Polymer property prediction and optimization using neural networks

Nilay K. Roy, Walter D. Potter, and David P. Landau

Abstract -- Prediction and optimization of polymer properties is a complex and highly non-linear problem with no easy method to predict polymer properties directly and accurately. The problem is especially complicated with high molecular weight polymers such as engineering plastics which have the greatest use in industry. The effect of modifying a monomer (polymer repeat unit) on polymerization and the resulting polymer properties is not easy to investigate experimentally given the large number of possible changes. This severely curtails the design of new polymers with specific end-use properties. In this paper we show how properties of modified monomers can be predicted using Neural Networks. We utilize a database of polymer properties and employ a variety of networks ranging from backpropagation networks to unsupervised self-associating maps. We select particular networks that accurately predict specific polymer properties. These networks are classified into groups that range from those that provide quick training to those that provide excellent generalization. We also show how the available polymer database can be used to accurately predict and optimize polymer properties.

Keywords: Neural networks, backpropagation, self-associating, learning rate, momentum, initial weights, polymers, main chain and side chains, pendant groups, steric factors, potentials, affinity, glass-transition, polymerization, modulus.

Manuscript submitted September 28, 2004.

Dr. Roy is a Postdoctoral Research Fellow with the Chemistry Department at Brandeis University, Edison-Lecks Building, Room 94-116, MS-015, PO Box 549110, 415 South St., Waltham MA, 02454-9110, USA. He can be reached via email at nilayr@brandeis.edu, or by telephone at 781-736-2567.

Dr. Potter is Director of the Artificial Intelligence Center, and Professor of Computer Science at the University of Georgia, Athens, GA 30602, USA. (potter@cs.uga.edu).

Dr. Landau is Director of the Center for Simulation Physics, Department of Physics and Astronomy, University of Georgia, Athens, GA 30602, USA.

1. INTRODUCTION

Artificial neural networks (ANNs) are model-free estimators that perform robust multi-dimensional, non-linear vector mappings [1]. The most commonly used ANN is the standard backpropagation network in which every layer is linked or connected to the immediately previous layer. It has been shown that this type of network with at most two hidden layers can solve any non-linear problem provided there are sufficient numbers of hidden nodes [2]. Most polymer modeling requires the handling of highly non-linear problems in which the components are not linearly separable. The relationships between the parameters being modeled and the actual behavior of these variables in the real world must be correlated as precisely as possible. However, in most cases this is not possible and several approximations and simplifications are often made at various stages. When dealing with sparse, noisy or incomplete data, conventional methods (decision theoretic/statistical and syntactic [3]) frequently fail. In addition, conventional methods lack generalization, fail to incorporate statistical and systematic fluctuations, and in most cases are limited to finite state spaces. Thus, the important correlations between the model developed and the real properties may be lost or not captured correctly.

High molecular weight polymer systems represent complex classes of materials and are very difficult to model. Besides being highly non-linear, there are a large number of parameters that need to be accurately defined if such systems are to be properly characterized. Conventional methods can only just begin to model: (1) simple polymer systems such as linear high polymers, copolymers, and their blends, and (2) low molecular weight polymers and copolymers. These methods cannot handle high molecular weight binary and ternary blends, polymer dispersed liquid crystals, ionomers or interpenetrating networks. We have earlier shown that these can be handled using a classic example of polymer blend design [4]. In this earlier paper, we give an overall approach to polymer blend design in which the use of Neural

Networks was but one stage in a process that also made use of Genetic Algorithms and Markov chains to accurately predict blend miscibility. Here we present the first step: the prediction and optimization of modified polymer properties using Neural Networks in much greater depth and detail.

2. DATA SET AND NEURAL NETWORKS

We restrict our studies to modified polymers in which we consider replacing pendant side groups on the side or main chain of the polymer. This is experimentally more feasible and also exploits the information available on existing polymers. One or more of the resulting polymers can be used as parent polymers in studying miscibility and other properties in polymer blend systems.

Of the numerous polymers available probably the most important are the engineering polymers [5]. We concentrate on their properties (mechanical, thermal, magnetic, optical, electrical, environmental and deteriorative) and the relationships with their structures (microscopic, mesoscopic and macroscopic). The prediction of polymer properties from just the structure of the monomer is somewhat unreliable. But trained neural networks that are given optimized input data do an excellent job of characterizing a new modified polymer. This new polymer can then be easily synthesized. We can predict a wide range of properties using this technique, and it is possible to investigate the criteria for successful polymerization and stability such as the heat of polymerization [6].

2.1 Polymer Database

The polymer database we used was selected carefully so as to include all types of polymers available. Within each category, several different types of polymers were included to provide a comprehensive set of data. This is given in Table I. The number of different individual polymers in each category is given by the number in parentheses. Thus, the total data set consisted of 440 individual polymers. For each polymer, information stored included its molecular weight (polydispersity), mechanical and thermal properties, chemical structure, and data reliability number.

TABLE I GOES HERE

Table II gives the description and format of the fields. Fields 15 to 18, and 21 are reported average values from the best four sources.

TABLE II GOES HERE

This is because these values depend on the technique used to find them and the polydispersity of the samples. The last two fields are codes (integers from 1 to 10) that reflect the quality of the data and the extent to which the different blend systems for the given polymer have been characterized, respectively. Numbers greater than five indicate data that are fairly reliable. Numbers less than five indicate data that may be unreliable (due to being out of date, or because the experimental or theoretical techniques had a high source of error). All polymers had data status codes greater than five. The data status codes (DSC) were calculated using

$$DSC = \left\lfloor \frac{1}{4} \sum_{<1,4>} \frac{1}{CV_i} \times 10 \right\rfloor \qquad (1)$$

where

$$CV_i = \frac{1}{4} \sum_{<1,4>} \frac{s_i}{y_i} \times 100 \,. \qquad (2)$$

s_i is the standard deviation, and y_i is the mean value for each of the fields 15 to 18, and 21 for each polymer. CV_i is the average value of the coefficient of variance for each of the five fields 15 to 18 and 21. Note, for each polymer, CV_i was never greater than 0.10. The blend status code (BSC) of each polymer indicates the number of characterized blends having the polymer as one component and a DSC value of at least 5 for the given blend. Every polymer has a BSC of at least one. If a given polymer has more than 10 characterized blends with it as a component and with the DSC value of each of these at least 5, the BSC value is set to 10. Such a polymer clearly has its blends well characterized. The use of these codes ensures that the dataset is reliable and all the polymers in the dataset meet a minimum criterion.

Fields 1 to 14 represent general characteristics. T_α (field 21) is the primary glass-transition temperature at which the onset of long-range segmental mobility occurs [7] while T_β and T_γ with T_δ are lower order relaxation temperatures associated with motions in the back-bone and side-chain, respectively [8]. The aspect ratio [9, 10] is a 2-D measure of the asymmetry in a monomer (polymer repeat unit) and is the ratio of the length of the long axis to the short axis of the monomer. Monomers are three-dimensional objects and hence in addition to their topological and combinatorial content their 3-D character is of profound importance. The 3-D Weiner number is based on the 3-D (geometric, topographic) distance matrix, whose elements represent the shortest Cartesian distance between two $i-j$ pairs. The matrix is real and symmetric. The number is extracted from the matrix as indicated in reference [11]. For engineering plastics, the four transition temperatures together with the melting temperature, the aspect and Weiner numbers, and the elastic modulus with the given polydispersity (weight and number average molecular weight) serve to provide a comprehensive description of the material from the thermal, structural and mechanical stand point. The question of reducing the inputs during the training and final selection of the networks was not considered as all the inputs were important and could not be discarded if theoretical accuracy in the modeling was desired.

2.1.1 Polymer Properties Selected

Impact resistance is very likely the most desired property of an engineering plastic [12]. One indicator of good impact resistance is the T_α / T_γ ratio [13] and the dynamic elastic modulus [14]. The higher this ratio is, the better the impact resistance is. However, high impact resistance with almost no elastic properties results in a brittle polymer that has no commercial use. Therefore, in the complete description of the overall mechanical properties of a polymer these properties must be included. They serve to accurately describe the polymer's mechanical behavior over the entire useful temperature range. While reinforcing a polymer and creating cross-linked or composite materials yields extremely good materials, the optical properties may be sacrificed. Such materials cannot be easily molded and lack any elastic properties, greatly reducing their use and marketability [15]. In this work we concentrate on two engineering plastics and their various modifications. The first, bisphenol-A polycarbonate (PC) [16], is commercially available as Lexan/Makrolon/Calibre and is widely used in bullet-proof glass. It has a T_α / T_γ ratio of 2.5 and a dynamic elastic modulus at $20\,^oC$ of 5.02×10^9 dynes/cm^2. The second, poly(2,6-dimethyl-1,4-phenylene oxide) (PPO) [17], is widely used in the electrical industry as insulation in wiring and armatures, and in capacitors and dielectrics. Its T_α / T_γ ratio is 1.7 and its dynamic elastic modulus at $20\,^oC$ is 6.21×10^9 dynes/cm^2.

From these two polymers 24 different modified PCs and 19 different modified PPOs were selected to present to the final trained networks.

The modifications were made so as to ensure that a range of possibilities was considered. This included replacing the backbone and/or side chain carbon atoms (C) by silicon atoms (Si). As in the case of poly(silicone), it is well known that a silicon substitution not only dramatically improves the dynamic mechanical response of the polymer, but is also a favorable species for polymerization.

Other important substitutions are also considered. The effects of amide-type group (-CO-R-NH) substitutions in the main chain and side chain are important. Poly(amides) like Nylon 6 and Nylon 66 contribute to a major class of commercial materials. They have useful electrical, optical and mechanical properties.

Another important class of polymers is poly(urethanes) (-NH-CO-NH-) based on the amide-type group. These polymers have extensive commercial applications, high impact resistance, and useful high-temperature mechanical properties. They are used as foams, and can be readily cross-linked to form densely packed, yet flexible, rubbers. The amide-type group is highly polar, a distinct advantage, causing favorable intermolecular interactions to dominate. This results in a better van-der-Waals type of weak interactive force between the polymer chains that directly leads to greater dimensional stability. Further, the amide-type groups break the linearity of the polymer chain, resulting in better "excluded volume" types of interactions.

The other types of substitutions were based purely on steric factors. In side chains, the effects of replacing methyl ($-CH_2$) groups with longer linear molecules like ethyl ($-CH_2-CH_3$) or even pentyl ($-CH_2-CH_2-CH_2-CH_2-CH_3$) will result in changes in excluded volume interactions without changing the polarity of the polymer. In many cases an increase in impact resistance is expected.

Besides the amide-type groups mentioned above, substitutions involving oxygen and halogen atoms are important cases that are also considered. A wide array of polymers like poly(esters) and poly(vinyl halides) reflect the importance of these atoms in the main and pendant side chains. For each of the modifications presented to the trained networks, their attractive and repulsive interaction parameters were calculated and checked to see that they were in the range for the given polymer class, thus effectively screening out abstruse structures. Details of this are given in [4]. From over 40 different modifications of PC and PPO presented to the final trained networks several candidates were found with improved properties. While the structures of all these polymers are not given, those with improved properties are presented and discussed in more detail.

2.2 Neural Networks Used

The neural networks used here include the standard supervised backpropagation network with one, two, and three hidden layers; the jump-connected type of network; the recurrent networks with feedback; and networks with two hidden slabs each with their own activation function (see Figure 1). Specialized supervised networks used include the probabilistic type, the general regression type, and the polynomial net. The only type of unsupervised network used was the Kohonen network. Further, eight different activation functions were used:

$f(x) = 1/(1 + e^{-x})$	(standard logistic)	(f_1)
$f(x) = x$	(linear)	(f_2)
$f(x) = \tanh(x)$	(hyperbolic tanh)	(f_3)
$f(x) = \tanh(1.5x)$		(f_4)
$f(x) = \sin x$	(sine)	(f_5)
$f(x) = 2/(1 + e^{-x}) – 1$	(symmetric logistic)	(f_6)
$f(x) = e^{-x^2}$	(Gaussian)	(f_7)

f(x) = 1 - e^{-x^2} (Gaussian compliment) (f_8)

Henceforth we shall refer to these functions as f_1 to f_8, respectively.

2.2.1 Network Types

2.2.1.1 Standard Backpropagation Networks

The first type of network considered was the standard backpropagation network with one, two, and three hidden layers. The advantage of more hidden layers is that different activation functions can be selected. While the most commonly used activation is logistic, in many cases other functions or combinations of functions are known to perform better.

The equation for the output in each layer with *J* nodes is

$$y_j = g(u_j)\,, \tag{3}$$

where $g(u_j)$ is the activation function of u_j given by (f_1)-(f_8) above, and

$$u_j = a_{0j} + \sum_{i=1}^{I} a_{ij} x_i\,. \tag{4}$$

I is the number of inputs x_i, the a_{ij} are weights from input *i* to node *j* in the layer and a_{0j} is the bias weight of node *j*.

The output of each layer is fed into the next layer. In the backpropagation algorithm the weights are adjusted in the backpropagation stage so that the error, given by

$$E = \frac{\frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{K}(z_{kn} - t_{kn})^2}{NK}\,, \tag{5}$$

is minimized. *N* is the number of examples in the data set, *K* is the number of outputs of the network, t_{kn} is the *k*th target output for the *n*th example, and z_{kn} is the *k*th actual output for the *n*th example.

For each layer the error derivative is

$$\frac{\partial E}{\partial a} = \frac{\partial E}{\partial y}\cdot\frac{\partial y}{\partial u}\cdot\frac{\partial u}{\partial a}\,, \tag{6}$$

where

$$\frac{\partial E}{\partial y} = \sum_{k=1}^{K}\frac{\partial E}{\partial z_k}\cdot\frac{\partial z_k}{\partial v_k}\cdot\frac{\partial v_k}{\partial y}\,. \tag{7}$$

Subscript *j* is omitted in (7) as one node is involved. The first and second terms in the product in (7) are the change in the network's error with respect to the next layers output node's output ($\frac{\partial E}{\partial z_k}$) and the change in the next layer's output node's output with respect to the weighted sum of the next layer's input ($\frac{\partial z_k}{\partial v_k}$). Equations for the next layer into which the present layer's input goes are generalized from (3) and (4) by a change of variables. Now (7) can be rewritten as

$$\frac{\partial E}{\partial y} = \sum_{k=1}^{K} p_k \frac{\partial v_k}{\partial y} \tag{8}$$

where p_k is the quantity that is being propagated from the nodes in the next layer to the nodes in the current layer. If we set $\frac{\partial v_k}{\partial y} = b_k$ then (8) takes the form

$$\frac{\partial E}{\partial y} = \sum_{k=1}^{K} p_k b_k\,. \tag{9}$$

The second term in the product in (6) depends on the form of the activation function (f_1-f_8). For the case of f_1 this is

$$\frac{\partial y}{\partial u} = y(1-y)\,. \tag{10}$$

Let

$$q = \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial u}, \qquad (11)$$

then

$$q = \left[\sum_{k=1}^{K} p_k b_k \right] y(1-y). \qquad (12)$$

If $\frac{\partial u}{\partial a_0} = x_{i0}$ for the bias weight and $\frac{\partial u}{\partial a_i} = x_i$, we have

$$\frac{\partial E}{\partial a_i} = \begin{cases} q x_{i0} & \text{for a bias weight} \\ q x_i & \text{for an input weight} \end{cases}. \qquad (13)$$

Let

$$d_m = \sum_{n=1}^{N} \left(\frac{\partial E}{\partial a_m} \right)_n, \qquad (14)$$

then the weight change for the next epoch *m+1* with respect to the current epoch *m* is given by

$$\partial w_{m+1,m} = w_{m+1} - w_m = c_m, \qquad (15)$$

and

$$c_m = \mu \cdot c_{m-1} - (1-\mu) \cdot \varepsilon \cdot d_m. \qquad (16)$$

μ is the momentum and ε is the learning rate with values in the range 0 to 1.

From (3) and (14) we see now that the standard backpropagation algorithm involves: (a) a forward pass using the forward equation (3) in which the outputs for each layer are calculated, and (b) a backward error propagation pass using the backward equation (14) in which the error is minimized and the weights adjusted.

The second type of network used was the jump connected network [18]. This is a backpropagation network in which every layer is connected or linked to every previous layer. As shown in Figure 1(a)-(c) there is a choice of one to three hidden layers. They are similar to the standard backpropagation network except that (4) will have an additional term of the type

$$\sum_{j=J+1}^{J+I} b_{jk} x_{j-J}, \qquad (17)$$

where the index *j* runs over the links to the current node in the given layer from the nodes directly (jump connected) from the previous layer. x_{j-J} are the *I* inputs whose links to the current layer node have index values running from *J+1* to *J+I*.

The disadvantage of using more than one hidden layer is an increase in training time, but in certain cases better models may be obtained.

The third type of backpropagation network used is a recurrent network [19] (Figure 1(d)-(f)). The three types of recurrent networks used are: (i) in which the input layer is fed back into the input layer itself, (ii) in which the hidden layer is fed back into the input layer, and (iii) in which the output layer is fed back into the input layer. They are most successful when there is a time series in the input data. In the first type, the long-term memory remembers the new input data and uses it when the next pattern is processed. In the second type, the long-term memory remembers the hidden layer, which contains features detected in the raw data or previous patterns. In the last type, long-term memory remembers outputs previously predicted. If there is no temporal structure in the data it may not work.

A recurrent network may respond to the same input pattern differently at different times, depending on the patterns that have previously been presented as inputs. These networks are trained the same way as the standard backpropagation networks except that patterns must always be presented in the same order. During the training it would be interesting to see the effect of randomly presenting the training data (as in the standard backpropagation network) versus grouping the data into the sets according to polymer classes and groups and presenting the ordered data sequentially to the network (as in recurrent networks).

The feedback link contains a feedback factor that needs to be adjusted carefully. In case of recurrent networks the input-output relationship is

$$y(n) = \mu_1 y(n-1) + \mu_2 x(n)\,, \tag{18}$$

where *y(n)* is the output of the given layer at epoch *n* and *x(n)* is the input. The feedback for epoch *n-1* is *y(n-1)*. This can be from the input layer (Elman network in Figure 1(d)), the hidden layer (Jordan-Elman network in Figure 1(e)), or the output layer (Jordan network in Figure 1(f)). From (18) we see

$$\frac{\partial y(n)}{\partial \mu_2} = x(n)\,, \tag{19}$$

and

$$\frac{\partial y(n)}{\partial \mu_1} = y(n-1) + \mu_1 \frac{\partial y(n-1)}{\partial \mu_1}\,. \tag{20}$$

The first term in (19) is like (20) but the second term in (20) depends on μ_1 recursively. Thus, μ_1 is the factor “feedback 1” that goes into long term memory and μ_2 is the factor “feedback 2”. The factor “feedback 1” indicates what proportion of the neuron values goes into long term memory, and “feedback 2” indicates what proportion of the neuron values in the current pattern from either the input, hidden or output layer (depending on the network type) are fed into the long term memory. Both of these values must add up to one. If more emphasis is to be placed on historical patterns, then a higher proportion of neuron values are put on feedback 1. If more emphasis is to be placed on recent patterns then feedback 2 is set greater than 0.5. The same static error criteria as in the case of standard backpropagation can be used as fixed point learning is carried out.

The last backpropagation network used was one with multiple hidden layers (Figure 1(g)-(i)). The hidden layers act like feature detectors. Different activation functions applied to the hidden layer detect different features as a pattern processes through a network. A Gaussian function (f_7) in one hidden slab detects features in the mid-range of the data while the Gaussian compliment (f_8) in another hidden slab detects features in the upper and lower extremes of the data. Combining the results leads to better prediction. When each slab has different activation functions it offers three ways of viewing data. The output layer receives two different views of the data's features as detected in the hidden slabs plus the original inputs. Data that have their multi-dimensional distribution surfaces uniform or flat with small local fluctuations are ideally suited for these networks. The equations are similar to the standard backpropagation case with jump connections and can be easily generalized. For each layer there is an equal division of the outputs/inputs if there are two slabs in a given layer.

2.2.1.2 Probabilistic Networks

In addition to the above backpropagation networks, four other networks not based on the backpropagation algorithm were used. The first of these was the probabilistic network [20,21] (PNN). PNNs are known for their ability to train quickly on sparse data sets. A PNN separates data into a specified number of output categories. They are three layer networks wherein the training patterns are presented to the input layer and the output layer has one neuron for each possible category. The network produces activations in the output layer corresponding to the probability density function estimator for that category. The highest output represents the probable category. Two main calibration techniques are used here: iterative and genetic adaptive.

2.2.1.3 General Regression Neural Network

Another network we use that is able to train quickly on sparse data sets is the General Regression Neural Network [22,23] (GRNN). As with a PNN, the calibration criteria for a GRNN is the same: iterative and genetic adaptive.

2.2.1.4 Group Method of Data Handling Neural Network

This is the Polynomial network, also known as the Group Method of Data Handling (GMDH) Network [24,25]. A by-product of GMDH is that it recognizes (and can present to the user) the most significant variables as it trains.

2.2.1.5 Kohonen Network

The last network used is the unsupervised Kohonen network [26]. The pattern presentation is based on rotation where each pattern is applied to the network one at a time.

2.2.2 Training Method

The most important criterion for successful network training and optimization is accurate generalization. Care must be taken to prevent the memorization of input data. An evaluation set is created either from the training data or a separate set of data with known outputs. This is then used to check the accuracy of the trained network. The process of selecting the final network, and the training and optimization involved, covers a number of steps. For various values of momentum, learning rate, initial weight distribution, number of nodes in the various layers, and the number of layers itself, a significant amount of trial and error is involved to test every type of network for the best combination of parameters. The training set must be varied and contain the entire spectrum for the given problem to be represented. To stop training, the correlation coefficient and R^2 coefficient are monitored. Training stops when they are within predefined values. The correlation coefficient "*r*" is a statistical measure of the strength of the relationship between the actual versus predicted results. It ranges from -1 to +1. The closer *r* is to 1, the stronger the positive linear relationship, and the closer *r* is to -1, the stronger the negative linear relationship. When *r* is near 0 there is no linear relationship.

$r = \frac{\sigma_{xy}}{\sigma_x \sigma_y}$, where $\sigma_{xy} = \sum xy - \frac{(\sum x)(\sum y)}{n}$, $\sigma_{xx} = \sum x^2 - \frac{(\sum x)^2}{n}$, and $\sigma_{yy} = \sum y^2 - \frac{(\sum y)^2}{n}$. n equals number of patterns, *x* refers to the set of actual outputs, and *y* refers to the predicted outputs.

The second coefficient, R^2, is a statistical indicator usually applied to multiple regression analysis. It compares the accuracy of the model to the accuracy of a trivial benchmark model wherein the prediction is just the mean of all of the samples. A perfect fit would result in an R^2 value of 1, a very good fit near 1, and a very poor fit less than 0. If the neural model predictions are worse than predictions obtained by just using the mean of the sample case outputs, the R^2 value will be less than 0. $R^2 = 1 - \frac{\sigma_e}{\sigma_m}$, where $\sigma_e = \sum (y - \tilde{y})^2$ and $\sigma_m = \sum (y - \bar{y})^2$. $\tilde{y}$ is the predicted value of the actual value *y*, and $\bar{y}$ is the mean of the *y* values.

Other parameters monitored are the percentage of the network answers that are within 5%, 10%, 20%, 30%, and over 30% of the actual answers used to train the networks. The mean squared error, mean absolute error, minimum absolute error, and maximum absolute error are also tracked. For each type of neural network there is much experimentation needed to find the best network. Then, from among the different network types that can be used we have to select the one that is best for the given problem.

The inputs for training the backpropagation networks and the GRNN network were taken from the polymer database shown in Table II as fields numbered 6 to 9, 11 to 18, 20, 26, and 27. The number of neurons in the input layer was the number of inputs in every case. The outputs for training of these types of nets were fields numbered 25 and 28, and were the number of neurons in the output layer. In all, the original pattern file had 440 patterns. 80% were extracted at random to form the training set (352 patterns), and 20% were used in the test set (88 patterns). It was found that the multi-layer backpropagation networks gave the best results as compared to the other types of networks considered (PNN, GRNN, GMDH and Kohonen). In the case of the PNN and Kohonen networks, the inputs were the same as above, however there was no need for outputs. After training and testing, the inputs for the 24 modified PCs and 19 modified PPOs were presented to the final trained PNN network. The network classified each of these into one of the 440 polymers. Thus PNN and Kohonen networks provided only a rough estimate of the mechanical properties of the modified polymers. However, this was a tremendous help in verifying the viability of a particular modified polymer before being presented to the other networks. The modified PCs and PPOs were finally selected only if these nets successfully classified them in their polymer groups. The number of inputs and the number of neurons in the input layer of the GMDH network were the same as in the case of PNN and GRNN nets. However, the number of outputs was always one. The first output was the field numbered 25 followed by 28. Thus, two final GMDH networks were created to which the modified polymers were presented.

3. RESULTS AND DISCUSSIONS

As shown in Figure 1 and discussed above, there are several possible networks. In the end, one network with a given set of parameters must be selected to make the predictions. This involves a considerable amount of training with all network types for all possible parameter value combinations. Another method would be to take the predictions from the various best nets found for each type of network and average the values. However, in this case, there is a drawback in that the averaging process can actually lead to the deterioration of the final predicted values. As discussed in [4], polymer blend miscibility is very sensitive to the interaction parameters. These must be predicted extremely accurately for predicting miscibility. We have seen that the ratio T_α / T_γ is extremely sensitive and important to predicting good mechanical properties. Thus we do not average our predictions of this value over other network predictions. We select one network from a single network architecture that we use for our final predictions. For the sake of comparison we also present results from the other network architectures. The main advantage of this comparison is that it enables us to select networks that train quickly and provide acceptable results as compared to the best network we found. Further, some network architectures like the Kohonen and PNN networks serve a useful purpose in deciding which modified polymers to investigate further.

3.1 Final Networks

The best network found for this problem was the multi-layer backpropagation network. The bottom-up technique for selecting the number of hidden nodes was used in favor of the top-down method. The number of calibration events was 50 and training was saved based on the best test set. The training ended when the number of events since the minimum average error exceeded 500,000. As indicated by equations (14) to (16), weight updates were dictated by the learning rate, momentum, and a portion of the previous weight changes. Pattern selection was random during training.

Figure 2 shows the effects of varying the number of hidden nodes for a single hidden layer backpropagation network. From this we see that the mean squared error and the correlation coefficient were best for 17 hidden nodes. A higher number of hidden nodes did not appear to improve these values further. Thus the number of hidden nodes was fixed at 17. In this region, the 'Percent above 30%' (of the actual values) also shows a minimum while the correlation coefficient exhibited a maximum. In Figure 3 the results of varying the initial weights are given, while in Figure 4 the effects of varying momentum are shown. Figure 5 shows the effect of varying the learning rate. The activation functions f_1, f_3, f_4 and f_6 were tried. f_1 gave the best results.

The addition of a second hidden layer did not improve the results. For the 'Percent within 5%' and the 'Percent over 30%' values, the case when the number of nodes was equal to 17 (N_{17}) gave better results than the case when the number of nodes was equal to 100 (N_{100}). The N_{100} case gave marginally better R^2, r^2, and correlation coefficient values. The N_{17} case also had the lowest mean squared error and mean absolute error. Given such narrow differences made making choices difficult, but finally considering the fact that additional nodes may cause redundancy in many weights, and that the N_{17} case did have the lowest mean square error, mean absolute error and better 'Percent within 5%' and 'Percent over 30%' values as compared to the N_{100} case, it was selected. It also had the second highest value for the correlation coefficient. Note that the graphs in Figure 2 tended to flatten out after 15 nodes, suggesting some statistical fluctuations when using more than 15 nodes. This may indicate that the (near) optimum number of nodes for accurately defining the given problem was reached. Genetic algorithms or other combinatorial optimization methods could also be effectively used, but in terms of simplicity this was the better approach. Figure 3 shows the results obtained when varying the initial weights from 0.3. The number of hidden nodes was fixed at 17. All other parameters including the learning rate and momentum were the same as in the case above. As compared to the case of 0.3 for 17 nodes in Figure 2, all the values for correlation coefficient, r^2, and mean squared error are not as good. Figure 4 gives the results of varying the momentum with learning rate fixed. All other parameters were the same as in the first case with the number of hidden nodes fixed at 17. Momentum of 0.4 gave only a marginally poorer value of correlation coefficient, r^2, and mean squared error as compared to a momentum of 0.5. Lower or higher values of momentum gave poor results.

Figure 5 gives the results of varying learning rate keeping momentum fixed at 0.5. The number of hidden nodes was 17, and all other parameters were the same as in the first case. Increasing the learning rate caused further deterioration in the values of correlation coefficient, r^2, and mean squared error, from gradual to more rapid as the learning rate increased.

Finally, adding a second hidden layer to the network did not show any improvements as can be seen from Table III. All parameters are the same as in the first case. The number of hidden nodes in the first hidden layer is 17. While increasing the number of hidden nodes in the second layer did improve the results, they were in all cases not as good as the final selected model. Figure 3(a) and Figure 3(d) show the reason an initial weight of 0.3 was chosen in the final model. This value gave the smallest mean squared error (Figure 3(c)), and highest 'Percent $< 5\%$' values (Figure 3(b)). For momentum of 0.5, Figure 4(a) shows a maxima, and the mean squared error was minimal (Figure 4(c)). For this value, as seen in Figure 4(d), the graph flattened out. From Figure 4(b), 'Percent $< 5\%$' gave a maximum with a corresponding minimum for 'Percent $> 30\%$'. This value appeared to be a (near) optimal number and was selected in the final model.

Figure 5 shows that a choice of learning rate of 0.05 leads to the deterioration of the results with an increase in learning rate. The details of the final selected model are given in Table IV. The min/max values of each of the input variables were set by scanning the input before being applied to the network. Pattern extraction was random. Figure 6 gives a scatter plot of the actual versus predicted output when the trained network is applied to the production file with 88 patterns randomly selected from the 440 patterns in the database where the exact values are known and can be compared with the predicted values from the final model. Note that this set differed from the 88 pattern testing set.

TABLE III GOES HERE

In Figure 6, points on the straight line indicate that the actual and predicted output were identical. If most of the points lie on this line, there is a danger of memorization and lack of generalization by the network. Similarly, wide deviations from the straight line indicate a poorly trained network that may not give accurate predictions.

Tables V to VIII give the details of the best nets for the other network architectures. While the final predictions using these were not as good as the case when the final selected model was used (Table IV), several important points were noted. In the case of the jump connection nets (Table V), the training time increased as the number of hidden layers increased. The number of nodes in each hidden layer also increased. This indicates that jump connections added a degree of complexity that was not particularly useful.

In the case of the recurrent networks (Table VI), the final network shown in Table VI(c) was the network that trained the fastest compared to the rest of the networks used. The predictions using this network however were no better than those for the final selected model (Table IV). Table VII gives the details of the final nets with two hidden layers with the possibility of using different activation functions in the two hidden layers. A typical choice of activation function would be one of the Gaussian functions, f_7 or f_8. However the final model did not find any hidden features or improve the final predictions.

In Table VIII, the best GRNN and GMDH networks found are listed. Both of these networks failed to give better results than the final best model. Training times were as high as in the case of jump connected networks. Note that all training times given for the various networks are rounded to the nearest 1000 epochs.

TABLE IV GOES HERE

TABLE V GOES HERE

TABLE VI GOES HERE

TABLE VII GOES HERE

TABLE VIII GOES HERE

An epoch is one complete training cycle for a given set of inputs until the next set of weight adjustments (backpropagation networks), or coefficient adjustments (GRNN and GMDH networks). To compare the different networks and select the final model, the 88 pattern testing set was used. Training time was not a concern here as accuracy in prediction was more important. As is shown, the best network was the final model which gave slightly better values for all the statistical parameters considered as compared to the other networks (Figure 7). Although one or two statistical parameters of the other models were slightly better, the selected final model was consistently better, overall.

3.2 Predicted Properties

Table IX gives the results when the final selected model is used. Figure 8 gives their structures. We found that for these modifications better T_α / T_γ ratios and dynamic mechanical modulus values were predicted as compared to the parent polymers. For the other cases both the values were not as good as for the parent. On the basis of these results, we were able to conclude that both steric factors, and the intra- and inter-molecular polarities of the polymer play a vital role in the final outcome of the prediction of the mechanical properties of the polymers tested.

TABLE IX GOES HERE

From steric considerations, we were able to conclude that symmetry and slight increases in the excluded volume improved the mechanical properties. Modification PC-5 with symmetric butyl groups ($-CH_2-CH_2-CH_2-CH_3$) gave an improvement, but when these groups were pentyl or higher there was a drastic loss in the dynamic mechanical modulus. Similarly removing one such group had a detrimental effect on the mechanical properties.

We also predicted better mechanical properties when silicon, Si, substitutions were made, as can be seen in the case of modification PC-2, modification PC-3, and modification PPO-1. Silicon, like carbon, exhibits sp^3 hybridization (both have similar ionization potentials). However, silicon has a bigger atomic radius and hence larger bonding and anti-bonding orbitals, resulting in larger bond lengths. This appears to increase the excluded volume interactions, which directly enhances the weak van-der-Waals type attractive forces between the polymers. As a result, we see a better mechanical response. There does appear to be a critical limit beyond which increases in the excluded volume actually begin to have a reverse effect.

The addition of oxygen, or halide atoms on main or pendant side groups, did not result in better mechanical properties. The strong dependence of good mechanical properties due to Si substitutions is established from the results of modification PC-1, 2, 3, and 5 and modification PPO-1. Also we have found that substitutions involving amino-type groups with the constraint that they are symmetric on the backbone (modification PC-4, and modification PPO-2) favor an increase in the mechanical properties as compared to the parents. In fact the polarities and the resulting charge distributions of the mer appear to be more important than steric factors. This again reiterates the fact that polymer dynamics are mainly characterized by inter-molecular interactions like the weak van-der-Waals type of attractive forces. Steric factors, though important, appear to be secondary factors in polar polymers like PC and PPO. This is one reason why we now use these modifications [4] where blend miscibility is governed by intra- and inter-molecular interactions between the polymer species through the models on blending.

4. CONCLUSIONS

We show how neural networks can be successfully used to predict polymer properties. We have demonstrated that the complex structure-property relationships in polymers can be captured effectively by neural networks. In addition, we can use networks to fine tune polymers to exhibit desired properties. While Kohonen and PNN networks provide a general indication of the feasibility of a given polymer modification, specialized networks can be built with a high degree of accuracy or via quick training to predict polymer properties. Further the large amount of available information on polymers can be used. The study of the detailed complex polynomials and the coefficients built by GRNN and GMDH networks help to quantify the structure-property relationships between the different classes of polymers. These can easily be compared to theoretical models built such as the Takayanagi model for the mechanical properties of polymers [9], and is the subject of further investigation. Finally, as shown in [4], we are not restricted to using this method exclusively but can complement other techniques in very useful ways. Also, we are not restricted to a particular set of parameters, but can study other areas of polymer design including polymerization kinetics, stability and degradation to name just a few.

ACKNOWLEDGEMENT: The authors wish to thank the editor, the reviewers, and Mr. Brian A. Smith for their helpful comments and suggestions for improving the presentation of this paper.

REFERENCES

[1] D. Muller and J. Reinhardt, *Neural Networks: An Introduction,* Springer-Verlag, New York, 1990.

[2] S. Haykin, *Neural Networks: A Comprehensive Foundation,* Macmillan College Publishing Company, New York, 1994.

[3] D. W. Heerman, *Computer Simulation Methods in Theoretical Physics,* Springer-Verlag, Heidelberg, 1986.

[4] N. K. Roy, D. P. Landau, and W. D. Potter, "Designing Polymer Blends Using Neural Networks, Genetic Algorithms and Markov Chains," *Applied Intelligence,* 20(3)**,** 2004, pp. 215-229.

[5] M. V. Volkenstein, *Configurational Statistics of Polymer Chains,* Wiley (Interscience), New York, 1963.

[6] G. Odian, *Principles of Polymerization,* McGraw-Hill, New York, 1970.

[7] L. H. Peebles, *Molecular Weight Distribution in Polymers,* Interscience, New York, 1971.

[8] R. M. Ogorkiewicz, *Engineering Properties of Thermoplastics,* John Wiley and Sons, New York, 1970.

[9] D. W. van Krevelen and P. J. Hoftyzer, *Properties of Polymers: Their Estimation and Correlation with Chemical Structure*, Elsevier Science Publishers, Amsterdam, 1976.

[10] J. M. Schultz, *Polymer Materials Science,* Prentice Hall, Englewood Cliffs, New Jersey, 1974.

[11] N. Trinajstic, *Chemical Graph Theory,* CRC Press, London, 1992.

[12] I. M. Ward, *Mechanical Properties of Solid Polymers,* Wiley-Interscience, New York, 1989.

[13] I. E. Neilsen, *Mechanical Properties of Polymers,* Reinhold, New York, 1991.

[14] T. Murayama, *Dynamic Mechanical Analysis of Polymeric Materials,* Elsevier, New York, 1978.

[15] E. E. Bear, *Engineering Design for Plastics,* Reinhold, New York, 1964.

[16] H. Schnell, *Chemistry and Physics of Polycarbonates,* Wiley, New York, 1964.

[17] J. J. Heijbouer, "Dynamic Mechanical Properties and Impact Strength," *Polym.Sci.Polym.Symp.,* C 16, 1968, pp. 3755-3763.

[18] P. Wasserman, *Neural Computing, Theory and Practice,* Van Nostrand Reinhold, New York, 1989.

[19] D. P. Mandic and J. A. Chambers, *Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability,* John Wiley & Sons, Inc., New York, 2001.

[20] D. Specht, "Probabilistic Neural Networks for Classification, Mapping and Associative Memory," *Proceedings of the IEEE International Conference on Neural Networks,* 1, 1988, pp. 525-532.

[21] L. Rutkowski, "Adaptive Probabilistic Neural Networks for Pattern Classification in Time-varying Environment," *IEEE Trans. on Neural Networks,* 15, 4, 2004, pp. 811-827.

[22] D. Specht, "A General Regression Neural Network," *IEEE Trans. on Neural Networks,* 2, 6, 1991, pp. 568-576.

[23] L. Rutkowski, "Generalized Regression Neural Networks in Time-Varying Environment," *IEEE Trans. on Neural Networks,* 15, 3, 2004, pp. 576-596.

[24] E. S. J. Farlow, *Statistics:Textbooks and Monographs*, 54, 1984.

[25] N. Y. Nikaloev and H. Iba, "Learning Polynomial Feedforward Neural Networks by Genetic Programming and Backpropagation", *IEEE Trans. on Neural Networks,* 14, ,2 2003, pp. 337-350.

[26] M. Caudill, "Neural Network Primer," *AI Expert*, 25, 1990.

TABLE I: List of the 440 polymers in the polymer database.

1.Main-chain Acyclic Carbon Polymers
Poly(dienes) (4)
Poly(alkenes) (7)
Poly(acrylic acid) (11)
Poly(acrylic acid ester) (9)
Poly(acrylamides) (2)
Poly(methacrylic acid) (4)
Poly(methacrylic acid ester) (4)
Poly(methacrylamides) (2)
Poly(vinyl ether) (2)
Poly(vinyl thioether) (1)
Poly(vinyl alcohol) (5)
Poly(vinyl ketones) (2)
Poly(vinyl halides) (5)
Poly(vinyl nitriles) (5)
Poly(vinyl esters) (25)
Poly(styrenes) (48)
2.Main-chain Carbocyclic Polymers
Poly(phenylenes) (19)
3.Main-chain Acyclic Hetroatom Polymers
3.1)Main-chain -C-O-C Polymer
Poly(oxides) (22)
Poly(carbonates) (46)
Poly((esters) (70)
Poly(anhydirdes) (1)
Poly(urethanes) (32)
3.2)Main-chain O-Heteroatom Polymers
Poly((sulphonates) (1)
Poly(siloxanes) (15)
3.3)Main-chain -C-(S)-C- and -C-S-N- Polymers
Poly(sulphides) (8)
Poly((thioesters) (1)
Poly(sulphones) (2)
Poly(sulphonamides) (2)
3.4)Main-chain -C-N-C Polymers
Poly(amides) (63)
Poly(imides) (3)
Poly(ureas) (4)
Poly(ureas) (4)
Poly(phosphazenes) (4)
Poly(silanes) (1)
Poly(silazanes) (1)
4.Main-chain Hetrocyclic Polymers
Poly(acetals) (3)
Poly(carboranes) (2)
Poly(piperazines) (2)
Poly(oxadiazoles) (2)

TABLE II: Description of the fields in the database for each polymer.

No	Field Name	Data Type	Description
1	ID	Auto No	Primary Key
2	Polymer_Class	Text	Class of Polymer
3	Polymer_Name	Text	IUPAC Polymer Name
4	Mer_Chemical_Formula	Text	Chemical Formula of Mer
5	No_C_atoms	Number	No of C Atom Bonds in Mer
6	No_C-C_bonds	Number	No of C-C Single Bonds in Mer
7	No_C-H_bonds	Number	No of C-H Single Bonds in Mer
8	No_C-2-C_bonds	Number	No of C-C Double Bonds in Mer
9	No_C-3-C_bonds	Number	No of C-C Triple Bonds in Mer
10	No_H_atoms	Number	No of H Atom Bonds in Mer
11	No_Si_atoms	Number	No of Si Atom Bonds in Mer
12	No_Halide_atoms	Number	No of Halide Atom Bonds in Mer
13	No_P_atoms	Number	No of P Atom Bonds in Mer
14	No_N_atoms	Number	No of N Atom Bonds in Mer
15	No_O_atoms	Number	No of O Atom Bonds in Mer
16	No_S_atoms	Number	No of S Atom Bonds in Mer
17	No_TransitionMetal_atoms	Number	No of Transition Metal Atom Bonds in Mer
18	No_Cyclic_Rings	Number	No of Cyclic Rings in Mer
19	Mol_Wt_Num_Avg	Number	No average molecular weight
20	Mol_Wt_Wt_Avg	Number	Weight average mol weight
21	T_alpha	Number	T_α (^oK)
22	T_beta	Number	T_β (^oK)
23	T_gamma	Number	T_γ (^oK)
24	T_delta	Number	T_δ (^oK)
25	T_alpha_T_gamma	Number	T_α / T_γ
26	Mer_Aspect_Ratio	Number	Aspect Ratio of Mer
27	Mer_3d_Weiner_Number	Number	3-D Weiner Number of Mer
28	Dynamic_Modulus	Number	Dynamic Modulus (20^oC, dynes/cm^2)
29	T_m	Number	Melting Point (^oK)
30	Data_Status_Code	Number	Data Status Code
31	Blend_Status_Code	Number	Blend Status Code

TABLE III: Effect of adding a second layer to the standard backpropagation single hidden layer neural network. All activation functions are logistic (f_1). Input scaling function is logistic (f_1). Number of hidden nodes in first hidden layer = 17. Learning Rate = 0.05. Momentum = 0.5. Initial Weights = 0.3.

Second Hidden Layer Nodes	5	10	15
R^2	0.854	0.919	0.893
r^2	0.913	0.922	0.919
Mean Squared Error	0.072	0.068	0.069
Mean Absolute Error	0.285	0.226	0.225
Min. Absolute Error	0.008	0.007	0.007
Max. Absolute Error	0.300	0.285	0.289
Correlation coefficient	0.924	0.936	0.939
Percent within 5%	32.56	34.58	34.74
Percent within 5% to 10%	32.27	33.04	32.95
Percent within 10% to 20%	17.54	18.59	19.01
Percent within 20% to 30%	8.32	7.38	6.39
Percent over 30%	9.31	6.41	6.91
Training Time (epochs)	3,478,000	3,965,000	4,219,000

TABLE IV: Details of final selected model.

Type of Neural Network	Standard Backpropagation
Number of Hidden Layers	1
Number of input nodes	14
Number of hidden nodes	17
Number of output nodes	2
Activation function	Logistic (f_1)
Scaling function	Logistic (f_1)
Weight updates	Momentum
Pattern Selection	Random
Initial Weights	0.3
Learning Rate	0.05
Momentum	0.5
Percent over 30%	9.31
Training Time	1,573,000 epochs

TABLE V: Details of best nets for the different neural network architectures. (a) Three layers with a jump connection. (b) Four layers with jump connections. (c) Five layers with jump connections.

a)	
Type	Three layers - jump connection
Hidden Layers	1
Input nodes	14
Hidden nodes	86
Output nodes	2
Activation function	Logistic (f_1)
Scaling function	Logistic (f_1)
Weight updates	Momentum
Pattern Selection	Random
Initial Weights	0.1
Learning Rate	0.025
Momentum	0.2
Training Time	4,678,000 epochs
b)	
Type	Four Layers - jump connection
Hidden Layers	2
Input nodes	14
First hidden layer nodes	86
Second hidden layer nodes	138
Output nodes	2
Activation function	Logistic (f_1)
Scaling function	Logistic (f_1)
Weight updates	Momentum
Pattern Selection	Random
Initial Weights	0.15
Learning Rate	0.01
Momentum	0.3
Training Time	5,346,000 epochs
c)	
Type	Four Layers - jump connection
Hidden Layers	2
Input nodes	14
First hidden layer nodes	86
Second hidden layer nodes	138
Third hidden layer nodes	215
Output nodes	2
Activation function	Logistic (f_1)
Scaling function	Logistic (f_1)
Weight updates	Momentum
Pattern Selection	Random
Initial Weights	0.2
Learning Rate	0.005
Momentum	0.4
Training Time	12,387,000epochs

TABLE VI: Details of best nets for the different neural network architectures. (a) Recurrent Net with input layer dampened feedback. (b) Recurrent Net with hidden layer dampened feedback. (c) Recurrent Net with output layer dampened feedback.

a)	
Type	Recurrent Net (Input layer feedback to input layer)
Hidden Layers	2
Input nodes	14
First hidden layer nodes	40
Second hidden layer nodes	40
Output nodes	2
Activation function	Logistic (f_1)
Scaling function	Logistic (f_1)
Feedback factor	0.5
Weight updates	Momentum
Pattern Selection	Random
Initial Weights	0.1
Learning Rate	0.05
Momentum	0.2
Training Time	967,000 epochs
b)	
Type	Recurrent Net (Hidden layer feedback to input layer)
Hidden Layers	2
Input nodes	14
First hidden layer nodes	32
Second hidden layer nodes	32
Output nodes	2
Activation function	Logistic (f_1)
Scaling function	Logistic (f_1)
Weight updates	Momentum
Pattern Selection	Random
Initial Weights	0.2
Learning Rate	0.05
Momentum	0.3
Training Time	1,873,000 epochs
c)	
Type	Recurrent Net (Output layer feedback to input layer)
Hidden Layers	2
Input nodes	14
First hidden layer nodes	30
Second hidden layer nodes	30
Output nodes	2
Activation function	Logistic (f_1)
Scaling function	Logistic (f_1)
Weight updates	Momentum
Pattern Selection	Random
Initial Weights	0.3
Learning Rate	0.05
Momentum	0.4
Training Time	902,000 epochs

TABLE VII: Best nets. (a) Two hidden slabs with two activation functions. (b)Three hidden slabs with three activation functions. (c) Two hidden slabs with two activation functions and a jump connection.

a)	
Type	Two hidden layers
Hidden Layers	2
Input nodes	14
First hidden layer nodes	35
Second hidden layer nodes	35
Output nodes	2
Activation function (first hidden layer)	f_7
Activation function (second hidden layer)	f_8
Scaling function	Logistic
Weight updates	Momentum
Pattern Selection	Random
Initial Weights	0.5
Learning Rate	0.01
Momentum	0.1
Training Time	5,207,000 epochs
b)	
Type	Three hidden layers
Hidden Layers	3
Input nodes	14
First hidden layer nodes	55
Second hidden layer nodes	225
Third hidden layer nodes	55
Output nodes	2
Activation function (first hidden layer)	f_7
Activation function (second hidden layer)	f_1
Activation function (third hidden layer)	f_8
Scaling function	Logistic
Weight updates	Momentum
Pattern Selection	Random
Initial Weights	0.5
Learning Rate	0.01
Momentum	0.1
Training Time	5,993,000 epochs
c)	
Type	Two hidden layers with a jump connection
Hidden Layers	2
Input nodes	14
First hidden layer nodes	90
Second hidden layer nodes	90
Output nodes	2
Activation function (first hidden layer)	f_7
Activation function (second hidden layer)	f_8
Scaling function	Logistic
Weight updates	Momentum
Pattern Selection	Random
Initial Weights	0.5
Learning Rate	0.05

Momentum	0.1
Training Time	6,007,000 epochs

TABLE VIII: Details of best nets for the different neural network architectures. (a) GRNN. (b) GMDH.

a)	
Type	GRNN
Hidden Layers	1
Input nodes	14
Hidden layer nodes	440
Output nodes	2
Calibration	Iterative
Training Time	24,003,000 epochs
b)	
Type	GMDH
Hidden Layers	1
Input nodes	14
Output nodes	1
Selection Criteria	Regularity[22]
Training Time	35,325,000 epochs

TABLE IX: a) Results of applying the final model to 24 modified bisphenol-A polycarbonates. b) Results of applying the final model to 19 modified poly(2,6-dimethyl-1,4-phenylene oxide)

Monomer	T_α / T_γ	Dynamic Modulus (20^oC, dynes/cm^2)
a)		
Modification PC-1	3.13	5.67×10^9
Modification PC-2	2.70	5.22×10^9
Modification PC-3	2.74	5.38×10^9
Modification PC-4	3.37	6.39×10^9
Modification PC-5	2.58	5.06×10^9
b)		
Modification PPO-1	1.98	6.32×10^9
Modification PPO-2	2.29	6.59×10^9

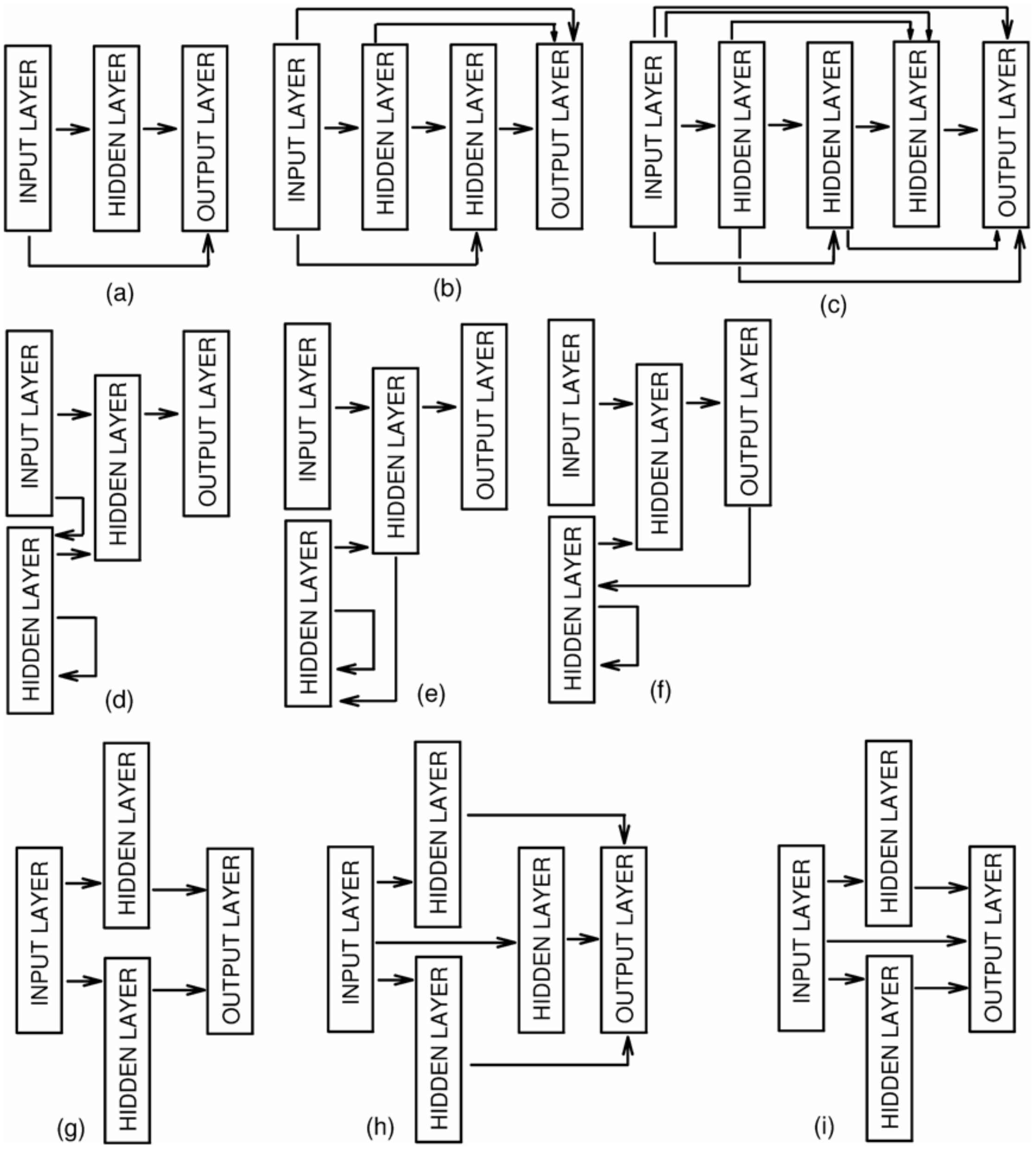

FIGURE 1: Neural Network architectures based on the standard backpropagation algorithm: (a) Three layers with a jump connection. (b) Four layers with jump connections. (c) Five layers with jump connections. (d) Recurrent Net with input layer dampened feedback. (e) Recurrent Net with hidden layer dampened feedback. (f) Recurrent Net with output layer dampened feedback. (g) Two hidden slabs with possibility of two activation functions. (h) Three hidden slabs with possibility of three activation functions. (i) Two hidden slabs with possibility of two activation functions and a jump connection.

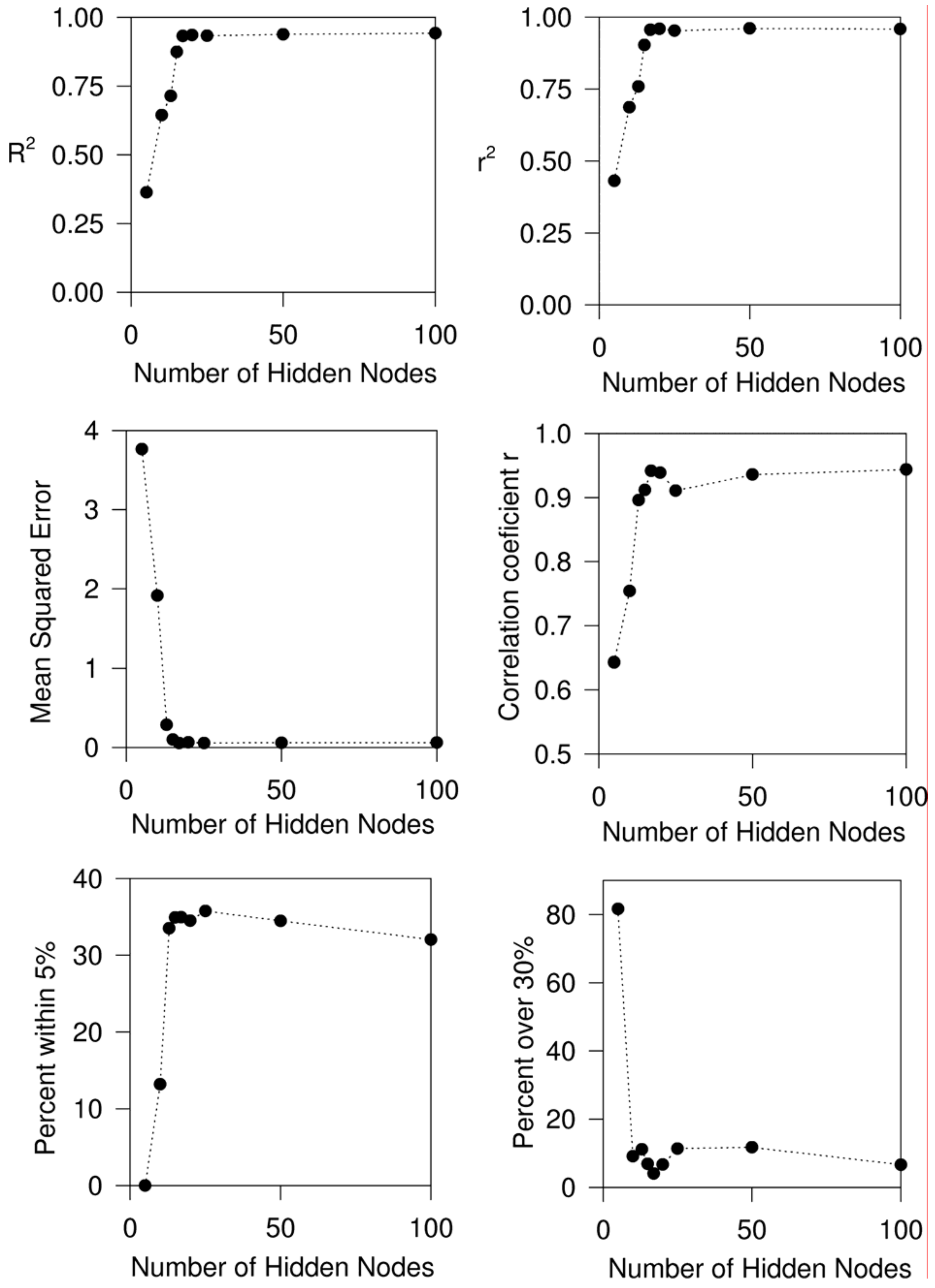

FIGURE 2: Error plots for variation of number of hidden nodes in a single hidden layer backpropagation neural network.

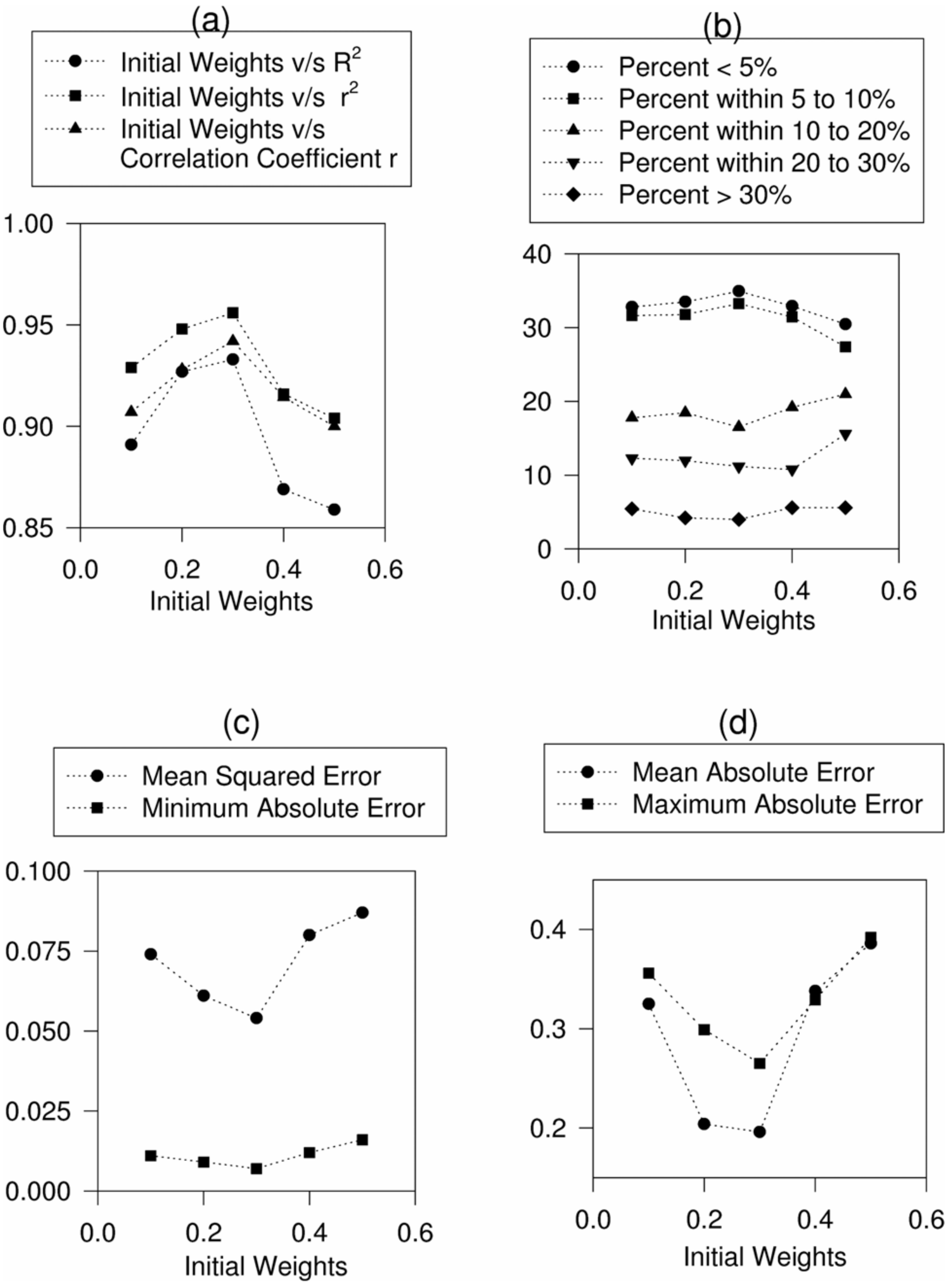

FIGURE 3: Effect of varying the initial weights.

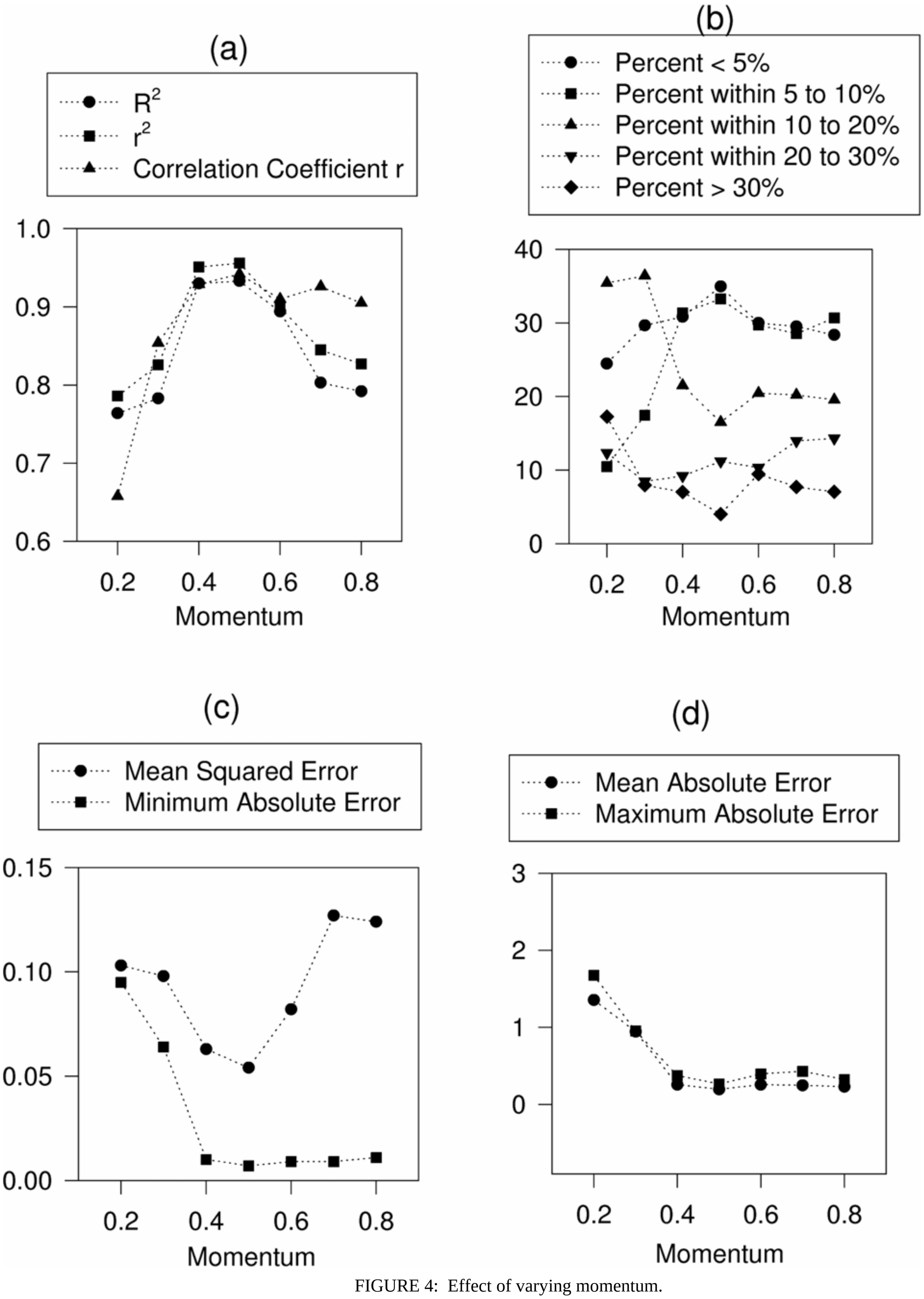

FIGURE 4: Effect of varying momentum.

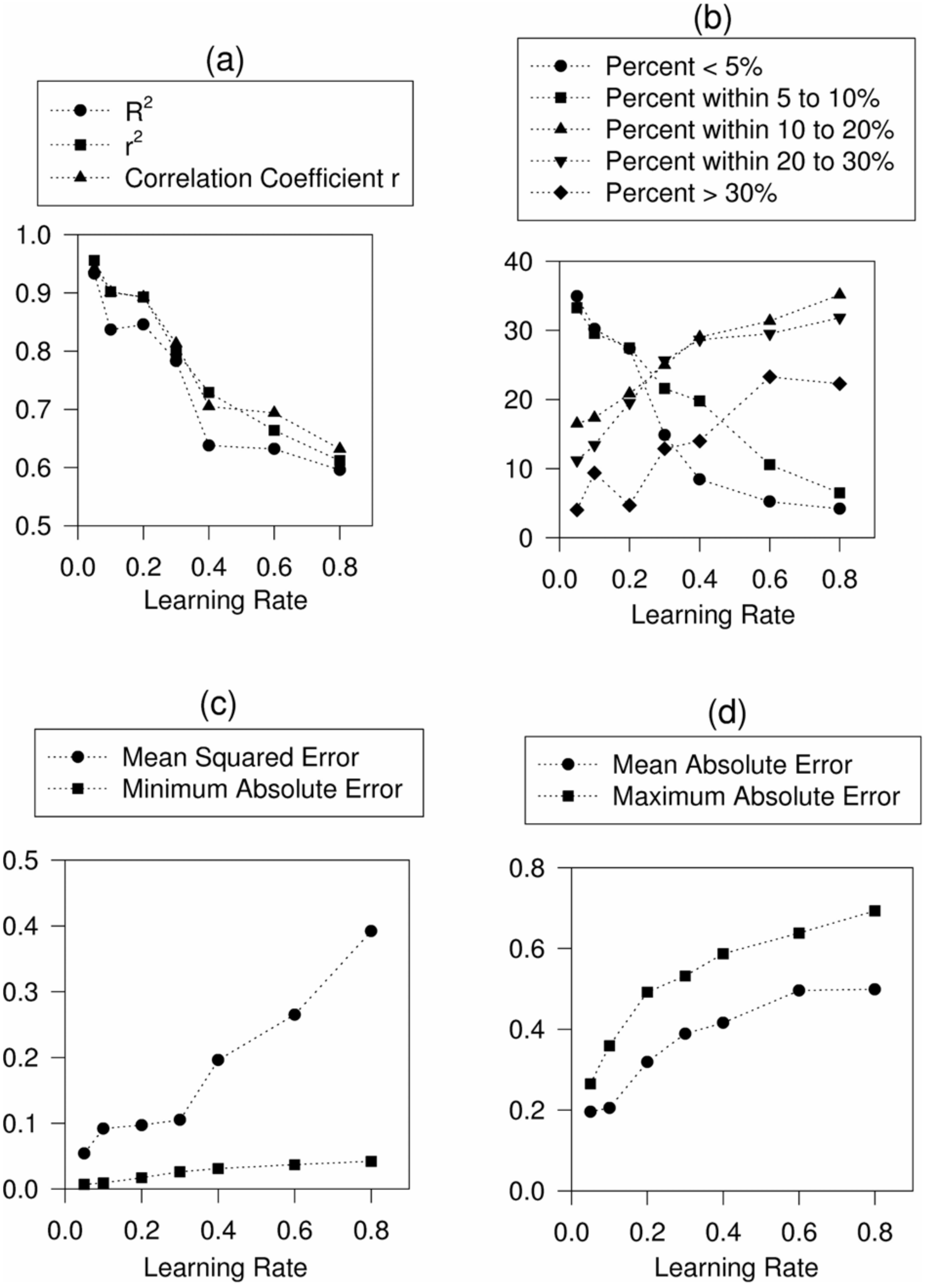

FIGURE 5: Effect of varying learning rate.

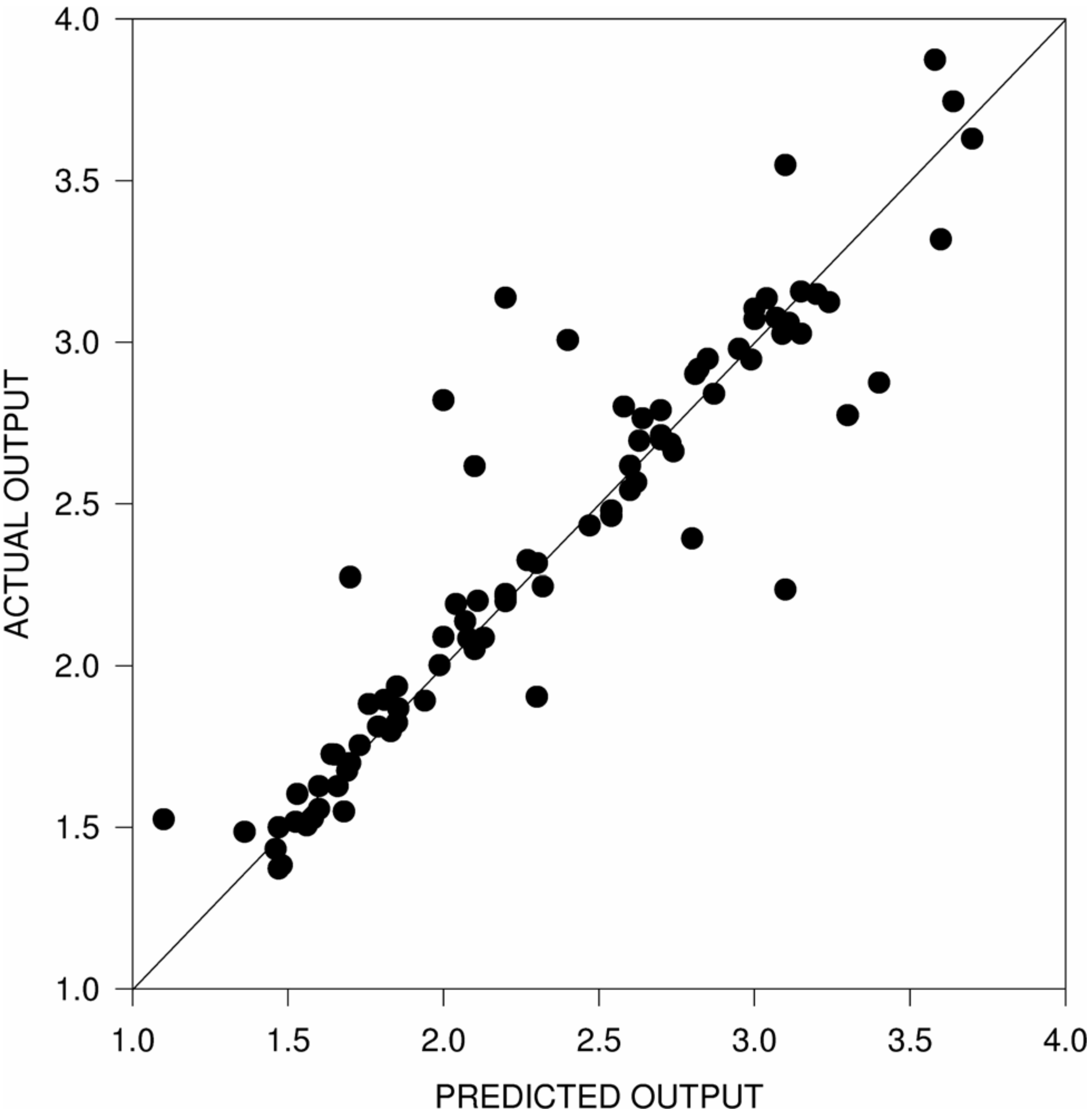

FIGURE 6: Actual versus predicted output when final model is applied to the 88 pattern testing set.

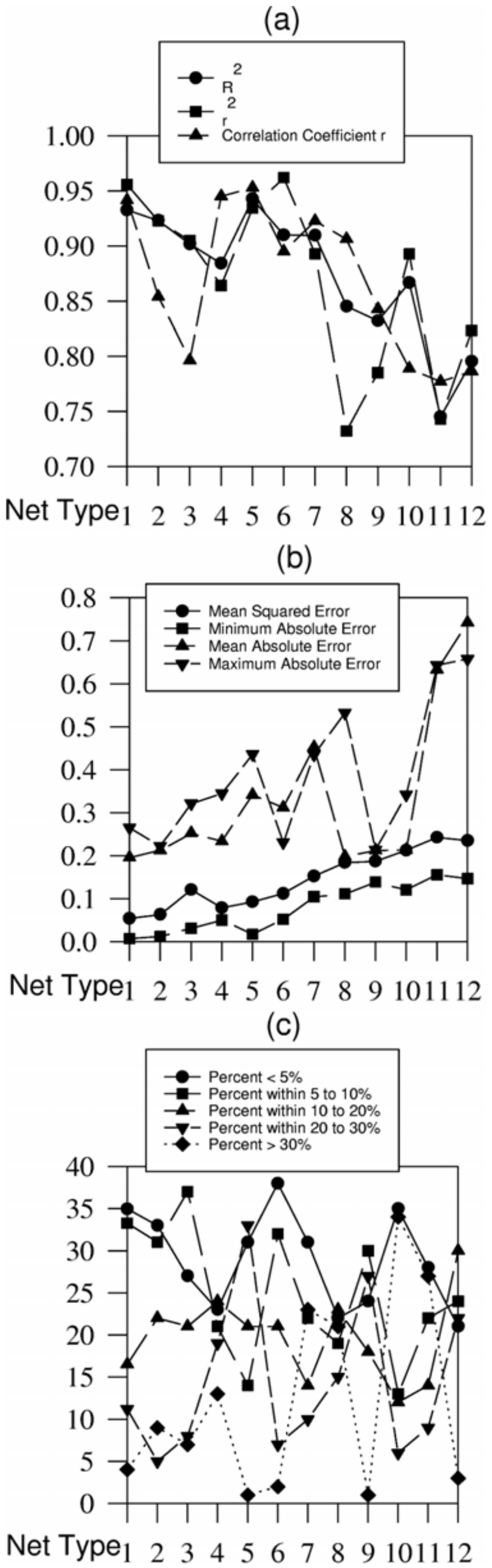

FIGURE 7: Comparisons of the different best networks. (Net Type: 1. Final Selected Model, 2. Three layers with a jump connection, 3. Four layers with jump connections, 4. Five layers with jump connections, 5. Recurrent Net with input layer dampened feedback, 6. Recurrent Net with hidden layer dampened feedback, 7. Recurrent Net with output layer dampened feedback, 8. Two hidden slabs with possibility of two activation functions, 9. Three hidden slabs with possibility of three activation functions, 10. Two hidden slabs with possibility of two activation functions and a jump connection, 11. GRNN, and 12. GMDH.)

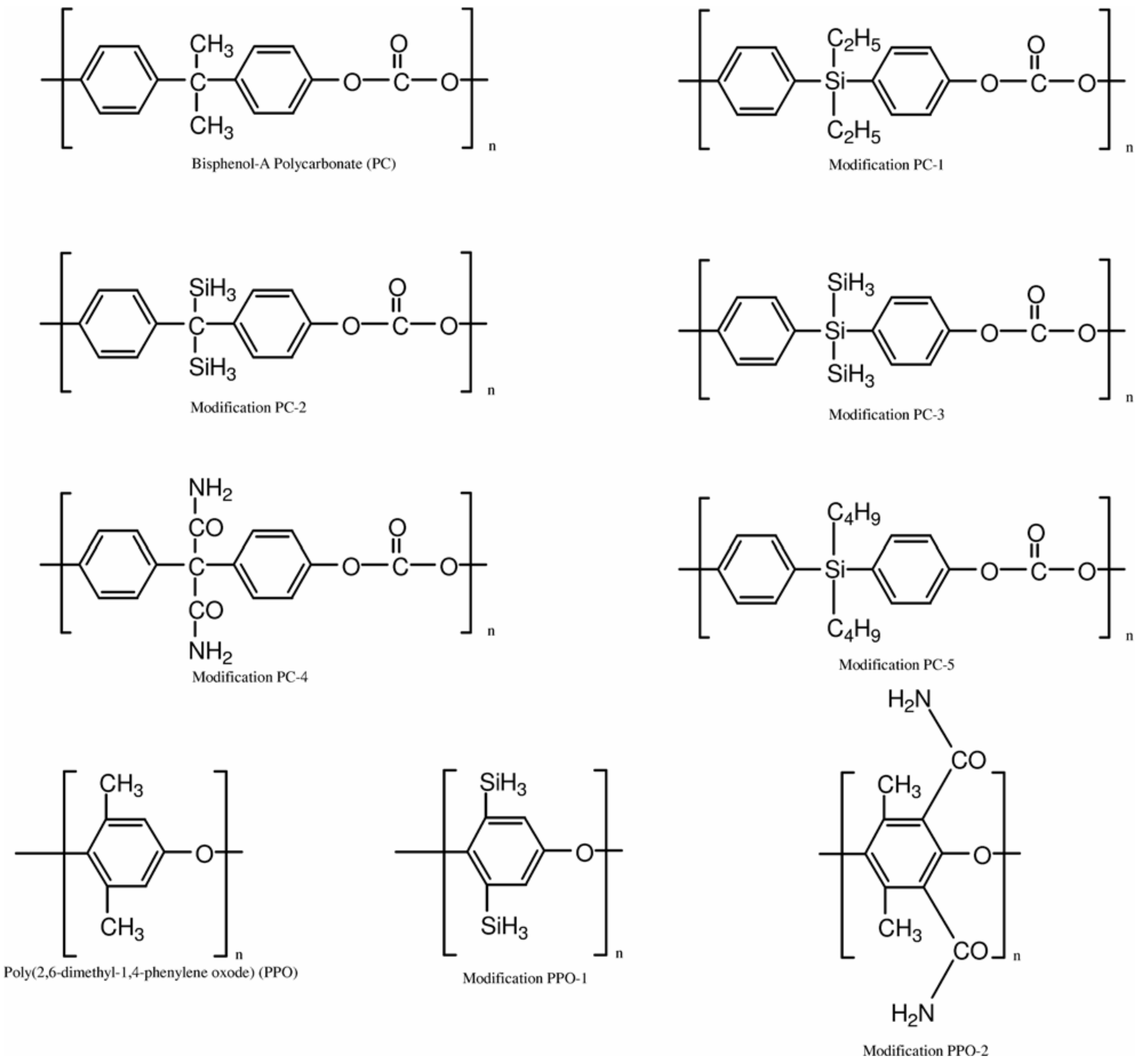

FIGURE 8: Structures of the mers of PC and PPO with their modifications predicted to have better mechanical properties.