J. Pharm. Pharmacol. 1996, 48: 581–591
Received July 21, 1995
Accepted November 13, 1995

Neural Network Computer Simulation of Medical Aerosols

C. J. RICHARDSON AND D. J. BARLOW

Department of Pharmacy, King's College London, London, UK

Abstract

Preliminary investigations have been conducted to assess the potential for using artificial neural networks to simulate aerosol behaviour, with a view to employing this type of methodology in the evaluation and design of pulmonary drug-delivery systems.

Details are presented of the general purpose software developed for these tasks; it implements a feed-forward back-propagation algorithm with weight decay and connection pruning, the user having complete run-time control of the network architecture and mode of training. A series of exploratory investigations is then reported in which different network structures and training strategies are assessed in terms of their ability to simulate known patterns of fluid flow in simple model systems. The first of these involves simulations of cellular automata-generated data for fluid flow through a partially obstructed two-dimensional pipe. The artificial neural networks are shown to be highly successful in simulating the behaviour of this simple linear system, but with important provisos relating to the information content of the training data and the criteria used to judge when the network is properly trained. A second set of investigations is then reported in which similar networks are used to simulate patterns of fluid flow through aerosol generation devices, using training data furnished through rigorous computational fluid dynamics modelling. These more complex three-dimensional systems are modelled with equal success.

It is concluded that carefully tailored, well trained networks could provide valuable tools not just for predicting but also for analysing the spatial dynamics of pharmaceutical aerosols.

Artificial neural networks (ANNs) are being used in many diverse fields in science and engineering to solve problems in novel ways (Zupan & Gasteiger 1991). Their use in the pharmaceutical sciences is a new but expanding field, with applications including quantitative structure–activity relationships (Aoyama et al 1990; Elrod et al 1990; Bodor et al 1991; Hussain & Johnson 1992; Hussain et al 1992b; Cherquaoi & Villemin 1994), molecular graphics (Gasteiger et al 1994), pharmacokinetics (Veng-Pedersen & Modi 1992; Hussain et al 1993) and product formulation (Hussain et al 1991, 1992a, c, 1994).

In our own research we have been interested to explore the use of ANNs in the simulation of aerosol behaviour, with the ultimate aim of employing this technique in the analysis and design of pulmonary drug delivery systems. Before embarking upon detailed investigation, however, we have elected first to determine the viability of such an approach using simple model systems. In the initial work detailed here, a series of ANNs is used to analyse and predict the behaviour of a simple (cellular automata-generated) linear model of aerosol flow through a partially obstructed two-dimensional pipe. We demonstrate that ANNs are highly successful in predicting the rules (in this case known) that govern aerosol behaviour in this system.

Further work is then reported in which we use the same type of ANN to predict the behaviour of single-phase computational fluid dynamics (CFD) models of three-dimensional aerosol generation devices. We show that the increase in complexity of the systems being modelled and the move from two to three dimensions does not reduce the ability of ANNs to predict flow dynamics.

We conclude that further application of neural networks to modelling aerosol flow (not only in aerosol devices but also in pulmonary airways) could be of value in assessing the effectiveness of pulmonary administration of drugs, and that such methods may prove superior to the existing models of these processes (Heyder & Rudolf 1984).

Correspondence: D. J. Barlow, Department of Pharmacy, King's College London, Manresa Road, London SW3 6LX, UK.

Materials and Methods

Feed-forward back-propagation ANNs

The networks used are two-layer perceptron feed-forward back-propagation ANNs (Rumelhart et al 1986), with the general structure shown in Fig. 1. In each ANN, the activation of input neurons is imposed by the input data set. The activation of neurons in the output layer is calculated by considering the weighted sum of the outputs of input neurons and the bias for that output neuron. With N input neurons of output x_i, the activation of the jth output neuron, a_j is calculated as:

$$a_j = w_{jo} + \sum_{i=1}^{i=N} w_{ji} x_i \qquad (1)$$

where w_{j0} is the bias for the jth output neuron and w_{ji} is the weight of the connection between the ith input neuron and the jth output neuron. The activation of a neuron is modified by some form of differentiable activation function to produce its output, so that $a'_j = f(a_j)$ (see below).

During training, modifications of the network weights and biases are made by back propagation of error. At iteration number n the new weights are calculated as:

$$w_{ji}^{n+1} = w_{ji}^{n} + \beta(\delta_i x_j) + \alpha(w_{ji}^{n} - w_{ji}^{n-1}) \qquad (2)$$

and the new biases as:

$$w_{jo}^{n+1} = w_{jo}^{n} + \beta(\delta_i) + \alpha(w_{jo}^{n} - w_{jo}^{n-1}) \qquad (3)$$

where α is a momentum factor, β is the learning rate and δ is the error signal. For the output layer:

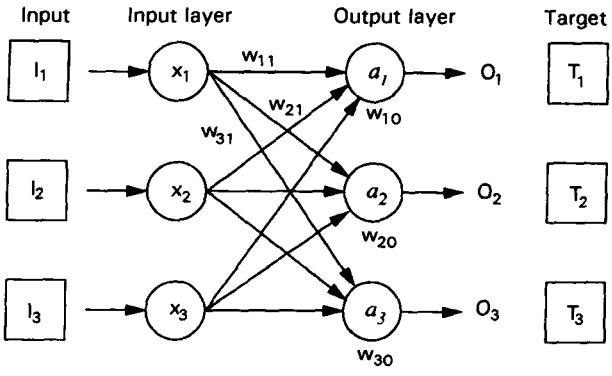

FIG. 1. Structure of a simple, two-layer perceptron.

$$\delta_i = (T_i - O_i)f'(O_i) \tag{4}$$

where T_i is the ith element of the target in the training set, O_i is the output of the ith output neuron and $f'(x)$ is the first derivative of the activation function.

Back-propagation of error is performed by batch training, i.e. the required changes to weights and biases are calculated for each input-target pair as they are encountered in the training set but the corrections to the network are applied only after all pairs have been tested. Training continues until the number of iterations exceeds a set maximum, or the mean square error reaches a set minimum. The mean square error, ε, is calculated as:

$$\epsilon = \frac{\sum_{i=1}^{i=N}(T_i - O_i)^2}{N} \tag{5}$$

In the more developed ANNs reported here (see below), the network connectivity is dynamically updated using the techniques of weight decay and connection pruning. It has been shown that this adaptive regularization can improve the performance of a feed-forward back-propagation network (Le Cun et al 1990).

Weight decay introduces a decay term, γ, into the weight update function (eqn 2) which tends to prevent the connection weights from deviating markedly from zero:

$$w_{ji}^{n+1} = w_{ji}^{n}\beta + (\delta_i x_j) + \alpha(w_{ji}^{n} - w_{ji}^{n-1}) - \gamma w_{ji}^{n} \tag{6}$$

Weight decay tends to reduce preferentially the strength of redundant connections, and these are removed by connection pruning — with all connections with weights below a specified threshold (ξ) being deleted. The network pruning is performed at regular (user-defined) intervals, and after each pruning epoch the weight decay term is reduced in magnitude. This has the effect that initial pruning events determine a rough connection pattern, which is then fine-tuned during the final stages of a training session.

Network structure and function

The nature of the task given to a neural network determines its structure. In this work the aim was to produce networks which could predict the dynamic behaviour of fluid flow systems. The task for a network, when given the state of the fluid to be modelled at time t, was to predict the resulting state at time $t+1$. A two-layer perceptron was the simplest network structure suited to this task, the data for time t being presented to the input layer of the network and the prediction for time $t+1$ being produced as the activation of output layer neurons. A correctly trained network, therefore, could when given a snapshot of the system, predict the subsequent state of the system, and so if repeatedly fed with its own output could give a simulation of the system's evolution through time, which could then be compared with the experimental data. Although this introduces recurrent links into the network, the structures used here cannot be considered as recurrent networks as such, as these links are not present during the training process. Rather they enable a previously trained network to predict the complete trajectory of a variable through time (Bulsari & Palosaari 1993).

In all the examples presented below, the data were obtained by dividing the flow system to be modelled into a series of (two- or three-dimensional) cells. This controlled the size of the networks used. For each data cell in the experimental model there was a corresponding input-layer and output-layer neuron in the network. Thus, for the initial two-dimensional models with a 12×20 grid of data cells, the corresponding network had input and output layers each containing 240 neurons. The initial CFD model produced data in a 16×16 plane through the experimental volume, and the corresponding network contained two layers each with 256 cells. Two 16×16 data planes were extracted from the final CFD model, giving a network with two layers of 512 cells.

Three network connection paradigms are presented here. The canonical connection scheme connects every neuron in the input layer to every neuron in the output layer, producing a fully connected network (Fig. 2). In an effort to reduce network complexity, locally connected networks were introduced in which any given neuron in the bulk of the input layer is only connected to the nine "local" neurons on the output layer — the corresponding neuron on the output layer and the eight neurons surrounding it (Fig. 2). To eliminate edge effects, the connectivity of neurons at the perimeter of the input layer is reduced, with edge and corner neurons being connected to six and four output neurons, respectively.

Finally, to eliminate the need for an arbitrary decision as to the definition of "local" in this context, the devices of weight decay and connection pruning were introduced. These enabled the network to alter the number of its internal connections dynamically during the training process.

For all networks detailed here, a variety of parameters needs to be decided upon before each training run. Whereas some affect the quality of learning, most influence only the degree

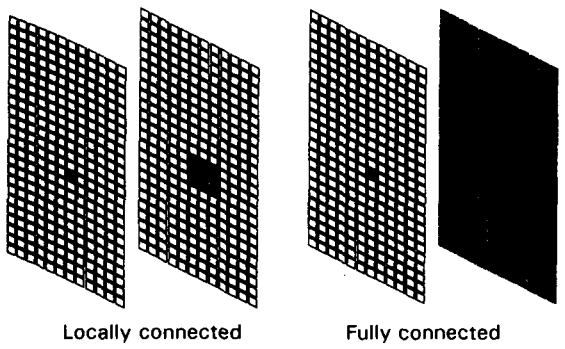

FIG. 2. Two different network connection schemes. In each case the one highlighted input layer neuron forms connections with the set of highlighted output layer neurons.

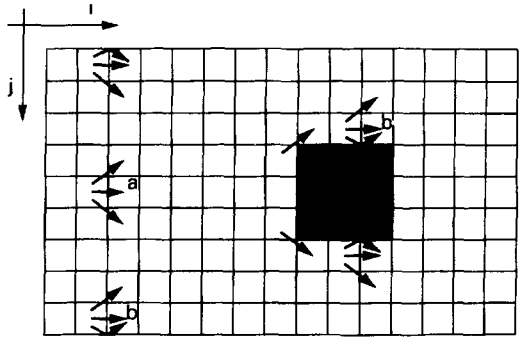

FIG. 3. Rules governing cellular automata behaviour. The 2-D pipe is partitioned into a rectangular grid of cells, with a centrally positioned 3 × 3 cell obstruction. The aerosol from selected grid cells flows in the directions indicated by the arrows.

and speed of convergence. In all runs, the momenta and gains (α and β), the activation function (f(x), which can be tanh, sigmoid or linear), the maximum number of iterations, the weight decay term (γ), and the interval between pruning events and the pruning threshold (ξ) were adjusted by hand to give the best performance.

Implementation

All calculations were performed on a Silicon Graphics Crimson UNIX work-station, with a MIPS R4000 CPU, R4010 FPU and 32 Mbytes of main memory. Although the studies could have been performed using one of the many neural network packages available in the public domain, the training and analyses were actually performed with a suite of custom-written programs, PUDDLE (PUlmonary Drug Delivery Learning Engine), because this facilitated manipulations of the network connectivity, and enabled the use of machine-specific graphics libraries. With PUDDLE, the user has complete run-time control over all aspects of network size and structure and all parameters controlling the learning process. The output from simulations of system behaviour can be displayed graphically, either as static colour/grey scale and contour images for the production of hard copies, or as animated images in the same format. All graphical output exploits the IRIX GL graphics library's in-built facilities for fast polygon handling and double-buffering.

Cellular automata model of 2-dimensional aerosol flow

The data used to train networks in the initial studies presented here used a cellular automata model of aerosol fluid flow in a two-dimensional pipe. Each cell of the automaton corresponds to an element of the pipe which is partitioned into a 12 × 20 rectangular grid, and the entry for each cell corresponds to the concentration of aerosol in that element of the pipe. Aerosol particles are redistributed according to the rules detailed below and shown in Fig. 3.

For a cell (x,y), given the state of the network at time t, the contents of that cell, $c^{t+1}{}_{x,y}$, at time t + 1 can be calculated as follows.

For the cells of type (a):

$$c_{i,j}^{t+1} = (1-\phi)c_{i,j}^{t} + \phi(1/(1+2\cos\pi/4))c_{i-1,j}^{t} + \phi((\cos\pi/4)/(1+2\cos\pi/4))(c_{i-1,j-1}^{t} + c_{i-1,j+1}^{t}) \quad (7)$$

For the cells of type (b) and also those adjacent to the top wall of the pipe and the obstacle:

$$c_{i,j}^{t+1} = (1-\phi)c_{i,j}^{t} + \phi(1/(1+2\cos\pi/4))c_{i-1,j}^{t} + \phi((1+\cos\pi/4)/(1+2\cos\pi/4))c_{i-1,j-1}^{t} \quad (8)$$

In summary, therefore, for a given element of the pipe, a proportion of the particles remains therein until the next time step, and those flowing out of the element do so into the three elements directly down-flow. The proportion leaving an element is governed by the variable ϕ, which in a real system might correspond to flow rate, gas viscosity, or perhaps particle density. In the model reported here, $\phi = 0{\cdot}75$ and there is a 3 × 4 cell obstruction positioned centrally, half-way down the pipe.

The experimental data are generated as sets of snapshots of the aerosol flow through the pipe through time. A bolus of material (of random density) is introduced at the top of the pipe and the flow of the aerosol cloud is calculated iteratively over a series of time steps from t = 0 to t = 30 using the rules contained in the automata.

Computational fluid dynamics model of 3-dimensional fluid flow

In order to generate more complex and realistic data for use in ANN training, two different three-dimensional fluid flow systems were simulated by means of computational fluid dynamics (CFD) techniques using the commercially available CFD software Phoenics. Phoenics uses finite-volume techniques to transform the differential equations governing heat, mass and momentum flow into a set of algebraic equations by integration over each cell in the computational domain. These algebraic equations are then solved for each cell using iterative approximation methods. In both the models used here, the simulations involve transient flow conditions, and are solved using Phoenics' non-parabolic mode with time steps 10 ms apart. Heat transfer in Phoenics is controlled by setting the laminar Prandtl number for temperature; in both CFD models it has a value of 0·71. Obstructions such as the baffle and jet walls were modelled by setting their block porosity to zero. Boundaries which allow free flow of fluid were created as constant-pressure patches, with diffusive movement of enthalpy through the patch set to zero; solid walls disallow both mass and enthalpy transport.

In both the CFD models the flow is simulated in a single-phase system. As a means of modelling the mass-transport and

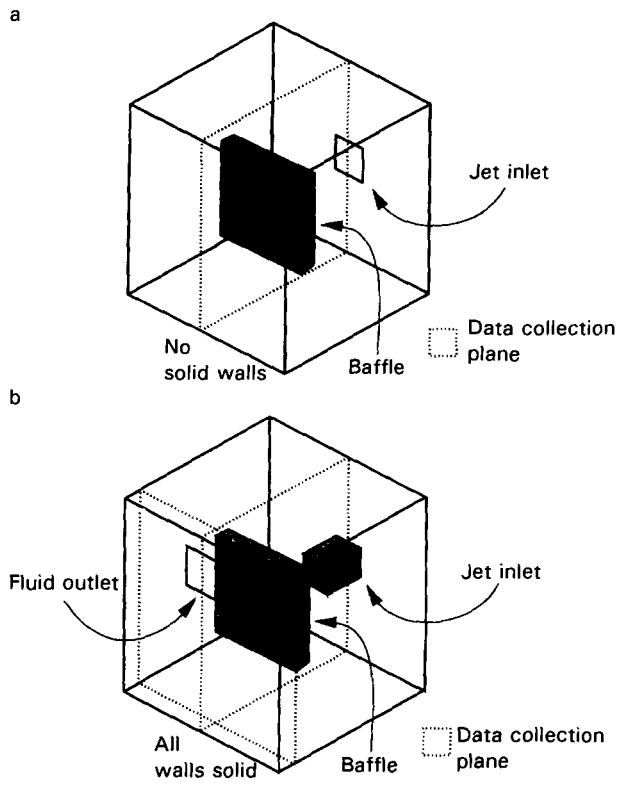

FIG. 4. First model (a) and second model (b) for CFD calculations.

diffusion properties of aerosol clouds, temperature is used as a marker for aerosol concentration. By using this technique we greatly reduce the computational tasks involved, while retaining the important features of realistic fluid flow. No buoyancy terms were employed, to eliminate the effects of convection.

The first CFD model simulated comprised a 5 cm cube filled with cold fluid, with a 5 m s^{-1} jet of hot fluid emerging through a nozzle and impinging upon a centrally placed 8 mm × 8 mm baffle (Fig. 4a). The boundaries of this space allow free flow of fluid. The calculations were performed in a non-linear 16 × 16 × 16 grid, with the smaller unit cell dimensions at the centre of the tank affording greater accuracy in the calculations of the complex flow around the baffle. The data derived from this model are point temperature values on a plane passing through the centre of the tank, bisecting the jet vertically. Three simulations of jet flow over 180 ms were performed, with jet temperatures 5°, 10° and 20° above the bulk fluid temperature.

The second CFD model simulated involved a solid-walled cuboid device (of the same dimensions as the first) providing a crude approximation to a jet nebulizer device. An inlet tube 1 cm long with a square 4 mm × 4 mm cross-section emerges centrally from one wall, with a fluid jet emerging from this tube (after being heated by its walls) and impinging upon a centrally positioned baffle (Fig. 4b). Fluid leaves the device through a 1 cm × 4 mm outlet positioned near the top of the wall opposite the inlet tube. The calculations were performed on a nonlinear 16 × 16 × 16 grid of cells for three runs of 180 ms each, with jet temperatures 5°, 10° and 20° above the bulk fluid temperature. The data derived from this model are point temperature values on two planes passing through the device. The first plane, as in the previous model, passes through the centre of the device bisecting the jet vertically. The second plane (perpendicular to the first) passes vertically through the rear of the device, behind and parallel to the baffle.

Results

The networks used here to model fluid flow became more complex and refined as our understanding of the modelling processes involved in training two-layer perceptrons to predict flow increased. The networks used are unusual in that they possess a large number of both input and output neurons, and so have a very large number of internal connections, which increases the number of variables to be refined during the training process.

For all networks the training was continued until the mean square error fell below a user-specified value, or until a maximum number of iterations was exceeded. In all the examples presented, the limit for mean square error was set at 10^{-7}. Rather than relying upon the mean square error to assess the performance of a trained network, however, other, more stringent, methods were also used. Given that a correctly functioning network can predict the aerosol distribution at time t + 1 given the distribution at time t, feeding the output of such a network to its input layer should generate a series of predictions of aerosol distribution. Such a simulation of the system's evolution through time can be compared with the behaviour of the system itself. The starting state for such a simulation can either be one which the trained network has encountered in its training set (testing the network's ability to memorize how that particular aerosol cloud flows) or one which has not been presented to the network (testing its ability to generalize about flow). Unless mentioned below, all illustrations in this paper are of simulations obtained using previously unseen examples of flow.

With the cellular automata model, we have used a further method to assess the quality of the training process. Knowing the rules embedded in the automaton (and having a direct mapping between the experimental system and the network structure) enables calculation of the correct values for neuron connection weights. During training, a successful network should "discover" these values. By examining the connection weights and biases, the success of training can be determined.

The first network (network I) employed to model aerosol flow through the partially obstructed two-dimensional pipe was the simplest. This network was fully connected, used the sigmoid activation function, $f(x) = 1/(1 + e^{-x})$, and scaled both the input and output data into the most linear portion of the activation function, between 0·1 and 0·8. The training data set was derived from a single example of flow through the pipe, with 10 time steps used from a run of 30 steps. This was the only network reported here to achieve the limit of mean square error, reaching 10^{-7} after 20 400 iterations. Table 1 shows the correct and achieved local connection weights in network I. Despite the low mean square error achieved, the network has not discovered the rules governing aerosol flow through the pipe. This is demonstrated by Fig. 5, which shows time steps 1–9 and 21–29 in a network simulation of flow, above the corresponding time steps from the cellular automata calculations of flow. The starting state for this simulation was one

Table 1. Local connection weights: correct values and values extracted from trained networks I to V.

0·0000	0·0000	0·0000	0·0159	0·0170	0·0000	− 0·0364	0·0731	− 0·0715
0·0000	0·5000	0·0000	− 0·0023	− 0·0015	0·0000	0·08420-·1228	0·0929	
0·1470	0·2070	0·1470	0·0384	0·0000	0·0000	0·2294	0·1235	0·1427
a Correct values			b Network I			c Network II		
0·0133	0·0038	− 0·0037	− 0·0240	0·0607	0·0024	0·0000	0·0572	0·0147
− 0·0016	0·4968	0·0046	0·0521	0·4242	0·0380	0·1278	0·2012	0·1068
0·1563	0·2053	0·1539	0·1059	0·2541	0·1466	0·1469	0·1940	0·1344
d Network III			e Network IV			f Network V		

which the network had encountered in its training set. Aerosol concentration is shown by a grey-scale and isoconcentration contours. The simulation is poor in many respects. There is erroneous appearance of aerosol particles well ahead of the main cloud front, and at the end of the simulation, where only particles deposited above the obstruction should remain, there is aerosol distributed throughout the pipe.

One possible reason for poor performance of a fully connected network was that there were too many variables in the network when compared with the amount of data in the training set. The next network employed (network II) was locally connected, used the tanh activation function, $f(x) = \tanh(x)$, and used data scaled into the mostly linear portion of this activation function between 0·0 and 0·9. The training data set was again derived from a single example of flow through the pipe, with 10 time-steps used from a run of 30 steps. This network failed to reach the mean square error threshold, attaining a mean square error of $2{\cdot}78 \times 10^{-4}$ after 2×10^6 iterations (taking approx. 6 h in real time). At this stage, the network had effectively stopped learning, with very little change being recorded in the mean square error. The local connection weights were now closer to the desired values (Table 1), and the performance of the network in simulations was much improved (Fig. 6). The network still, however, incorrectly generated aerosol particles down the left-hand side of the pipe, well ahead of the advancing aerosol cloud. The flow on the right-hand side was, on the other hand, simulated correctly, as was the build up of aerosol particles above the obstruction, the flow around one side of the obstruction, and the development of clear air behind the main aerosol cloud.

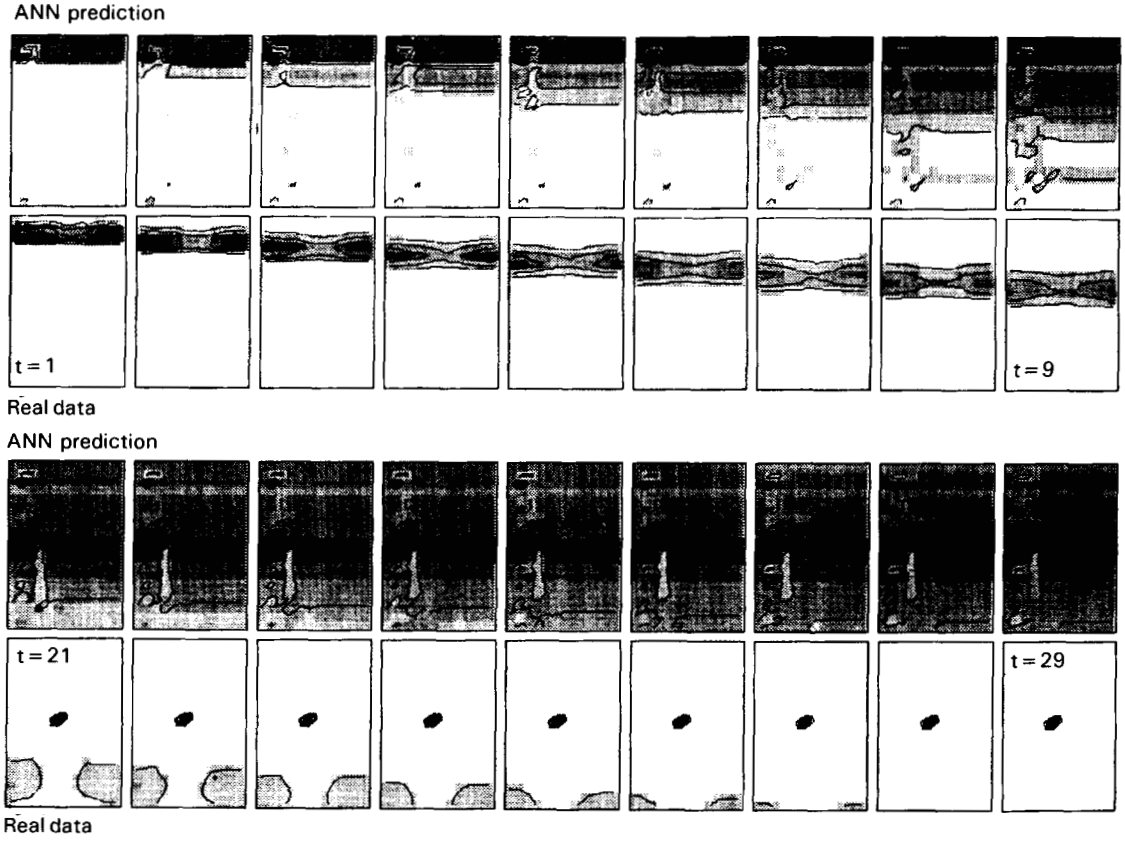

FIG. 5. A comparison of the real and (ANN) network I-simulated aerosol flow in the 2-D pipe. The pattern of aerosol density is shown by means of a grey scale with superimposed isodensity contours. The two rows of panels presented on top are for time steps 1–9, and those below are for time steps 21–29.

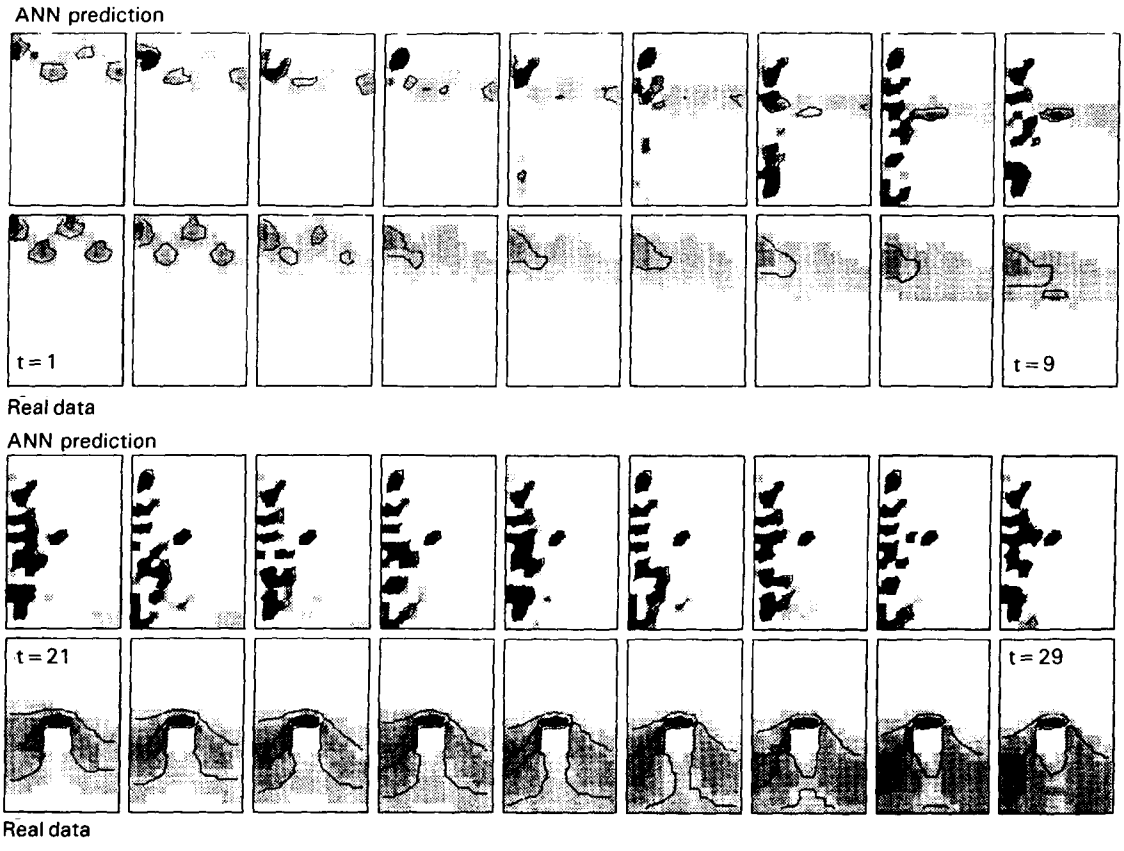

FIG. 6. A comparison of the real and (ANN) network II-simulated aerosol flow in the 2-D pipe. Details are as described for Fig. 5.

At this stage, it was assumed that the training sets still contained too little information for learning to proceed correctly. Training sets with a higher information content were artificially generated by presenting the cellular automata with a series of randomly generated concentrated, inhomogeneous aerosol clouds and combining 15 such examples of flow in one training set. This approach could not have been attempted with a real, physical system. The network with this data (network III) was identical to network II, with the training set being scaled in the range 0-0·025. A mean square error of $5{\cdot}58 \times 10^{-6}$ was attained after 1 000 005 iterations (taking approx. 2 h in real time), when the decrease in mean square error had effectively stopped. Performance was very good, with the network correctly discovering the rules governing flow (Table 1). In consequence there was very little difference seen between the cellular automata data and the network prediction, and the flow through the pipe and around the obstruction was found to be modelled very well (Fig. 7).

To generate data with a high information content in a less artificial way, the training set for the next network (network IV) was produced from six distinct examples of flow through the pipe (29 input-target pairs from each to give a total of 174). Network IV employed a linear activation function, with no scaling of the training set. A mean square error of 0·11 was attained after 1 000 152 iterations, when the decrease in mean square error had effectively stopped. Training was not as successful as with network III (Table 1). Fig. 8 shows that although the general characteristics of flow were successfully modelled the fine internal structure of the aerosol cloud and the rate at which the aerosol moved down the pipe were no predicted exactly.

The final network to be trained with cellular automata data incorporated weight decay and connection pruning. The initial decay constant (ξ) was 0·0001, with all weights below 0·01 being pruned every 300 000 iterations. This network (network V) was also trained with high information content data, and performed almost as well as those with imposed local connectivity. A final mean square error of 0·32 was attained after 1 000 152 iterations, when learning had effectively finished. By the end of the last pruning epoch, connectivity had been reduced from 100% to 6%. Fig. 9 shows the pattern of connections from cell [6,6] on the input layer, where a local connection scheme has been discovered during the training process. The local connection weights achieved after training were, however, not as close to the desired values as those in network IV (Table 1) and the pruning process did not completely remove all unnecessary connections. As a result, the ANN simulations showed an incorrect lateral movement of aerosol within the pipe, and an erroneous generation of aerosol as flow proceeds down the pipe. This is demonstrated in Fig. 10, which shows the inability of the network to model the fine structure of the aerosol cloud, an excessive aerosol concentration towards the end of the simulation, and the rear of the aerosol cloud advancing too slowly.

Learning with CFD models

As the flows involved in CFD simulations are more complex than those of the cellular automata, the desired values for con-

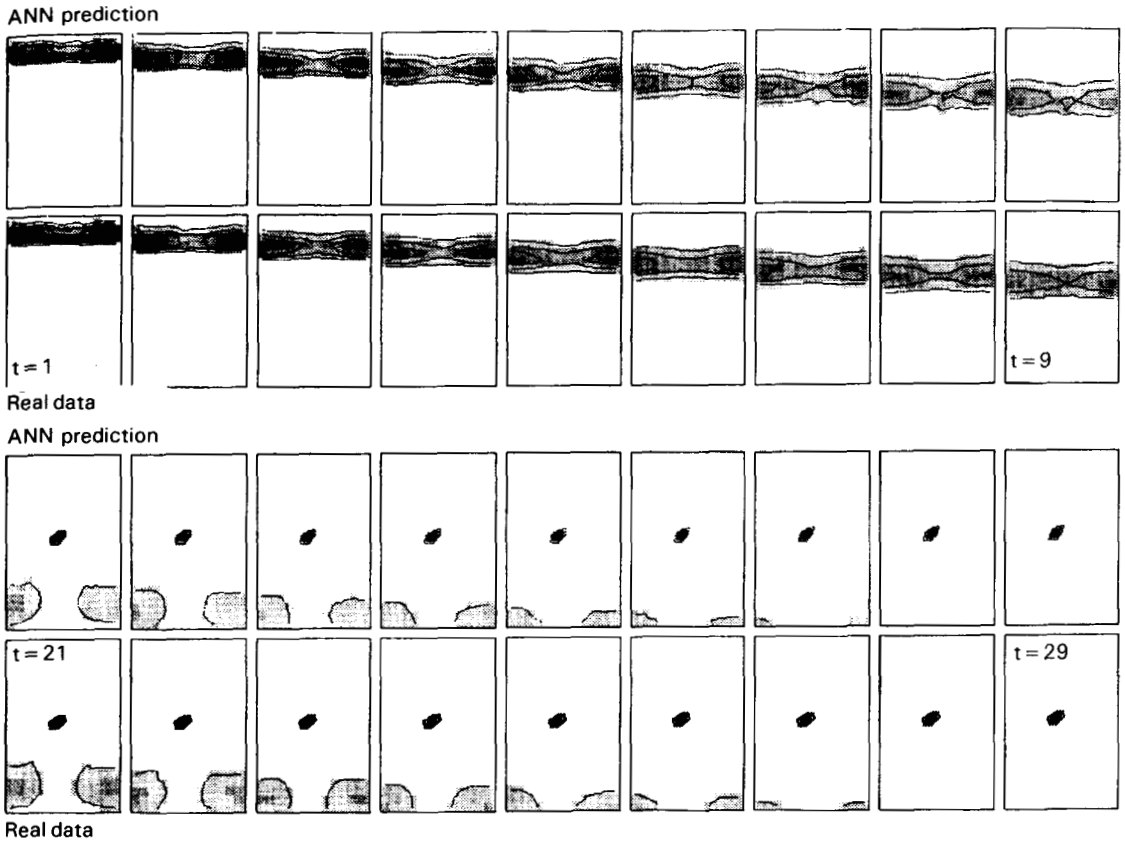

FIG. 7. A comparison of the real and (ANN) network III-simulated aerosol flow in the 2-D pipe. Details are as described for Fig. 5.

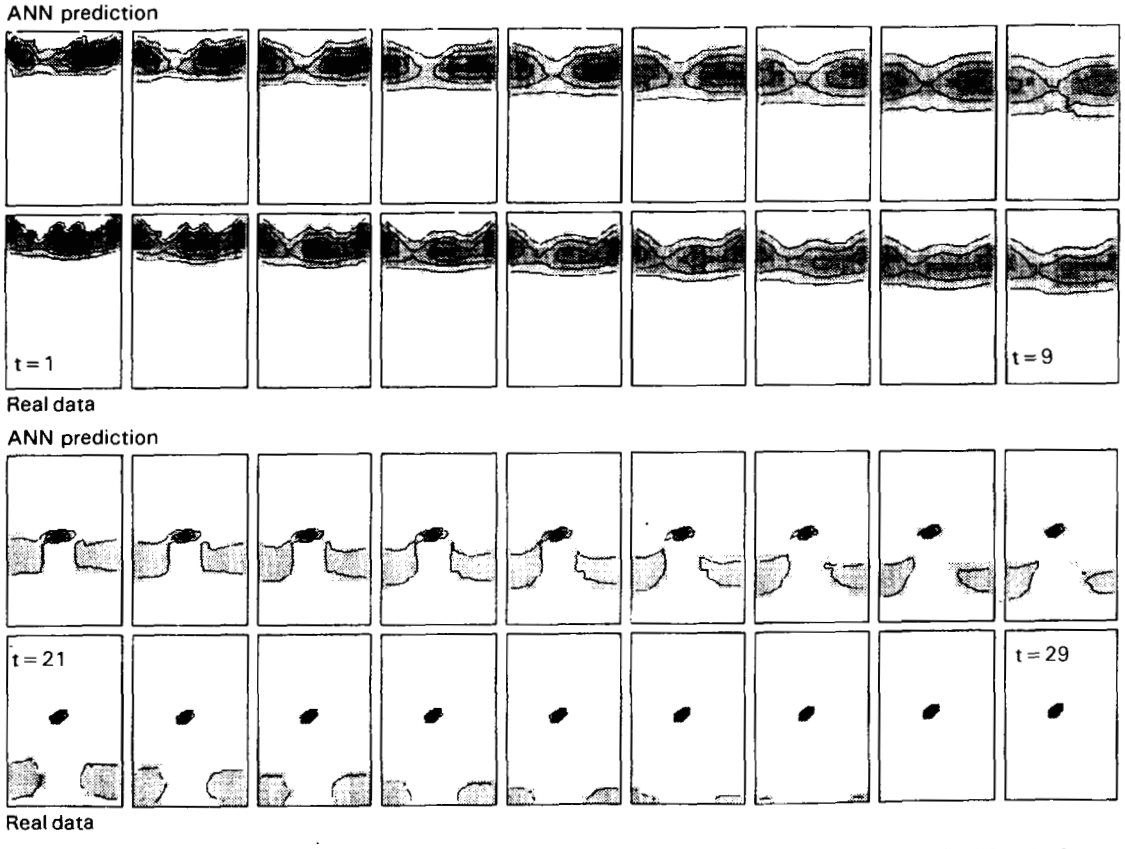

FIG. 8. A comparison of the real and (ANN) network IV-simulated aerosol flow in the 2-D pipe. Details are as described for Fig. 5.

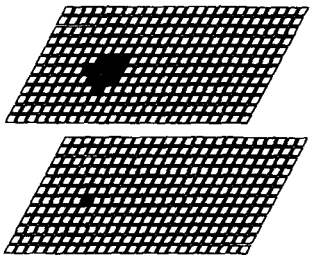

FIG. 9. An example of connectivity obtained using weight decay and connection pruning in the training of network V. The highlighted input layer neuron [6,6] has connections to the 11 highlighted output layer neurons.

nection weights cannot be predicted. Assessment of the quality of training can only be performed by examining the simulation performed using trained networks. The results obtained from training a network to predict the behaviour of flow in the first of the two CFD simulations were very promising. The network (network VI) used a linear activation function, $f(x) = x$, with unscaled training set data, obtained from all three CFD runs. Both weight decay and connection pruning were used, with the initial decay constant (ξ) being 0·0001 and all connection weights below a threshold of 0·0 being pruned every 200 050 iterations. When learning had effectively finished, the final mean square error was 0·029 after 1 000 008 trials (taking two days in real time). After the final pruning epoch, the degree of connectivity had been reduced from 100% to 58%. An example of the level of connectivity remaining after training is given in Fig. 11, which shows the connections to the output layer. Connectivity has been made local, but with a larger extent than seen in the cellular automata model. Fig. 12 shows time steps 1–5 and 11–15 from a simulation with network VI (contours shown are isotherms for CFD data), and there is very little difference between the network prediction and the CFD data.

The move to the second CFD system produced longer training times (four days in real time) and less accurate results. The trained networks could, nonetheless, still predict the patterns of aerosol flow. The network used (network VII) was similar to network VI, and was also trained with data from three CFD runs. Weight decay and pruning were used, with an initial decay constant of 0·0001 and pruning every 200 050 iterations. The final mean square error of 5·62 was obtained after 2 000 016 trials, at which point the mean square error was decreasing only very slowly. After the final pruning epoch, the degree of connectivity had been reduced from 100% to 24%. Fig. 13 shows a simulation with network VII, and is presented in a manner slightly different from previous simulations. On the left of each frame is the first data plane, on the right the second plane. Although the general pattern of flow is predicted well, there is one area at the top of the jet in the first plane where all

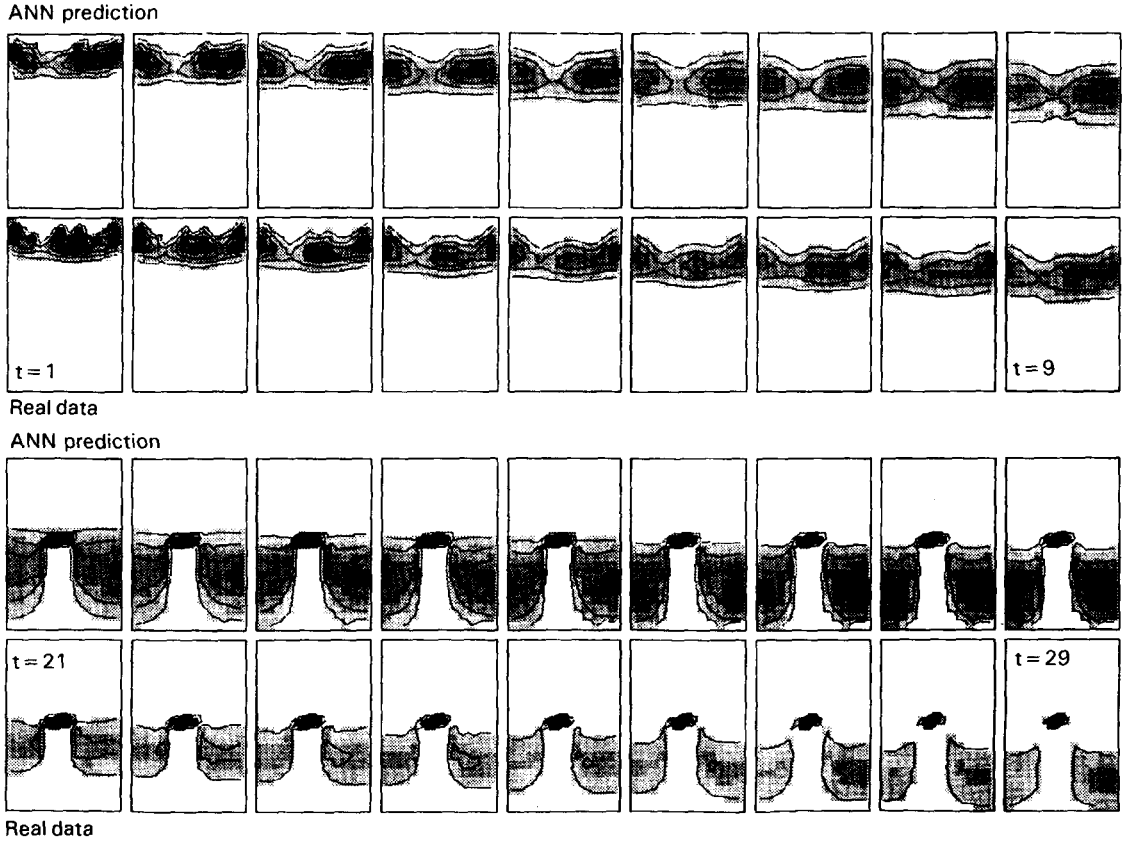

FIG. 10. A comparison of the real and (ANN) network V-simulated aerosol flow in the 2-D pipe. Details are as described for Fig. 5.

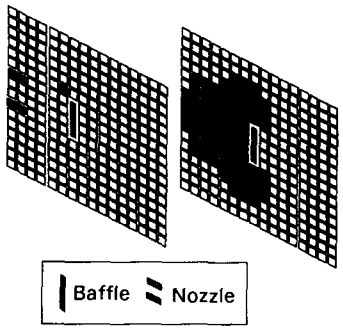

FIG. 11. An example of connectivity obtained using weight decay and connection pruning in the training of network VII. The highlighted input layer neuron [8,6] has connections to the 72 highlighted output layer neurons.

aerosol is missing; this probably occurred because valid connections were incorrectly pruned. In addition, the network predicts an aerosol cloud less dense than that present in the CFD data.

Discussion

The initial work presented here has demonstrated that ANNs can be successfully employed to model and predict aerosol flow in simple experimental systems. We have, moreover, developed techniques for successfully training our two-layer perceptrons and assessing their performance once trained.

In the presentation of results, importance has been given both to the move to local connectivity and to the exploration of the effect of the information content of the training set. These changes were driven by the need to increase the ratio of training data to variables within the network. In generating the results presented here we have seen not only that the user's choice of the many network parameters affects network performance but also that the quality and amount of information contained within the training set data is very important. Whereas fine tuning of such parameters as the momentum factor, α, and the learning rate, β, in most cases the speed of learning, the information content of the training set or the level of network connectivity, critically affect the success of learning. Initial experiments with low information content training sets did not produce networks that could successfully predict patterns of aerosol flow. Increasing the amount of information given to these networks enabled them not only to memorize the behaviour of aerosol clouds which they had encountered but also to generalize, predicting the behaviour of previously unseen clouds. This has relevance to the experimenter applying this approach to other experimental systems; care must be taken to obtain data covering as many different states of the system as possible. In order to obtain ANNs correctly trained to model aerosol flow in either generation devices or pulmonary airways, the experimenter must ensure that the data describing aerosol flow has a sufficiently high information content, a task which may well be difficult to achieve given current methods of investigating aerosol flow.

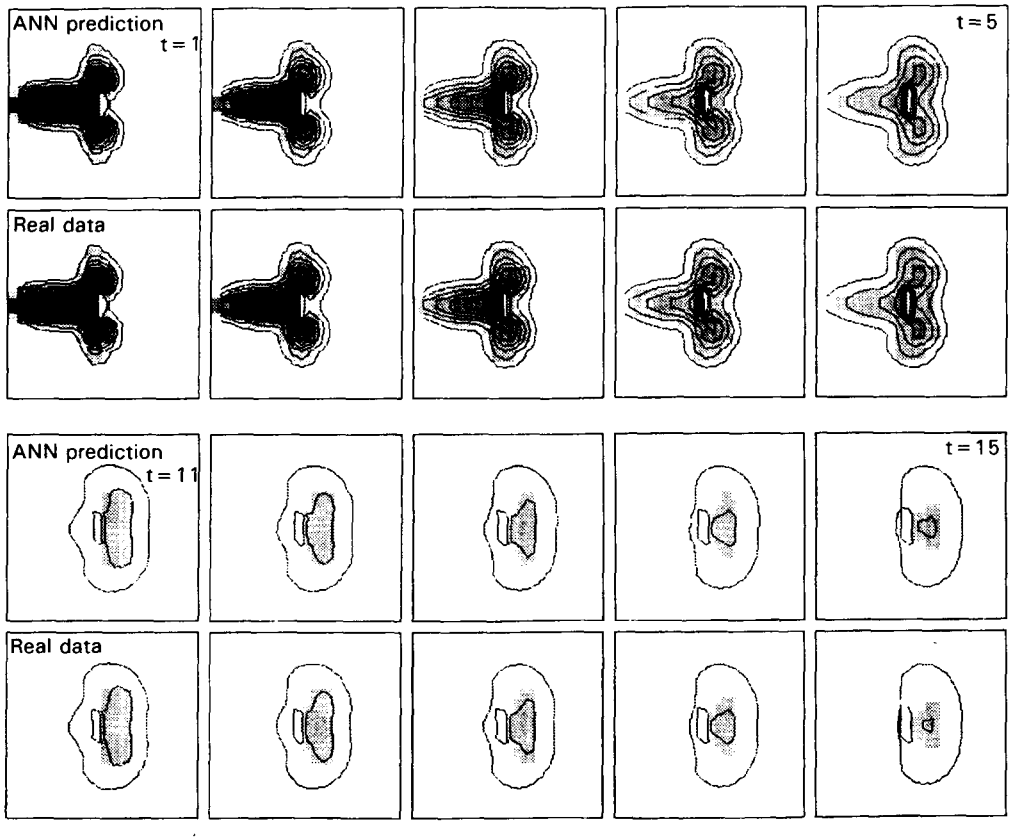

FIG. 12. A comparison of the real and (ANN) network VI-simulated fluid flow in the first of the CFD models (shown in Fig. 4a). The pattern of fluid temperature is shown by means of a grey scale with superimposed isothermal contours. The two rows of panels presented on top are for time steps 1–5, and those below are for time steps 11–15.

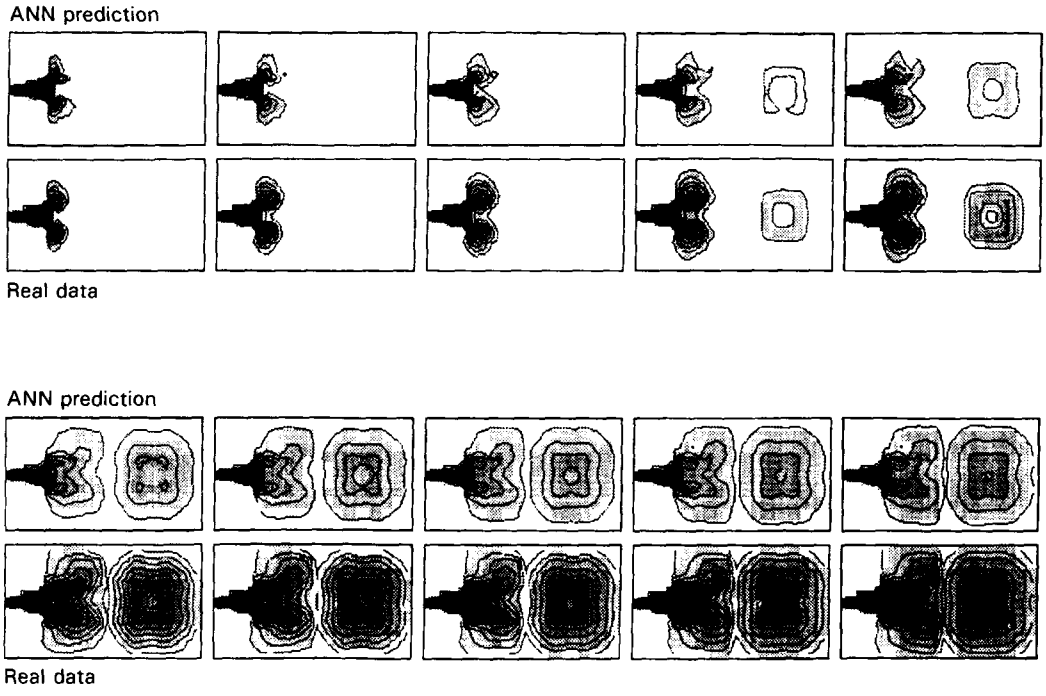

FIG. 13. A comparison of the real and (ANN) network VII-simulated fluid flow in the second of the CFD models (shown in Fig. 4b). Details are as described for Fig. 12.

The amount of information required to train a network successfully will be affected by the number of variables within that network that are adjusted during training. If the number of input and output neurons is not changed, this depends solely on the level of connectivity. With the simplest connection paradigm, complete connectivity, networks are not successfully trained, even when provided with a high level of information in the training set (data not shown). The move to local connectivity reduced the number of connections in a 12 × 20 network from 57 600 to 1972 (that is, almost 30-fold) and was made on the assumption that the behaviour at a given point in the modelled system is determined only by the state of the system around that point. This was valid for the cellular automata model of aerosol flow, and should remain valid to varying extents for other aerosol systems. In real physical systems, however, the extent of the region considered as "local" to a particular point may be larger than the eight other points in its immediate vicinity. In the lung, for example, features such as the laryngeal jet (Heyder & Rudolf 1984) mean that flow in the larynx can affect aerosol deposition much deeper down the bronchial tree.

Such considerations prompted the adoption of weight decay and connection pruning, by which connectivity is dynamically adjusted during training. The results obtained using this method were acceptable, although slightly poorer than those from networks with imposed local connectivity. It is interesting to note that the change from 3% connectivity in the purely local scheme to 6% connectivity "discovered" by connection pruning, abolishes the ANNs capacity for perfect learning of the rules governing flow in the cellular automata. The need to start with full connectivity also increased the amount of computer time required to train a network. This dynamic connection modification process is, however, the only appropriate way to work towards local connectivity when the aerosols being investigated are more complex than our initial cellular automata model, as evinced by our later work.

Our investigations have also enabled us to develop methods of assessing the quality of network training. When the system being investigated is not fully understood, or when more complicated network structures are used, examination of internal connection weights will not be of use. Here, however, we have demonstrated that the comparison of network simulations of flow with experimental data is an appropriate method of assessment. Other measures of the progress of training, such as the mean square error, can only indicate whether a network has been trained — they do not reveal whether it is correctly trained. A solution that works only for the data presented in the training set (giving a network that can memorize, but not generalize) can produce a misleadingly low mean square error. In this case, simulations over the complete time course of aerosol flow will only model experimental flow when the starting state for the stimulation is the same as that of the experiment that provided the data for the training set, something which has been observed in some of the work reported here. In particular, network I yields the lowest mean square error but is the poorest performer of all the networks presented. Testing with experimental data sufficiently different from the training set is, therefore, required to be sure that a network is correctly trained.

With the usefulness of ANNs as tools for investigating aerosol flow established, training with data from CFD models tested this approach with more realistic data. In the first of the CFD studies, the flow phenomena studied were essentially two-dimensional despite the three-dimensional nature of the experimental system with data taken from a plane of symmetry running through the experimental space. The processes

involved in flow were more complex, involving turbulence in addition to bulk flow and diffusion.

The second CFD study moved closer toward true three-dimensional modelling. Predicting aerosol behaviour at every grid point in our experimental model would have been prohibitive, involving as it would 4096 data points (and some 16·8 million connections in a fully connected ANN). Instead, data were taken from two perpendicular planes in the experimental device, with the network being to able to use cues from the longitudinal plane to predict the arrival of the aerosol cloud at the second, transverse plane.

In both CFD models, ANNs appeared to be able to model aerosol flow successfully. The extent of this modelling ability is not investigated here; the only variable in all the CFD experiments was temperature. The ability of an ANN to predict the effect of changes in jet velocity has not been measured, and further work with such changes is needed. More complex systems would undoubtedly require an increase in the complexity of the networks used to model them, with a corresponding increase in the amount of information required to train such networks. As the aerosol flows being modelled become more realistic, obtaining training data with enough variability and information content is likely to become more difficult.

Despite these reservations, however, the successful use of ANNs in modelling both simple and more stringent models of flow leaves us optimistic that this approach is worthy of further exploration and may be useful in the modelling of aerosol flow in generation devices and perhaps even in the human lung.

Acknowledgement

The funding for this work is provided by the Pharmacy Department at King's College London.

References

Aoyama, T., Suzuki, Y., Ichikawa, H. (1990) Neural networks applied to pharmaceutical problems. 3: Neural networks applied to quantitative structure–activity relationship analysis. J. Med. Chem. 33: 2583–2590

Bodor, N., Harget, A., Huang, M. (1991) Neural network studies. 1: Estimation of the aqueous solubility of organic compounds. J. Am. Chem. Soc. 113: 9480–9483

Bulsari, A. B., Palosaari, S. (1993) Application of neural networks for system identification of an adsorption column. Neural Comput. Appl. 1: 160–165

Cherquaoi, D., Villemin, D.(1994) Use of a neural network to determine the boiling point of alkanes. J. Chem. Soc. Faraday Trans. 90: 97–102

Elrod, D. W., Maggiora, G. M., Trenary, R. G. (1990) Applications of neural networks in chemistry. 1. Prediction of electrophilic aromatic substitution reactions. J. Chem. Inf. Comput. Sci. 30: 477–484

Gasteiger, J., Li, X., Uschold, A. (1994) The beauty of molecular surfaces as revealed by self-organizing neural networks. J. Mol. Graphics 12: 90–97

Heyder, J., Rudolf, G. (1984) Mathematical models of particle deposition in the human respiratory tract. J. Aerosol Sci. 15: 697–707

Hussain, A. S., Johnson, R. D. (1992) Development of a neural network for a qsar problem. Abs. Paper Am. Chem. Soc. 203: 45

Hussain, A. S., Yu, X., Johnson, R. D. (1991) Application of neural computing in pharmaceutical product development. Pharm. Res. 8: 1248–1252

Hussain, A. S., Johnson, R. D., Shivanand, P. (1992a) Evaluation of artificial neural network tools for response surface analysis. Pharm. Res. 9: S165

Hussain, A. S., Kane, A. K., Yu, X., Bronaugh, R. (1992b) Chemical structure–skin permeability relationships: a neural network analysis. Pharm. Res. 9: S191

Hussain, A. S., Shivanand, P., Johnson, R. D. (1992c). Computer-aided formulation design of hydrophilic matrix tablets: prediction of drug release profile. Pharm. Res. 9: S147

Hussain, A. S., Johnson, R. D., Vacharajani, N. N., Ritschel, W. A. (1993) Feasibility of developing a neural network for prediction of human pharmacokinetic parameters from animal data. Pharm. Res. 10: 466–469

Hussain, A. S., Shivanand, P., Johnson, R. D. (1994) Application of neural computing in pharmaceutical product development: computer-aided formulation design. Drug Dev. Ind. Pharm. 20: 1739–1752

Le Cun, Y., Denker, J., Solla, S. A. (1990) Optimal brain damage. In: Touretzky, D. (ed.) Advances in Neural Information Processing Systems, Volume 2. Morgan Kaufmann, San Mateo, CA, pp 598–605

Rumelhart, D. E., Hinton, G. E., Williams, R. J. (1986) Learning internal representations by error propagation. In: Rumelhart, D. E., McClelland, J. L. (eds) Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1. Foundations. MIT press, Cambridge, MA, pp 318–362

Veng-Pedersen, P., Modi, N. (1992) Neural networks in pharmacodynamic modeling – is current practice of complex kinetic systems at a dead end? J. Pharmacokin. Biopharm. 320: 397–412

Zupan, J., Gasteiger, J. (1991) Neural networks: a new method for solving chemical problems or just a passing phase? Anal. Chim. Acta 248: 1–30