1.1}$–$w1_{8.1}$	$w1_{1.2}$–$w1_{8.2}$	$w1_{1.3}$–$w1_{8.3}$	$w1_{1.4}$–$w1_{8.4}$	$w1_{1.5}$–$w1_{8.5}$	$w1_{1.6}$–$w1_{8.6}$	$w1_{1.7}$–$w1_{8.7}$
−0.1088	0.0823	−0.2245	0.1243	−0.2469	−0.0792	−0.0376
0.1039	−0.2536	0.2053	0.0416	−0.2662	−0.2494	0.1050
0.1120	−0.2355	0.1875	−0.0892	−0.0630	0.0519	0.0340
−0.0299	0.2324	0.0079	0.1146	−0.1273	−0.0676	0.1165
0.0254	−0.1304	0.0801	0.2228	−0.2303	−0.3016	0.0402
−0.2769	−0.1065	−0.4636	0.2760	0.0557	0.1519	−0.0564
0.1772	−0.0341	−0.0639	0.0750	0.1976	0.0390	0.2812
−0.0063	0.1130	−0.2791	0.1123	0.1426	−0.2565	−0.1458

$w1_{1.8}$–$w1_{8.8}$	$w1_{1.9}$–$w1_{8.9}$	$w1_{1.10}$–$w1_{8.10}$	$w1_{1.11}$–$w1_{8.11}$	$w1_{1.12}$–$w1_{8.12}$	$w1_{1.13}$–$w1_{8.13}$	$w1_{1.14}$–$w1_{8.14}$
0.0852	−0.8127	−0.0221	−0.3046	0.0529	−0.0664	0.2689
−0.0913	−0.1572	−0.1743	0.2485	0.1260	0.0583	−0.1251
−0.3325	1.5073	−0.4601	−0.1591	−0.0406	−0.1030	−0.1730
−0.2886	−1.9680	0.3093	−0.3246	0.1826	−0.0868	−0.0964
−0.0342	−1.0135	0.1532	0.1639	−0.1873	0.1378	0.1432
0.2536	1.7751	0.3244	−0.4904	−0.1072	0.8919	0.0454
−0.0954	1.0260	0.1011	−0.0145	0.2351	−0.2865	−0.1209
−0.3237	−2.5451	−0.2317	−0.2694	−0.0721	0.5026	0.0229

$w1_{1.15}$–$w1_{8.15}$	$w1_{1.16}$–$w1_{8.16}$	$w1_{1.17}$–$w1_{8.17}$	$w1_{1.18}$–$w1_{8.18}$	$w1_{1.19}$–$w1_{8.19}$	$w1_{1.20}$–$w1_{8.20}$	$w1_{1.21}$–$w1_{8.21}$
0.1011	−0.3197	−0.2227	−0.1544	−0.1606	0.1702	0.1140
−0.0076	−0.1628	0.2658	0.1560	−0.1389	−0.1480	0.0954
−0.2429	−0.2375	−0.4810	0.0874	−0.0576	−0.1542	−0.0660
0.0042	0.1321	−0.0381	0.0062	0.2117	−0.1518	0.0507
−0.2522	−0.1280	0.0711	0.0299	0.1093	−0.2015	0.2483
−0.0274	0.5797	0.1063	0.1967	−0.1031	0.3046	0.4171
−0.1384	−0.2059	0.2286	0.2856	−0.2025	−0.1518	−0.1085
−0.2243	0.2712	0.1270	−0.4816	0.0102	0.1238	0.2047

Table 14. Biases for the hidden neurons

B1(1)	B1(2)	B1(3)	B1(4)	B1(5)	B1(6)	B1(7)	B1(8)
−0.1107	0.1178	−0.1021	−0.0438	−0.1084	−0.0466	0.0546	−0.0737

Table 15. Weights between hidden neurons and the output neuron, and biase for the output neuron

w2(1)	w2(2)	w2(3)	w2(4)	w2(5)	w2(6)	w2(7)	w2(8)	B2(1)
−0.7930	−0.1734	1.5268	−1.9245	−0.9906	1.8851	1.0053	−2.6321	0.1109

TESTING RESULTS AND DISCUSSIONS

In order to test the capability of the trained neural network, 17 other samples are used as the testing set which have not been used in the training and validation session. Their CFAs, actual costs, and estimated costs are shown in Table 12. The error percentages of estimated costs versus actual costs are shown in Fig. 13.

Some observations from the testing results are summarised as follows:

1. Among the 17 testing samples, it can be found that the percentage errors of 15 samples (or 88% of the total samples) are within $\pm 10\%$. This is considered as a good cost estimation at the design stage.
2. The maximum deviation of cost estimate from the actual cost is about 13% which is still considered acceptable by the company, considering this is achieved in the early product design stage.
3. The performance of the trained neural network is consistent in the training, validation, and testing samples.

Since the testing samples were randomly collected from the existing products, it can be concluded that the trained neural network can perform good cost estimation on the RSC products.

Some of the reasons for the deviation between estimated cost and actual cost are:

1. Process plans can still be subjective. This may result in different plans for similar products.
2. The costs of raw materials such as papers and printing inks vary in time. Since the historical cost data werecollected from products that were manufactured in different time, the different raw material cost may affect the final product cost.

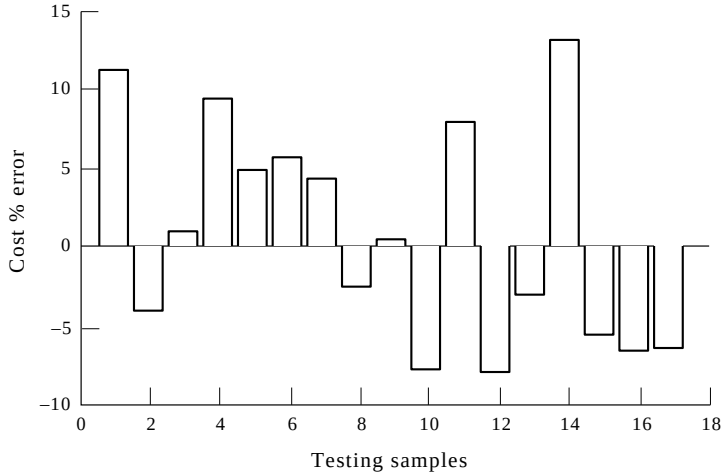


Fig. 13. Estimated costs vs. actual costs for the 17 testing samples.

CONCLUSIONS

The major drawbacks of traditional product cost estimation approaches include the requirement of detailed cost information from production, poor function approximation, and inability to update cost estimation algorithms using actual cost data directly. Although regression analysis approach can use new cost data for updating, the relationship between cost and input parameters may not be the same as the function assumed. By applying the principles of back-propagation neural networks, an algorithm has been presented with a prototype developed for cost estimation of packaging products, which shows promise of reducing, if not eliminating, these drawbacks. The testing results suggest that the neural network approach is able to achieve good product cost estimation based on design specifications. The algorithm involves extracting and quantifying the cost-related features from the product design specifications, and performing cost estimation by inputting the cost-related feature vector to the neural network which is trained based on historical cost data. The advantages and limitations of this approach are summarised below.

Advantages:

1. The cost-related features cover all the cost aspects related to a product design. Extracting such features can be easily done by a product designer. Detailed information on manufacturing processes, processing time, etc. is not required.
2. The approach can deal with complicated products since there is no limit on the number of cost-related features.
3. It is not necessary to know the actual form of cost functions. With the historical cost data, the neural network is capable of learning cost estimation knowledge and performing function approximation.
4. The neural network can be updated by re-training the neural network using new cost data from time to time to cover new cost-related features and/or adjust itself to the new manufacturing environment when engineering practice changes.
5. The neural network can help achieve better conceptual design through evaluating the costs of different design alternatives.

Limitations:

1. The number of neurons in the input layer is the same as the number of cost-related features defined and the output layer has only one neuron which is the cost. However, determining the number of hidden layers and the number of neurons in each hidden layer is a trial-and-error process and often there is no guideline on this, which can be time consuming.
2. Neural network training requires experience.
3. Accurate historical cost data must be available.

REFERENCES

1. Andreasson, M. M. and Olesen, J., The concept of dispositions. *Design,* **1**(1), 1990, 17–36.
2. Jo, H. H., Parsaei, H. R. and Sullivan, W. G., Principles of concurrent engineering. In *Concurrent Engineering—Contemporary Issues and Modern Design Tools*. Chapman & Hall, London, 1993, pp. 3–23.
3. Nichols, K., Getting engineering changes under control. *Journal of Engineering Design,* **1**(1), 1990, 5–16.
4. Beale, R. and Jackson, T., *Neural Computing: An Introduction*. Adam Hilger, Bristol, 1990.
5. Khanna, T., *Foundations of Neural Networks*. Addison-Wesley, Reading, MA, 1990.
6. Ostwald, P. F., *Engineering Cost Estimating*, 3rd edn. Prentice-Hall, Englewood Cliffs, NJ, 1992.
7. Dewhurst, P. and Boothroyd, G., Early cost estimating in product design. *Journal of Manufacturing Systems,* **7**(3), 1988, 183–191.
8. Boothroyd, G. and Reynolds, C., Approximate cost estimates for typical turned products. *Journal of Manufacturing Systems,* **8**(3), 1989, 185–193.
9. Boothroyd, G. and Radovanovic, P., Estimating the cost of machined components during the conceptual design of a product. *Annals of CIRP,* **38**(1), 1989, 157–160.
10. Poli, C., Escudero, J. and Fermandez, R., How part design affects injection-moulding tool costs. *Machine Design*, 1988, November 24, 101–104.

11. Wierda, L. S., Product cost-estimation by the designer. *Engineering Costs and Production Economics,* **13,** 1988, 189–198.
12. Hundal, M. S., Design to cost. In *Concurrent Engineering—Contemporary Issues and Modern Design Tools*. Chapman & Hall, London, 1993, pp. 330–351.
13. Ong, N. S., Activity-based cost tables to support wire harness design. *International Journal of Production Economics,* **29,** 1993, 271–289.
14. Shtub, A. and Zimerman, Y., A neural-network-based approach for estimating the cost of assembly. *International Journal of Production Economics,* **32,** 1993, 189–207.
15. Wu, B., An introduction to neural networks and their applications in manufacturing. *Journal of Intelligent Manufacturing,* **2,** 1992, 391–403.
16. Burke, L. I. and Rangwala, S., Tool condition monitoring in metal cutting: a neural network approach. *Journal of Intelligent Manufacturing,* **2**(5), 1991, 269–280.
17. Chung, Y. and Kusiak, A., Grouping parts with a neural network. *Journal of Manufacturing Systems,* **13**(4), 1994, 262–275.
18. Prabhakar, S. and Henderson, M. R., Automatic form-feature recognition using neural-network-based techniques on boundary representations of solid models. *Computer-Aided Design,* **24**(7), 1992, 381–393.
19. The MathWorks Inc., *Neural Network Toolbox for Use with MATLAB*, 1993.