Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2066279 C 2004/01/13

(11)(21) 2 066 279

(12) BREVET CANADIEN
CANADIAN PATENT

(13) C

(86) Date de dépôt PCT/PCT Filing Date: 1991/07/25
(87) Date publication PCT/PCT Publication Date: 1992/02/20
(45) Date de délivrance/Issue Date: 2004/01/13
(85) Entrée phase nationale/National Entry: 1992/03/31
(86) N° demande PCT/PCT Application No.: US 1991/005253
(87) N° publication PCT/PCT Publication No.: 1992/002864
(30) Priorité/Priority: 1990/08/03 (07/562,388) US

(51) Cl.Int.^{5}/Int.Cl.^{5} G06F 15/18, G06F 15/46, G05B 13/02

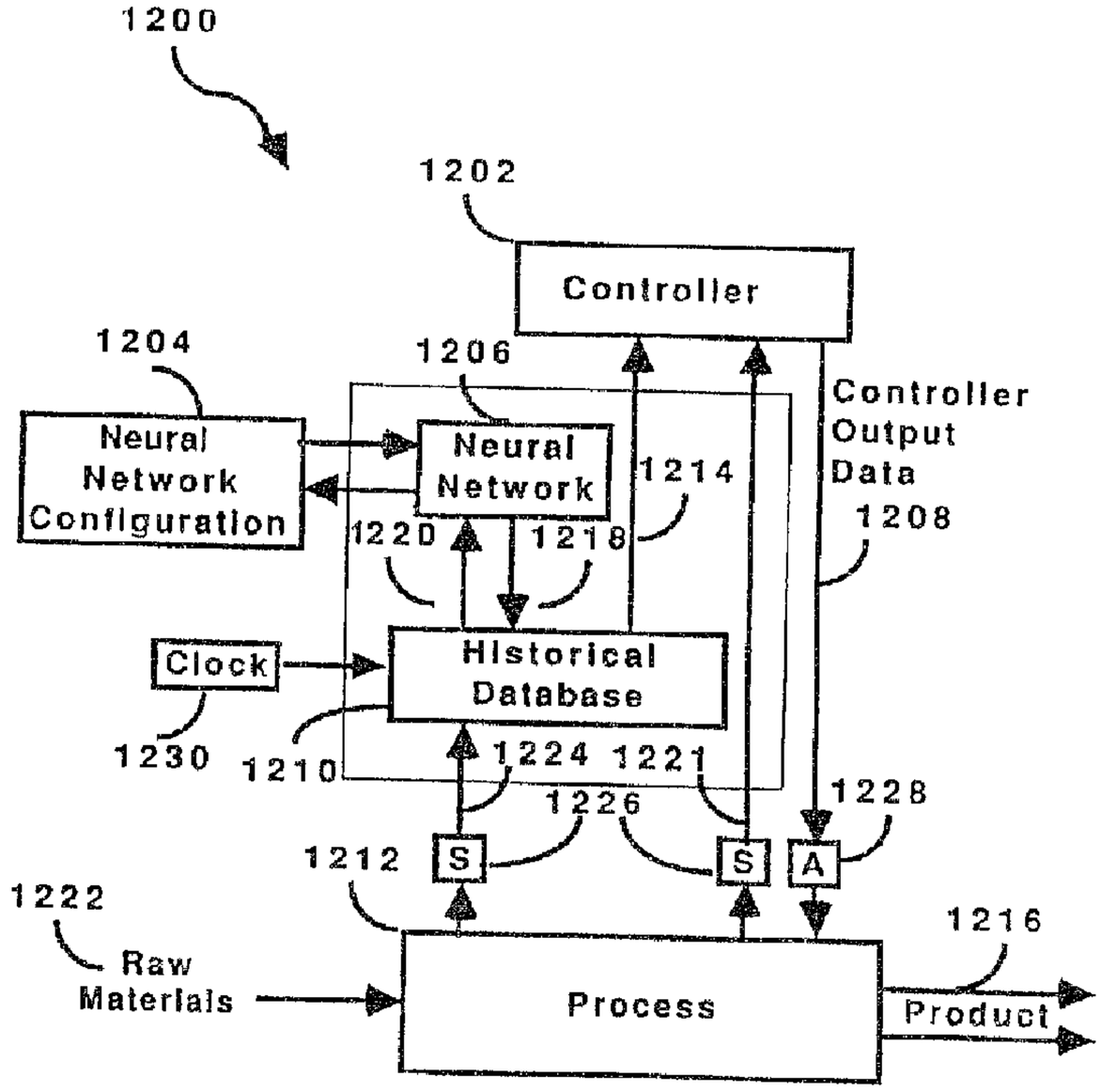
(72) Inventeur/Inventor:
SKEIRIK, RICHARD D., US

(73) Propriétaire/Owner:
PAVILION TECHNOLOGIES, INC., US

(74) Agent: BORDEN LADNER GERVAIS LLP

(54) Titre : RESEAU NEURONAL DE COMMANDE EN DIRECT DE PROCESSUS INDUSTRIEL UTILISANT DES INDICATEURS

(54) Title: ON-LINE PROCESS CONTROL NEURAL NETWORK USING DATA POINTERS



(57) Abrégé/Abstract:
An on-line process control neural network using data pointers allows the neural network to be easily configured to use data in a process control environment. The inputs, outputs, training inputs and errors can be retrieved and/or stored from any available data source without programming. The user of the neural network specifies data pointers indicating the particular computer system in which the data resides or will be stored; the type of data to be retrieved and/or stored; and the specific data value or storage location to be used. The data pointers include maximum, minimum, and maximum change limits, which can also serve as scaling limits for the neural network. Data pointers indicating time-dependent data, such as time averages, also include time boundary specifiers. The data pointers are entered by the user of the neural network using pop-up menus and by completing fields in a template. A historical database provides both a source of input data and a storage function for output and error data.

2066279

PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 :
G05B 13/02, G06F 15/76

A1

(11) International Publication Number: WO 92/02864

(43) International Publication Date: 20 February 1992 (20.02.92)

(21) International Application Number: PCT/US91/05253

(22) International Filing Date: 25 July 1991 (25.07.91)

(30) Priority data:
562,388 3 August 1990 (03.08.90) US

(71) Applicant: E.I. DU PONT DE NEMOURS & CO. (INC.) [US/US]; 1007 Market Street, Wilmington, DE 19898 (US).

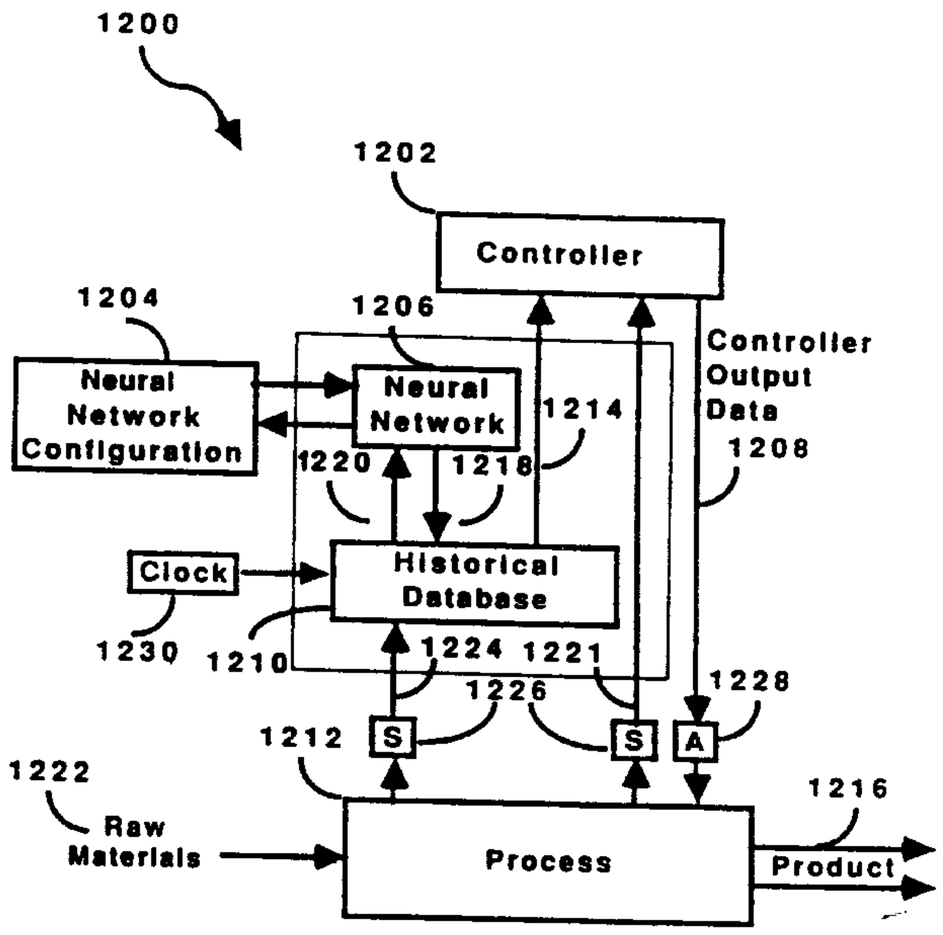
(72) Inventor: SKEIRIK, Richard, D. ; 11 Beech Hill Drive, Newark, DE 19711 (US).

(74) Agents: STERNE, Robert, Greene et al.; Sterne, Kessler, Goldstein & Fox, 1225 Connecticut Avenue, N.W., Suite 300, Washington, DC 20036 (US).

(81) Designated States: AT (European patent), BE (European patent), CA, CH (European patent), DE (European patent), DK (European patent), ES (European patent), FR (European patent), GB (European patent), GR (European patent), IT (European patent), LU (European patent), NL (European patent), SE (European patent).

Published
With international search report.

(54) Title: ON-LINE PROCESS CONTROL NEURAL NETWORK USING DATA POINTERS



(57) Abstract

An on-line process control neural network using data pointers allows the neural network to be easily configured to use data in a process control environment. The inputs, outputs, training inputs and errors can be retrieved and/or stored from any available data source without programming. The user of the neural network specifies data pointers indicating the particular computer system in which the data resides or will be stored; the type of data to be retrieved and/or stored; and the specific data value or storage location to be used. The data pointers include maximum, minimum, and maximum change limits, which can also serve as scaling limits for the neural network. Data pointers indicating time-dependent data, such as time averages, also include time boundary specifiers. The data pointers are entered by the user of the neural network using pop-up menus and by completing fields in a template. A historical database provides both a source of input data and a storage function for output and error data.

ON-LINE PROCESS CONTROL NEURAL NETWORK USING DATA POINTERS

RELATED APPLICATIONS

The following related U.S. patents name Richard D. Skeirik as the sole or one of the inventors:

U.S. Patent No. 4,920,499, issued April 24, 1990;
U.S. Patent No. 4,884,217 , issued November 28, 1989;
U.S. Patent No. 4,907,167, issued March 6, 1990;
U.S. Patent No. 4,910,691, issued March 20, 1990;
U.S. Patent No. 5,006,992, issued April 9, 1991;
U.S. Patent No. 4,965,742, issued October 23, 1990;
U.S. Patent No. 5,058,043, issued October 15, 1991.

BACKGROUND OF THE INVENTION

I. Field of the Invention

The present invention relates generally to monitoring and control of manufacturing processes, particularly chemical processes, and more specifically, to neural networks used in process control of such processes.

II. Related Art

Quality of products is increasingly important. The control of quality and the reproducibility of quality are the focus of many efforts. For example, in Europe, quality is the focus of the ISO (International Standards Organization,

2066279

Geneva, Switzerland) 9000 standards. These rigorous standards provide for quality assurance in production, installation, final inspection, and testing. They also provide guidelines for quality assurance between a supplier and customer. These standards are expected to become an effective requirement for participation in the EC (European Community) after the removal of trade barriers in 1992.

The quality of a manufactured product is a combination of all of the properties of the product which affect its usefulness to its user. Process control is the collection of methods used to produce the best possible product properties in a manufacturing process.

Process control is very important in the manufacture of products. Improper process control can result in a product which is totally useless to the user, or in a product which has a lower value to the user. When either of these situations occur, the manufacturer suffers (1) by paying the cost of manufacturing useless products, (2) by losing the opportunity to profitably make a product during that time, and (3) by lost revenue from reduced selling price of poor products. In the final analysis, the effectiveness of the process control used by a manufacturer can determine whether the manufacturer's business survives or fails.

A. Quality and Process Conditions

Figure 19 shows, in block diagram form, key concepts concerning products made in a manufacturing process. Referring now to Figure 19, raw materials 1222 are processed under (controlled) process conditions 1906 in a process 1212 to produce a product 1216 having product properties 1904. Examples of raw materials 1222, process conditions 1906, and product properties 1904 are shown in Figure 19. It should be

understood that these are merely examples for purposes of illustration.

Figure 20 shows a more detailed block diagram of the various aspects of the manufacturing of products 1216 using process 1212. Referring now to Figures 19 and 20, product 1216 is defined by one or more product property aim value(s) 2006 of its product properties 1904. The product property aim values 2006 of the product properties 1904 are those which the product 1216 needs to have in order for it to be ideal for its intended end use. The objective in running process 1212 is to manufacture products 1216 having product properties 1904 which are exactly at the product property aim value(s) 2006.

The following simple example of a process 1212 is presented merely for purposes of illustration. The example process 1212 is the baking of a cake. Raw materials 1222 (such as flour, milk, baking powder, lemon flavoring, etc.) are processed in a baking process 1212 under (controlled) process conditions 1906. Examples of the (controlled) process conditions 1906 are: mix batter until uniform, bake batter in a pan at a preset oven temperature for a preset time, remove baked cake from pan, and allow removed cake to cool to room temperature.

The product 1216 produced in this example is a cake having desired properties 1904. For example, these desired product properties 1904 can be a cake that is fully cooked but not burned, brown on the outside, yellow on the inside, having a suitable lemon flavoring, etc.

Returning now to the general case, the actual product properties 1904 of product 1216 produced in a process 1212 are determined by the combination of all of the process conditions 1906 of process 1212 and the raw materials 1222 that are utilized. Process conditions 1906 can be, for

2066279

example, the properties of the raw materials 1222, the speed at which process 1212 runs (also called the production rate of the process 1212), the process conditions 1906 in each step or stage of the process 1212 (such as temperature, pressure, etc.), the duration of each step or stage, and so on.

B. Controlling Process Conditions

Figure 20 shows a more detailed block diagram of the various aspects of the manufacturing of products 1216 using process 1212. Figures 19 and 20 should be referred to in connection with the following description.

To effectively operate process 1212, the process conditions 1906 must be maintained at one or more process condition setpoint(s) or aim value(s) (called a regulatory controller setpoint(s) in the example of Figure 14) 1404 so that the product 1216 produced will have the product properties 1904 matching the desired product property aim value(s) 2006. This task can be divided into three parts or aspects for purposes of explanation.

In the first part or aspect, the manufacturer must set (step 2008) initial settings of the process condition setpoint(s) (or aim value(s)) 1404 in order for the process 1212 to produce a product 1216 having the desired product property aim values 2006. Referring back to the example set forth above, this would be analogous to deciding to set the temperature of the oven to a particular setting before beginning the baking of the cake batter.

The second step or aspect involves measurement and adjustment of the process 1212. Specifically, process conditions 1906 must be measured to produce a process condition measurement(s) 1224. The process condition measurement(s) 1224 must be used to generate adjustment(s)

1208 (called controller output data in the example of Figure 12) to a controllable process state(s) 2002 so as to hold the process conditions 1906 as close as possible to process condition setpoint 1404. Referring again to the example above, this is analogous to the way the oven measures the temperature and turns the heating element on or off so as to maintain the temperature of the oven at the desired temperature value.

The third stage or aspect involves holding a product property measurement(s) of the product properties 1904 as close as possible to the product property aim value(s) 2006. This involves producing product property measurement(s) 1304 based on the product properties 1904 of the product 1216. From these measurements, adjustment to process condition setpoint 1402 to the process condition setpoint(s) 1404 must be made so as to maintain process condition(s) 1906. Referring again to the example above, this would be analogous to measuring how well the cake is baked. This could be done, for example, by sticking a toothpick into the cake and adjusting the temperature during the baking step so that the toothpick eventually comes out clean.

It should be understood that the previous description is intended only to show the general conditions of process control and the problems associated with it in terms of producing products of predetermined quality and properties. It can be readily understood that there are many variations and combinations of tasks that are encountered in a given process situation. Often, process control problems can be very complex.

One aspect of a process being controlled is the speed with which the process responds. Although processes may be very complex in their response patterns, it is often helpful to define a time constant for control of a process. The time

2066279

constant is simply an estimate of how quickly control actions must be carried out in order to effectively control the process.

In recent years, there has been a great push towards the automation of process control. The motivation for this is that such automation results in the manufacture of products of desired product properties where the manufacturing process that is used is too complex, too time-consuming, or both, for people to deal with manually.

Thus, the process control task can be generalized as being made up of five basic steps or stages as follows:

(1) the initial setting 2008 of process condition setpoint(s);

(2) producing process condition measurement(s) 1224 of the process condition(s) 1906;

(3) adjusting 1208 controllable process state(s) 2002 in response to the process condition measurement(s) 1224;

(4) producing product property measurement(s) 1304 based on product properties 1904 of the manufactured product 1216; and

(5) adjusting 1402 process condition setpoint(s) 1404 in response to the product property measurements 1304.

The explanation which follows explains the problems associated with meeting and optimizing these five steps.

C. <u>The Measurement Problem</u>

As shown above, the second and fourth steps or aspects of process control involve measurement 1224 of process conditions 1906 and measurement 1304 of product properties 1904. Such measurements are sometimes very difficult, if not impossible, to effectively perform for process control.

For many products, the important product properties 1904 relate to the end use of the product and not to the process conditions 1906 of the process 1212. One illustration of this involves the manufacture of carpet fiber. An important product property 1904 of carpet fiber is how uniformly the fiber accepts the dye applied by the carpet maker. Another example involves the cake example set forth above. An important product property 1904 of a baked cake is how well the cake resists breaking apart when the frosting is applied. Typically, the measurement of such product properties 1904 is difficult and/or time consuming and/or expensive to make.

An example of this problem can be shown in connection with the carpet fiber example. The ability of the fiber to uniformly accept dye can be measured by a laboratory (lab) in which dye samples of the carpet fiber are used. However, such measurements can be unreliable. For example, it may take a number of tests before a reliable result can be obtained. Furthermore, such measurements can also be slow. In this example, it may take so long to conduct the dye test that the manufacturing process can significantly change and be producing different product properties 1904 before the lab test results are available for use in controlling the process 1212.

It should be noted, however, that some process condition measurements 1224 are inexpensive, take little time, and are quite reliable. Temperature typically can be measured easily, inexpensively, quickly, and reliably. For example, the temperature of the water in a tank can often be easily measured. But oftentimes process conditions 1906 make such easy measurements much more difficult to achieve. For example, it may be difficult to determine the level of a foaming liquid in a vessel. Moreover, a corrosive process

may destroy measurement sensors, such as those used to measure pressure.

Regardless of whether or not measurement of a particular process condition 1906 or product property 1904 is easy or difficult to obtain, such measurement may be vitally important to the effective and necessary control of the process 1212. It can thus be appreciated that it would be preferable if a direct measurement of a specific process condition 1906 and/or product property 1904 could be obtained in an inexpensive, reliable, short time period and effective manner.

D. Conventional Computer Models as Predictors of Desired Measurements

As stated above, the direct measurement of the process conditions 1906 and the product properties 1904 is often difficult, if not impossible, to do effectively.

One response to this deficiency in process control has been the development of computer models (not shown) as predictors of desired measurements. These computer models are used to create values used to control the process 1212 based on inputs that are not identical to the particular process conditions 1906 and/or product properties 1904 that are critical to the control of the process 1212. In other words, these computer models are used to develop predictions (estimates) of the particular process conditions 1906 or product properties 1904. These predictions are used to adjust the controllable process state 2002 or the process condition setpoint 1404.

Such conventional computer models, as explained below, have limitations. To better understand these limitations and how the present invention overcomes them, a brief description of each of these conventional models is set forth.

1. Fundamental Models

A computer-based fundamental model (not shown) uses known information about the process 1212 to predict desired unknown information, such as product conditions 1906 and product properties 1904. A fundamental model is based on scientific and engineering principles. Such principles include the conservation of material and energy, the equality of forces, and so on. These basic scientific and engineering principles are expressed as equations which are solved mathematically or numerically, usually using a computer program. Once solved, these equations give the desired prediction of unknown information.

Conventional computer fundamental models have significant limitations, such as:

(1) They are difficult to create since the process 1212 must be described at the level of scientific understanding, which is usually very detailed;

(2) Not all processes 1212 are understood in basic engineering and scientific principles in a way that can be computer modeled;

(3) Some product properties 1904 are not adequately described by the results of the computer fundamental models; and

(4) The number of skilled computer model builders is limited, and the cost associated with building such models is thus quite high.

These problems result in computer fundamental models being practical only in some cases where measurement is difficult or impossible to achieve.

2066279

2. Empirical Statistical Models

Another conventional approach to solving measurement problems is the use of a computer-based statistical model (not shown).

Such a computer-based statistical model uses known information about the process 1212 to determine desired information that cannot be effectively measured. A statistical model is based on the correlation of measurable process conditions 1906 or product properties 1904 of the process 1212.

To use an example of a computer-based statistical model, assume that it is desired to be able to predict the color of a plastic product 1216. This is very difficult to measure directly, and takes considerable time to perform. In order to build a computer-based statistical model which will produce this desired product property 1904 information, the model builder would need to have a base of experience, including known information and actual measurements of desired information. For example, known information is such things as the temperature at which the plastic is processed. Actual measurements of desired information are the actual measurements of the color of the plastic.

A mathematical relationship (also called an equation) between the known information and the desired unknown information must be created by the developer of the empirical statistical model. The relationship contains one or more constants (which are assigned numerical values) which affect the value of the predicted information from any given known information. A computer program uses many different measurements of known information, with their corresponding actual measurements of desired information, to adjust these constants so that the best possible prediction results are

achieved by the empirical statistical model. Such computer program, for example, can use non-linear regression.

Computer-based statistical models can sometimes predict product properties 1904 which are not well described by computer fundamental models. However, there are significant problems associated with computer statistical models, which include the following:

(1) Computer statistical models require a good design of the model relationships (that is, the equations) or the predictions will be poor;

(2) Statistical methods used to adjust the constants typically are difficult to use;

(3) Good adjustment of the constants cannot always be achieved in such statistical models; and

(4) As is the case with fundamental models, the number of skilled statistical model builders is limited, and thus the cost of creating and maintaining such statistical models is high.

The result of these deficiencies is that computer-based empirical statistical models are practical in only some cases where the process conditions 1906 and/or product properties 1904 cannot be effectively measured.

E. Deficiencies in the Related Art

As set forth above, there are considerable deficiencies in conventional approaches to obtaining desired measurements for the process conditions 1906 and product properties 1904 using conventional direct measurement, computer fundamental models, and computer statistical models. Some of these deficiencies are as follows:

(1) Product properties 1904 are often difficult to measure.

2066279

(2) Process conditions 1906 are often difficult to measure.

(3) Determining the initial value or settings of the process conditions 1906 when making a new product 1216 is often difficult.

(4) Conventional computer models work only in a small percentage of cases when used as substitutes for measurements.

These and other deficiencies in the conventional technology are overcome by the system and method of the present invention.

SUMMARY OF THE INVENTION

The present invention is an on-line process control neural network system and method using data pointers. The neural network can easily be configured in a process control environment to use data in process data computer systems as its input data and training input data, and to store its output data and error data in the same or different process data computer systems. The data pointers have a specification function and a storage/retrieval function.

The specification function of the data pointer allows the user of the neural network to specify the source or destination of the data. A three-part pointer with additional qualifiers is used. The first part of the pointer indicates which system the data will come from or go to. An historical database can be included in the neural network to provide a time history of data, and can be indicated with the data system pointer. Other computer systems, such as distributed control systems, can also be indicated.

The second part of the data pointer specifies what type of data will be used. Often, current or most recently

available data is specified. For other purposes, historical data such as time-weighted averages or slopes can be specified. Also, parameters such as control loop setpoints, supervisory controller limits, and so forth can be specified.

The third part of the data pointer indicates which particular value is to be used. Measurement or variable numbers, control loop numbers, measurement tag names, and so forth can be specified.

Data pointers also include limit values, which can function as sanity checks or as scaling limits for the neural network. Maximum and minimum values can be specified which indicate the expected range of the data. Maximum change amount can be specified which indicates the maximum amount that a data value is expected to change from one cycle of the neural network to the next.

For time-dependent data types, such as time-weighted averages or slopes, time boundaries are specified. One time boundary is used for point values in the past. Two time boundaries are used for interval data types such as averages or slopes.

The data pointers are specified by the user of the neural network using a template approach. Specification choices from a limited set of options are made by selecting from pop-up menus which pop up from fields on the template. Specification of numbers or tag names are made by filling in fields on the template.

The storage/retrieval function of the data pointer performs the actual data transfers in the training and prediction mode of the neural network. The specifications in the data pointer are used to select the data storage/retrieval function to be used, and to provide the specific parameters to that function. The storage/retrieval function also performs scaling of the data, and includes sanity check responses. If

2066279

the data fails outside the limits specified in the data pointer, either the data can be clamped at the limit, or the operation of the neural network can be terminated.

An historical database can be included in the neural network to provide storage and retrieval functions. The neural network can be used for process measurement, supervisory control, or regulatory control functions. The data pointers allow on-line training methods to be easily implemented. A modular approach with natural language configuration of the neural network can be used to implement the neural network. Expert system functions can be provided in the modular neural network to provide decision-making functions.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention as defined by the claims is better understood with the text read in conjunction with the following drawings:

Figure 1 is a high level block diagram of the six broad steps which make up the computer neural net process control system and method of the present invention.

Figure 2 is an intermediate block diagram of important steps and modules which make up the store input data and training input data step and module 102 of Figure 1.

Figure 3 is an intermediate block diagram of the important steps and modules which make up the configure and train neural network step and module 104 of Figure 1.

Figure 4 is an intermediate block diagram of input steps and modules which make up the predict output data using neural network step and module 106 of Figure 1.

Figure 5 is an intermediate block diagram of the important steps and modules which make up the "retrain neural network" step and module 108 of Figure 1.

Figure 6 is an intermediate block diagram of the important steps and modules which make up the enable/disable control step and module 110 of Figure 1.

Figure 7 is an intermediate block diagram of the important steps and modules which make up the control process using output data step and module 112 of Figure 1.

Figure 8 is a detailed block diagram of the configure neural network step and module 302 of the relationship of Figure 3.

Figure 9 is a detailed block diagram of the new training input data? step and module 306 of Figure 3.

Figure 10 is a detailed block diagram of the train neural network step and module 308 of Figure 3.

2066279

Figure 11 is a detailed block diagram of the error acceptable? step and module 310 of Figure 3.

Figure 12 is a representation of the architecture of an embodiment of the present invention.

Figure 13 is a representation of the architecture of an embodiment of the present invention having the additional capability of using laboratory values from the historical database 1210.

Figure 14 is an embodiment of controller 1202 of Figures 12 and 13 having a supervisory controller 1408 and a regulatory controller 1406.

Figure 15 shows various embodiments of controller 1202 of Figure 14 used in the architecture of Figure 12.

Figure 16 is a modular version of block 1502 of Figure 15 showing the various different types of modules that can be utilized with a modular neural network 1206.

Figure 17 shows an architecture for block 1502 having a plurality of modular neural networks 1702-1702^n with pointers 1710-1710^n pointing to a limited set of neural network procedures 1704-1704^m.

Figure 18 shows an alternate architecture for block 1502 having a plurality of modular neural networks 1702-1702^n with pointers 1710-1710^m to a limited set of neural network procedures 1704-1704^n, and with parameter pointers 1802-1802'' to a limited set of network parameter storage areas 1806-1806^n.

Figure 19 is a high level block diagram showing the key aspects of a process 1212 having process conditions 1906 used to produce a product 1216 having product properties 1904 from raw materials 1222.

Figure 20 shows the various steps and parameters which may be used to perform the control of process 1212 to produce products 1216 from raw materials 1222.

Figure 21 shows a representative example of a fully connected feed forward neural network 1206 having an input layer 2104, a middle (hidden) layer 2108, an output layer 2110, and weights 2112 with each connection.

Figure 22 is an exploded block diagram showing the various parameters and aspects that can make up the neural network 1206.

Figure 23 is an exploded block diagram of the input data specification 2204 and the output data specification 2206 of the neural network 1206 of Figure 22.

Figure 24 is an exploded block diagram of the prediction timing control 2212 and the training timing control 2214 of the neural network 1206 of Figure 22.

Figure 25 is an exploded block diagram of various examples and aspects of controller 1202 of Figure 12.

Figure 26 is a representative computer display or "screen" of a preferred embodiment of the present invention showing part of the configuration specification of the neural network block 1206.

Figure 27 is a representative computer display or "screen" of a preferred embodiment of the present invention showing part of the data specification of the neural network block 1206.

Figure 28 is a computer screen which shows a pop-up menu for specifying the data system element of the data specification.

Figure 29 is a computer screen of the preferred embodiment showing in detail the individual items making up the data specification display of Figure 27.

2065279

Figure 30 is a detailed block diagram of an embodiment of the enable control step and module 602 of Figure 6.

Figure 31 is a very detailed block diagram of embodiments of steps and modules 802, 804 and 806 of Figure 8.

Figure 32 is a very detailed block diagram of embodiments of steps and modules 808, 810, 812 and 814 of Figure 8.

Figure 33 is a nomenclature diagram showing the present invention at a high level.

Figure 34 shows a representative example of the neural network 1206 of Figure 21 with training capability.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Table of Contents

I. Overview of Neural Networks 21
A. Construction of Neural Networks 23
B. Prediction 24
C. Neural Network Training 25
D. Advantages of Neural Networks 27

II. Brief Overview 28

III. Use in Combination with Expert Systems 30

IV. Preferred Method of Operation 32
A. Store Input Data and Training Input Data Step and Module 102 33
B. Configure and Train Neural Network Step and Module 104 35
1. Configure Neural Network Step and Module 302 38
2. Wait Training Input Data Interval Step and Module 304 46
3. New Training Input Data? Step and Module 306 46
4. Train Neural Network Step and Module 308 . 48
5. Error Acceptable ? Step and Module 310 . . 51
C. Predict Output Data Using Neural Network Step and Module 106 53
D. Retrain Neural Network Step or Module 55
E. Enable/Disable Control Module or Step 110 . . . 56
F. Control Process Using Output Data Step or Module 112 57

V. Preferred Structure (Architecture) 59

VI. User Interface 82

2066279

In describing the preferred embodiment of the present invention, reference will be made to Figure 33. This figure is a nomenclature diagram which shows the various names for elements and actions used in describing the present invention. Figure 33 is not necessarily intended to represent the method of the present invention, nor does it necessarily depict the architecture of the present invention. However, it does provide a reference point by which consistent terms can be used in describing the present invention.

In referring to Figure 33, the boxes indicate elements in the architecture and the labeled arrows indicate actions that are carried out. In addition, words that do not appear in boxes which break arrows represent information or data which is being transmitted from one element in the present invention to another.

As discussed below in greater detail, the present invention essentially utilizes neural nets to provide predicted values of important and not readily obtainable process conditions 1906 and/or product properties 1904 to be used by a controller 1202 to produce controller output data 1208 used to control the process 1212. As shown in Figure 12, a neural network 1206 operates in conjunction with an historical database 1210 which provides input sensor(s) data 1220.

Referring now to Figures 1 and 12, input data and training input data are stored in a historical database with associated timestamps as indicated by a step or module 102. In parallel, the neural network 1206 is configured and trained in a step or module 104. The neural network 1206 is used to predict output data 1218 using input data 1220, as indicated by a step or module 106. The neural network 1206 is then retrained in a step or module 108, and control using

the output data is enabled or disabled in a step or module 110. In parallel, control of the process using the output data is performed in a step or module 112. Thus, the present invention collects and stores the appropriate data, configures and trains the neural network, uses the neural network to predict output data, and enables control of the process using the predicted output data.

Central to the present invention is the neural network 1206. Various embodiments of the neural network 1206 can be utilized, and are described in detail below.

I. Overview of Neural Networks

In order to fully appreciate the various aspects and benefits produced by the present invention, a good understanding of neural network technology is required. For this reason, the following section discusses neural network technology as applicable to the neural network 1206 of the system and method of the present invention.

Artificial or computer neural networks are computer simulations of a network of interconnected neurons. A biological example of interconnected neurons is the human brain. Neural networks are computer representations of architectures which model the working of the brain. It should be understood that the analogy to the human brain is important and useful in understanding the present invention.

However, neural networks used in neural network 1206 of the present invention are computer simulations (or possibly analog devices) which provide useful predicted values based on input data provided at specified intervals.

Essentially, a neural network 1206 is a hierarchical collection of elements, each of which computes the results of an equation (transfer function or activation function). The

2066279

equation may include a threshold. Each element equation uses multiple input values, but produces only one output value. The outputs of elements in a lower level (that is, closer to the input data) are provided as inputs to the elements of higher layers. The highest layer produces the output(s).

Referring now to Figure 21, a representative example of the neural network 1206 is shown. It should be noted that the example shown in Figure 21 is merely illustrative of an embodiment of neural network 1206. As discussed below, other embodiments for neural network 1206 can be used.

The embodiment of Figure 21 has an input layer 2104, a middle (hidden) layer 2108, and an output layer 2110. Input layer 2104 includes a layer of input elements 2102 which take their input values from the external input data 1220. This is the known information used to produce the predicted values (output data) at outputs 1218. Even though input layer 2104 is referred to as a layer in the neural network 1206, input layer 2104 does not contain any processing elements; instead, it is a set of storage locations for input values on lines 2120.

The next layer is called the middle or hidden layer 2108. Such a middle layer 2108 is not required, but is usually used. It includes a set of elements 2106. The outputs from inputs 2102 of input layer 2104 are used as inputs by each element 2106. Thus, it can be appreciated that the outputs of the previous layer are used to feed the inputs of the next layer.

Additional middle layers 2108 can be used. Again, they would take the outputs from the previous layer as their inputs. Any number of middle layers 2108 can be utilized.

Output layer 2110 has a set of elements 2106. As their input values, they take the output of elements 2106 of the middle layer 2108. The outputs 1218 of elements 2106 of

output layer 2110 are the predicted values (called output data) produced by the neural net 1206 using the input data 1220.

For each input value for each element of each of the layers 2108 and 2110, an adjustable constant called a weight 2112 is defined. For purposes of illustration only, only two weights 2112 are shown. However, each connection between the layers 2104, 2108 and 2110 has an associated weight. Weights determine how much relative effect an input value has on the output value of the element in question.

When each middle element connects to all of the outputs from the previous layer, and each output element connects to all of the outputs from the previous layer, the network is called fully connected. Note that if all elements use output values from elements of a previous layer, the network is a feedforward network. The network of Figure 21 is such a fully connected, feedforward network. Note that if any element uses output values from an element in a later layer, the network is said to have feedback. Most neural networks used for neural network 1206 use the same equation in every element in the network.

A. Construction of Neural Networks

Neural network 1206 is built by specifying the number, arrangement and connection of the elements of which it is made up. In a highly structured embodiment of neural network 1206, the configuration is fairly simple. For example, in a fully connected network with one middle layer (and of course including one input and one output layer), and no feedback, the number of connections and consequently the number of weights is fixed by the number of elements in each layer. Such is the case in the example shown in Figure 21. Since the same equation is usually used in all elements, for this

2066279

type of network we need to know the number of elements in each layer. This determines the number of weights and hence total storage needed to build the network. The modular aspect of the present invention of Figure 16 takes advantage of this way of simplifying the specification of a neural network. Note that more complex networks require more configuration information, and therefore more storage.

The present invention contemplates other types of neural network configurations for use with neural network 1206. All that is required for neural network 1206 is that the neural network be able to be trained and retrained so as to provide the needed predicted values utilized in the process control.

B. Prediction

Referring now to Figure 21, a representative embodiment of a feed forward neural network will now be described. This is only illustrative of one way in which a neural network can function.

Input data 1220 is provided to input storage locations called inputs 2102. Middle layer elements 2106 each retrieve the input values from all of the inputs 2102 in the input layer 2104. Each element has a weight 2112 associated with each input value. Each element 2106 multiplies each input value 2102 times its associated weight 2112, and sums these values for all of the inputs. This sum is then used as input to an equation (also called a transfer function or activation function) to produce an output or activation for that element. The processing for elements 2106 in the middle or hidden layer 2108 can be performed in parallel, or they can be performed sequentially.

In the neural network with only one middle layer as shown in Figure 21, the output values or activations would then be computed. For each output element 2106, the output

values or activations from each of the middle elements 2106 is retrieved. Each output or activation is multiplied by its associated weight 2112, and these values are summed. This sum is then used as input to an equation which produces as its result the output data 1218. Thus using input data 1220, a neural network 1206 produces predicted values of output data 1218.

Equivalent function can be achieved using analog means.

C. Neural Network Training

The weights 2112 used in neural network 1206 are adjustable constants which determine (for any given neural network configuration) the values of the predicted output data for given input data. Neural networks are superior to conventional statistical models because neural networks can adjust these weights automatically. Thus, neural networks are capable of building the structure of the relationship (or model) between the input data 1220 and the output data 1218 by adjusting the weights 2112. While a conventional statistical model requires the developer to define the equation(s) in which adjustable constant(s) will be used, the neural network 1206 builds the equivalent of the equation(s) automatically.

Referring now to Figure 34, the present invention contemplates various approaches for training neural network 1206. One suitable approach is back propagation. Back propagation uses the error between the predicted output data 1218 and the associated training input data 1306 as provided by the training set (not shown) to determine how much to adjust the weights 2112 in the network 1206. In effect, the error between the predicted output data values and the associated training input data values is propagated back

2066279

through the output layer 2110 and through the middle layer 2108. This accounts for the name back propagation.

The correct output data values are called training input data values.

The neural network 1204 is trained by presenting it with a training set(s), which is the actual history of known input data values and the associated correct output data values. As described below, the present invention uses the historical database with its associated timestamps to automatically create a training set(s).

To train the network, the newly configured neural network is usually initialized by assigning random values to all of its weights 2112. Referring now to Figure 34, a representative embodiment of a neural network 1206 as configured for training purposes is shown. During training, the neural network 1206 uses its input data 1220 to produce predicted output data 1218 as described above under Section I.B., Prediction.

These predicted output data values 1218 are used in combination with training input data 1306 to produce error data 3404. These error data values 3404 are then propagated back through the network through the output elements 2106 and used in accordance with the equations or functions present in those elements to adjust the weights 2112 between the output layer 2110 and the middle or hidden layer 2108.

According to the back propagation method, which is illustrative of training methods that can be used for the neural network 1206, an error value for each element 2106 in the middle or hidden layer 2108 is computed by summing the errors of the output elements 2106 each multiplied by its associated weight 2112 on the connection between the middle element 2106 in the middle layer 2108 and the corresponding output elements in the output layer 2110. This estimate of

the error for each middle (hidden) layer element is then used in the same manner to adjust the weights 2112 between the input layer 2104 and the middle (hidden) layer 2108.

It can thus be seen that the error between the output data 1218 and the training input data 1306 is propagated back through the network 1206 to adjust the weights 2112 so that the error is reduced. More detail can be found in Parallel Distributed Processing, Explorations in the Microstructure of Cognition, by David E. Rumelhart and James L. McClelland, The MIT Press, Cambridge, Massachusetts, USA, 1986, and Explorations In Parallel Distributed Processing, A Handbook of Models, Programs, and Exercises, by James L. McClelland and David E. Rumelhart, The MIT Press, Cambridge, MA, 1988.

D. Advantages of Neural Networks

Neural networks are superior to computer statistical models because neural networks do not require the developer of the neural network model to create the equations which relate the known input data and training values to the desired predicted values (output data). In other words, neural network 1206 learns the relationships automatically in the training step 104.

However, it should be noted that neural network 1206 requires the collection of training input data with its associated input data, also called training set. The training set must be collected and properly formatted. The conventional approach for doing this is to create a disk file on a computer on which the neural network runs.

In the present invention, in contrast, this is done automatically using an historical database 1210 (Figure 12). This eliminates the errors and the time associated with the conventional approach. This also significantly improves the

2066279

effectiveness of the training function since it can be performed much more frequently.

II. Brief Overview

Referring to Figures 1 and 12, the present invention is a computer neural network system and method which produces predicted output data values 1218 using a trained network supplied with input data 1220 at a specified interval. The predicted data 1218 is supplied via an historical database 1210 to a controller 1202, which controls a process 1212 which produces a product 1216. In this way, the process conditions 1906 and product properties 1904 (Figures 19 and 20) are maintained at a desired quality level, even though important ones of them can not be effectively measured directly, or modeled using conventional, fundamental or conventional statistical approaches.

The present invention can be configured by a developer using a neural network configuration and module 104 step. Various parameters of the neural network can be specified by the developer by using natural language without knowledge of specialized computer syntax and training. In this way, the present invention allows an expert in the process being measured to configure the present invention without the use of a neural network expert.

Referring also to Figure 34, the neural network is automatically trained on-line using input data 1220 and associated training input data 1306 having timestamps (for example, from clock 1230). The input data and associated training input data are stored in an historical database 1210, which supplies this data 1220, 1306 to the neural network 1206 for training at specified intervals.

The (predicted) output data value 1218 produced by the neural network is stored in the historical database. The

stored output data value 1218 is supplied to the controller 1202 for controlling the process as long as the error data 1504 between the output data 1218 and the training input data 1306 is below an acceptable metric.

The error data 1504 is also used for automatically retraining the neural network. This retraining typically occurs while the neural network is providing the controller via the historical database with the output data. The retraining of the neural network results in the output data approaching the training input data as much as possible over the operation of the process. In this way, the present invention can effectively adapt to changes in the process, which can occur in a commercial application.

A modular approach for the neural network, as shown in Figure 16, is utilized to simplify configuration and to produce greater robustness. In essence, the modularity is broken out into specifying data and calling subroutines using pointers.

In configuring the neural network, as shown in Figure 22, data pointers 2204, 2206 are specified. A template approach, as shown in Figures 26 and 27, is used to assist the developer in configuring the neural network without having to perform any actual programming.

The present invention is an on-line process control system and method. The term "on-line" indicates that the data used in the present invention is collected directly from the data acquisition systems which generate this data. An on-line system may have several characteristics. One characteristic is the processing of data as the data is generated. This may also be called real-time operation. Real-time operation in general demands that data be detected, processed and acted upon fast enough to effectively respond to the situation. In a process control context, real time

means that the data can be responded to fast enough to keep the process in the desired control state.

In contrast, off-line methods can also be used. In off-line methods, the data being used was generated at some point in the past and there is no attempt to respond in a way that can effect the situation. It should be understood that while the preferred embodiment of the present invention uses an on-line approach, alternate embodiments can substitute off-line approaches in various steps or modules.

III. Use in Combination with Expert Systems

The above description of neural networks and neural networks as used in the present invention, combined with the description of the problem of making measurements in a process control environment given in the background section, illustrate that neural networks add a unique and powerful capability to process control systems. They allow the inexpensive creation of predictions of measurements that are difficult or impossible to obtain. This capability opens up a new realm of possibilities for improving quality control in manufacturing processes. As used in the present invention, neural networks serve as a source of input data to be used by controllers of various types in controlling the process.

Expert systems provide a completely separate and completely complimentary capability for process control systems. Expert systems are essentially decision-making programs which base their decisions on process knowledge which is typically represented in the form of if-then rules. Each rule in an expert system makes a small statement of truth, relating something that is known or could be known about the process to something that can be inferred from that knowledge. By combining the applicable rules, an expert

system can reach conclusions or make decisions which mimic the decision-making of human experts.

The systems and methods described in several of the United States patents and patent applications referenced above use expert systems in a control system architecture and method to add this decision-making capability to process control systems. As described in these patents and patent applications, expert systems provide a very advantageous function in the implementation of process control systems.

The present invention adds a different capability of substituting neural networks for measurements which are difficult to obtain. The advantages of the present invention are both consistent with and complimentary to the capabilities provided in the above-noted patents and patent applications using expert systems. In fact, the combination of neural network capability with expert system capability in a control system provides even greater benefits that either capability provided alone. For example, a process control problem may have a difficult measurement and also require the use of decision-making techniques in structuring or implementing the control response. By combining neural network and expert system capabilities in a single control application, greater results can be achieved than using either technique alone.

It should thus be understood that while the present invention relates primarily to the use of neural networks for process control, it can very advantageously be combined with the expert system inventions described in the above-noted patents and patent applications to give even greater process control problem solving capability. As described below, when implemented in the modular process control system architecture, neural network functions are easily combined

with expert system functions and other control functions to build such integrated process control applications. Thus, while the present invention can be used alone, it provides even greater value when used in combination with the expert system inventions in the above-noted patents and patent applications.

IV. Preferred Method of Operation

The preferred method of operation of the present invention stores input data and training data, configures and trains a neural network, predicts output data using the neural network, retrains the neural network, enables or disables control using the output data, and controls the process using output data. As shown in Figure 1, more than one step or module is carried out in parallel in the method of the present invention. As indicated by the divergent order pointer 120, the first two steps or modules in the present invention are carried out in parallel. First in a step or module 102, input data and training input data are stored in the historical database with associated timestamps. In parallel, the neural network is configured and trained in a step 104. Next, two series of steps or modules are carried out in parallel as indicated by the order pointer 122. First, in a step or module 106, the neural network is used to predict output data using input data stored in the historical database. Next, in a step or module 108, the neural network is retrained using training input data stored in the historical database. Next, in a step or module 110, control using the output data is enabled or disabled in parallel in a step or module 112, control of the process using the output data is carried out when enabled by step or module 110.

A. Store Input Data and Training Input Data Step and Module 102

As shown in Figure 1, an order pointer 120 indicates that a step 102 and a step 104 are performed in parallel. Referring now to step 102, it is denominated as the store input data and training input data step and module. Figure 2 shows step and module 102 in more detail.

Referring now to Figures 1 and 2, step and module 102 has the function of storing input data 1220 and storing training input data 1306. Both types of data are stored in an historical database 1210 (see Figure 12 and related structure diagrams), for example. Each stored input data and training input data entry in historical database 1210 utilizes an associated timestamp. The associated timestamp allows the system and method of the present invention to determine the relative time that the particular measurement or predicted value or measured value was taken, produced or derived.

A representative example of step and module 102 is shown in Figure 2, which is described as follows. The order pointer 120, as shown in Figure 2, indicates that input data 1220 and training input data 1306 are stored in parallel in the historical database 1210. Specifically, input data from sensors 1226 (see Figures 12 and 13) are produced by sampling at specific time intervals the sensor signal 1224 provided at the output of the sensor 1226. This sampling produces an input data value or number or signal. Each of these is called an input data 1220 as used in this application. The input data is stored with an associated timestamp in the historical database 1210, as indicated by a step and module 202. The associated timestamp that is stored in the historical database with the input data indicates the time at which the input data was produced, derived, calculated, etc.

2066279

A step or module 204 shows that the next input data value is stored by step 202 after a specified input data storage interval has lapsed or timed out. This input data storage interval realized by step and module 204 can be set at any specific value. Typically, it is selected based on the characteristics of the process being controlled.

As shown in Figure 2, in addition to the sampling and storing of input data at specified input data storage intervals, training input data 1306 is also being stored. Specifically, as shown by step and module 206, training input data is stored with associated timestamps in the historical database 1210. Again, the associated timestamps utilized with the stored training input data indicate the relative time at which the training input data was derived, produced or obtained. It should be understood that this usually is the time when the process condition or product property actually existed in the process or product. In other words, since it typically takes a relatively long period of time to produce the training input data (because lab analysis and the like usually has to be performed), it is more accurate to use a timestamp which indicates the actual time when the measured state existed in the process rather than to indicate when the actual training input data was entered into the historical database. This produces a much closer correlation between the training input data 1306 and the associated input data 1220. This close correlation is needed, as is discussed in detail below, in order to more effectively train and control the system and method of the present invention.

The training input data is stored in the historical database 1210 in accordance with a specified training input data storage interval, as indicated by a step and module 208. While this can be a fixed time period, it typically is not. More typically, it is a time interval which is dictated by

when the training input data is actually produced by the laboratory or other mechanism utilized to produce the training input data 1306. As is discussed in detail herein, this often times takes a variable amount of time to accomplish depending upon the process, the mechanisms being used to produce the training data, and other variables associated both with the process and with the measurement/analysis process utilized to produce the training input data.

What is important to understand here is that the specified input data storage interval is usually considerably shorter than the specified training input data storage interval of step and module 204.

As can be seen, step and module 102 thus results in the historical database 1210 receiving values of input data and training input data with associated timestamps. These values are stored for use by the system and method of the present invention in accordance with the steps and modules discussed in detail below.

B. Configure and Train Neural Network Step and Module 104

As shown in Figure 1, the order pointer 120 shows that a configure and train neural network step and module 104 is performed in parallel with the store input data and training input data step and module 102. The purpose of step and module 104 is to configure and train the neural network 1206 (see Figure 12).

Specifically, the order pointer 120 indicates that the step and module 104 plus all of its subsequent steps and modules are performed in parallel to the step and module 102.

Figure 3 shows a representative example of the step and module 104. As shown in Figure 3, this representative

2065279

embodiment is made up of five steps and modules 302, 304, 306, 308 and 310.

Referring now to Figure 3, an order pointer 120 shows that the first step and module of this representative embodiment is a configure neural network step and module 302. Configure neural network step and module 302 is used to set up the structure and parameters of the neural network 1206 that is utilized by the system and method of the present invention. As discussed below in detail, the actual steps and modules utilized to set up the structure and perimeters of neural network 1206 are shown in Figure 8.

After the neural network 1206 has been configured in step and module 302, an order pointer 312 indicates that a wait training data interval step and module 304 occurs or is utilized. The wait training data interval step and module 304 specifies how frequently the historical database 1210 will be looked at to determine if there is any new training data to be utilized for training of the neural network 1206. It should be noted that the training data interval of step and module 304 is not the same as the specified training input data storage interval of step and module 206 of Figure 2. Any desired value for the training data interval can be utilized for step and module 304.

An order pointer 314 indicates that the next step and module is a new training input data? step and module 306. This step and module 306 is utilized after the lapse of the training data interval specified by step and module 304. The purpose of step and module 306 is to examine the historical database 1210 to determine if new training data has been stored in the historical database since the last time the historical database 1210 was examined for new training data. The presence of new training data will permit the system and method of the present invention to train the neural network

1206 if other parameters/conditions are met. Figure 9 discussed below shows a specific embodiment for the step and module 306.

An order pointer 318 indicates that if step and module 306 indicates that new training data is not present in the historical database 1210, the step and module 306 returns the operation of step and module 104 to the step and module 304.

In contrast, if new training data is present in the historical database 1210, the step and module 306 as indicated by an order pointer 316 causes the step and module 104 to move to a train neural network step and module 308. Train neural network step and module 308 is the actual training of the neural network 1206 using the new training data retrieved from the historical database 1210. Figure 10, discussed below in detail, shows a representative embodiment of the train neural network step and module 308.

After the neural network has been trained, in step and module 308, the step and module 104 as indicated by an order pointer 320 moves to an error acceptable? step and module 310. Error acceptable? step and module 310 determines whether the error data 1504 produced by the neural network 1206 is within an acceptable metric, indicating error that the neural network 1206 is providing output data 1218 that is close enough to the training input data 1306 to permit the use of the output data 1218 from the neural network 1206. In other words, an acceptable error indicates that the neural network 1206 has been "trained" as training is specified by the user of the system and method of the present invention. A representative example of the error acceptable? step and module 310 is shown in Figure 11, which is discussed in detail below.

If an unacceptable error is determined by error acceptable? step and module 310, an order pointer 322

indicates that the step and module 104 returns to the wait training data interval step and module 304. In other words, this means that the step and module 104 has not completed training the neural network 1206. Because the neural network 1206 has not yet been trained, training must continue before the system and method of the present invention can move to a step and module 106 discussed below.

In contrast, if the error acceptable? step and module 310 determines that an acceptable error from the neural network 1206 has been obtained, then the step and module 104 has trained neural network 1206. Since the neural network 1206 has now been trained, step 104 allows the system and method of the present invention to move to the steps and methods 106 and 112 discussed below.

The specific embodiments for step and module 104 are now discussed.

1. Configure Neural Network Step and Module 302

Referring now to Figure 8, a representative embodiment of the configure neural network step and module 302 is shown. This step and module allow the uses of the present invention to both configure and re-configure the neural network. Referring now to Figure 8, the order pointer 120 indicates that the first step and module is a specify training and prediction timing control step and module 802. Step and module 802 allow the person configuring the system and method of the present invention to specify the training interval(s) and the prediction timing interval(s) of the neural network 1206.

Figure 31 shows a representative embodiment of the step and module 802. Referring now to Figure 31, step and module 802 can be made up of four steps and modules 3102, 3104, 3106, and 3108. Step and module 3102 is a specify training

timing method step and module. The specify training timing method step and module 3102 allows the user configuring the present invention to specify the method or procedure that will be followed to determine when the neural network 1206 will be trained. A representative example of this is when all of the training data has been updated. Another example is the lapse of a fixed time interval. Other methods and procedures can be utilized.

An order pointer indicates that a specify training timing parameters step and module 3104 is then carried out by the user of the present invention. This step and module 3104 allows for any needed training timing parameters to be specified. It should be realized that the method or procedure of step and module 3102 can result in zero or more training timing parameters, each of which has a value. This value could be a time value, a module number (in the modular embodiment of the present invention of Figure 16), or a data pointer. In other words, the user can configure the present invention so that considerable flexibility can be obtained in how training of the neural network 1206 occurs based on the method or procedure of step and module 3102.

An order pointer indicates that once the training timing parameter(s) 3104 has been specified, a specify prediction timing method step and module 3106 is configured by the user of the present invention. This step and module 3106 specifies the method or procedure that will be used by the neural network 1206 to determine when to predict output data values 1218 after it has been trained. This is in contrast to the actual training of the neural network 1206. Representative examples of methods or procedures for step and module 3106 are execute at a fixed time interval, execute after the execution of a specific module, or execute after a

specific data value is updated. Other methods and procedures can be used.

An order indicator in Figure 31 shows that a specify prediction timing parameters step and module 3108 is then carried out by the user of the present invention. Any needed prediction timing parameters for the method or procedure of step or module 3106 can be specified. For example, the time interval can be specified as a parameter for the execute at a specific time interval method or procedure. Another example is the specification of a module identifier when the execute after the execution of a particular module method or procedure is specified. Another example is a data pointer when the updating of a data value method or procedure is used. Other operation timing parameters can be used.

Referring again to Figure 8, after the specify training and prediction timing control step and module 802 has been specified, a specify neural network size step and module 804 is carried out. This step and module 804 allows the user to specify the size and structure of the neural network 1206 that is used by the present invention.

Specifically, referring to Figure 31 again, a representative example of how the neural network size can be specified by step and module 804 is shown. An order pointer indicates that a specific number of inputs step and module 3110 allows the user to indicate the number of inputs that the neural network 1206 will have. Note that the source of the input data for the specified number of inputs has not yet been fixed by the user in the step and module 3110. Only the actual number of inputs have been specified in the step and module 3110.

Once the number of inputs have been specified in step and module 3110, the user can specify the number of middle (hidden) layer elements in the neural network 1206 by using a

step or method 3112. By middle elements it is meant that one or more internal layers 2108 of the neural network can be specified by the user. The present invention contemplates a neural network having zero or more middle layers 2108. Typically, one middle layer is used; however, two or more middle layers are contemplated.

An order pointer indicates that once the number of middle elements have been specified in step and module 3112, the number of output data from 2106 the outputs of the neural network 1206 can be specified as indicated by a step or module 3114. Note that where the outputs of the neural network 1206 are to be stored is not specified in step or module 3114. Instead, only the number of outputs are specified in this step of the present invention.

As discussed herein, the present invention contemplates any form of presently known or future developed configuration for the structure of the neural network 1206. Thus, steps or modules 3110, 3112, and 3114 can be modified so as to allow the user to specify these different configurations for the neural network 1206.

Referring again to Figure 8, once the neural network size has been specified in step and module 804, the user can specify the training and prediction modes in a step and module 806. Step and module 806 allows both the training and prediction modes to be specified. It also allows for controlling the storage of the data produced in the training and prediction modes. It also allows for data coordination to be used in training mode.

A representative example of the specific training and prediction modes step and module 806 is shown in Figure 31. It is made up of step and modules 3116, 3118, and 3120.

As shown, an order pointer indicates that the user can specify prediction and train modes in a step and module 3116.

2066279

These are yes/no or on/off settings. Since the system and method of the present invention is in the train mode at this stage in its operation, step and 3116 typically goes to its default setting of train mode only. However, it should be understood that the present invention contemplates allowing the user to independently control the prediction or train modes.

When prediction mode is enabled or "on," the neural network 1206 will predict output data values 1218 using retrieved input data values 1220, as described below. When training mode is enabled or "on," the neural network 1206 will monitor the historical database 1210 for new training data and will train using the training data, as described below.

An order pointer indicates that once the prediction and train modes have been specified in the step and module 3116, the user can specify prediction and train storage modes in a step and module 3118. These are on/off, yes/no values. They allow the user to specify whether the output data produced in the prediction and/or modes will be stored for possible later use. In some situations, the user will specify that they will not be stored, and in such a situation they will be discarded after the prediction or train mode has occurred. Examples of situations where storage may not be needed are as follows. First, if the error acceptable metric value in the train mode indicates that the output data is poor and retraining is necessary, there may be no reason to keep the output data. Another example is in the prediction mode, where the output data is not stored but is only used. Other situations may arise where no storage is warranted.

An order pointer indicates that a specify training data coordination mode step and module 3120 is then specified by the user. Oftentimes, training input data 1306 must be

correlated in some manner with input data 1220. This step and module 3120 allows the user to deal with the relatively long time period required to produce training input data 1306 from when the measured state(s) existed in the process. First, the user can specify whether the most recent input data will be used with the training data, or whether prior input data will be used with the training data. If the user specifies that prior input data is to be used, the method of determining the time of the prior input data can be specified in this step and module 3120.

Referring again to Figure 8, once the specified training and prediction modes step and module 806 has been completed by the user, steps and modules 808, 810, 812 and 814 are carried out. Specifically, the user follows a specify input data step and module 808, a specify output data step and module 810, a specify training input data step and module 812, and a specify error data step and module 814. Essentially, these four steps and modules 808-814 allow the user to specify the source and destination of input and output data for both the (run) prediction and training modes, and the storage location of the error data determined in the training mode.

Figure 32 shows a representative embodiment used for all of the steps and modules 808-814 as follows.

Steps and modules 3202, 3204 and 3206 essentially are directed to specifying the data location for the data being specified by the user. In contrast, steps and modules 3208-3216 may be optional in that they allow the user to specify certain options or sanity checks that can be performed on the data as discussed below in more detail.

Turning first to specifying the storage location of the data being specified, a step or module 3202 is called specify data system. Typically, in a chemical plant, there is more

than one computer system utilized with a process being controlled. Step or module 3202 allows for the user to specify which computer system(s) contains the data or storage location that is being specified.

Once the data system has been specified, the user can specify the data type using a specify data type step and module 3204. The data type indicates which of the many types of data and/or storage modes are desired. Examples are current (most recent) values of measurements, historical values, time averaged values, setpoint values, limits, etc. After the data type has been specified, the user can specify a data item number or identifier using a step or module 3206. The data item number or identifier indicates which of the many instances of the specify data type in the specified data system is desired. Examples are the measurement number, the control loop number, the control tag name, etc. These three steps and modules 3202-3206 thus allow the user to specify the source or destination of the data (used/produced by the neural network) being specified.

Once this has been specified, the user can specify the following additional parameters. Specifically, where data is being specified which is time varying, the user can specify the oldest time interval boundary using a step and module 3208, and can specify the newest time interval boundary using a step and module 3210. For example, these boundaries can be utilized where a time weighted average of a specified data value is needed. Alternatively, the user can specify one particular time when the data value being specified is an historical data point value.

Sanity checks on the data being specified can be specified by the user using steps and modules 3212, 3214 and 3216 as follows. Specifically, the user can specify a high limit value using a step and module 3212, and can specify a

low limit value using a step and module 3214. Since sensors, for example, sometimes fail, this sanity check allows the user to prevent the system and method of the present invention from using false data from a failed sensor. Other examples of faulty data can also be detected by setting these limits.

The high and low limit values can be used for scaling the input data. Neural networks are typically trained and operated using input, output and training input data scaled within a fixed range. Using the high and low limit values allows this scaling to be accomplished so that the scaled values use most of the range. Typical ranges are 0 to 1 and -1 to 1.

In addition, the user often knows that certain values will normally change a certain amount over a specific time interval. Thus, changes which exceed these limits can be used as an additional sanity check. This can be accomplished by the user specifying a maximum change amount in step and module 3216.

Sanity checks can be used in the method of the present invention to prevent erroneous training, prediction, and control. Whenever any data value fails to pass the sanity checks, the data may be clamped at the limit(s), or the operation/control may be disabled. These tests significantly increase the robustness of the present invention.

It should be noted that these steps and modules in Figure 32 apply to the input, output, training input, and error data steps and modules 808, 810, 812 and 814.

When the neural network is fully configured, the weights are normally set to random values in their allowed ranges (-1 to 1 is commonly used as a weight range). This can be done automatically, or it can be performed on demand by the user (for example, using softkey 2616 in Figure 26).

2066279

2. Wait Training Input Data Interval Step and Module 304

Referring again to Figure 3, the wait training data interval step and module 304 is now described in greater detail.

Typically, the wait training input data interval is much shorter than the time period (interval) when training input data becomes available. This wait training input data interval determines how often the training input data will be checked to determine whether new training input data has been received. Obviously, the more frequently the training input data is checked, the shorter the time interval will be from when new training input data becomes available to when retraining has occurred.

It should be noted that the configuration for the neural network 1206 and specifying its wait training input data interval is done by the user. This interval may be inherent in the software system and method which contains the neural network of the present invention. Preferably, it is specifically defined by the entire software system and method of the present invention. Now the neural network 1206 is being trained.

3. New Training Input Data? Step and Module 306

An order pointer 314 indicates that once the wait training input data interval 304 has elapsed, the new training input data? step or module 306 occurs.

Figure 9 shows a representative embodiment of the new training input data? step and module 306. Referring now to Figure 9, a representative example of determining whether new training input data has been received is shown. A retrieve current training input timestamp from historical database

step and module 902 first retrieves from the historical database 1210 the current training input data timestamp(s). As indicated by an order pointer, a compare current training input data timestamp to stored training input data timestamp step and module 904 compares the current training input data timestamp(s) with a saved training input data timestamp(s). Note that when the system and method of the present invention is first started, an initialization value must be used for the saved training input data timestamp. If the current training input data timestamp is the same as the saved training input data timestamp, this indicates that new training input data does not exist. This situation on no new training input data is indicated by an order pointer 318.

This step and module 904 functions to determine whether any new training input data is available for use in training the neural network. It should be understood that, in various embodiments of the present invention, the presence of new training input data may be detected (determined) in alternate ways. One specific example is where only one storage location is available for training input data and the associated timestamp. In this case, detecting (determining) the presence of new training input data can be carried out by saving internally in the neural network the associated timestamp of the training input data from the last time the training input data was checked, and periodically retrieving the timestamp from the storage location for the training input data and comparing it to the internally saved value of the timestamp. Other distributions and combinations of storage locations for timestamps and/or data values can be used in detecting (determining) the presence of new training input data.

However, if the comparison of step and module 904 indicates that the current training input data timestamp is

different from the saved training input data timestamp, this indicates that new training input data has been received (detected). This new training input data timestamp is saved by a save current training input data timestamp step and module 906. After this current timestamp of training input data has been saved, the new training data? step and module 306 has been completed, and the present invention moves to the train neural network step and module 308 of Figure 3 as indicated by the order pointer.

4. Train Neural Network Step and Module 308

Referring again to Figure 3, the train neural network step and module 308 is the step and module where the neural network 1206 is trained. Figure 10 shows a representative embodiment of the train neural network step and module 308.

Referring now to step and module 308 shown in Figure 10, an order pointer 316 indicates that a retrieve current training input data from historical database step and module 1002 occurs. In step and module 1002, one or more current training input data values are retrieved from the historical database 1210. The number of current training input data values that is retrieved is equal to the number of outputs 2106 of the neural network 1206 that is being trained. The training input data is normally scaled. This scaling can use the high and low limit values specified in the configure and train neural network step 104.

An order pointer shows that a choose training input data time step and module 1004 is next carried out. Typically, when there are two or more current training input data values that are retrieved, the data time (as indicated by their associated timestamps) for them is different. The reason for this is that typically the sampling schedule used to produce the training input data is different for the various training

input data. Thus, current training input data often has different associated timestamps. In order to resolve these differences, certain assumptions have to be made. In certain situations, the average between the timestamps is used. Alternately, the timestamp of one of the current training input data could be used. Other approaches also can be employed.

Once the training input data time has been chosen in step and module 1004, the input data at the training input data time is retrieved from the historical database 1210 as indicated by a step and module 1006. The input data is normally scaled. This scaling can use the high and low limit values specified in the configure and train neural network step 104. Thereafter, the neural net 1206 predicts output data from the retrieved input data, as indicated by a step and module 406.

The predicted output data from the neural network 1206 is then stored in the historical database 1210, as indicated by a step and module 408. The output data is normally produced in a scaled form, since all the input and training input data is scaled. In this case, the output data must be de-scaled. This de-scaling can use the high and low limit values specified in the configure and train neural network step 104. Thereafter, error data is computed using the output data from the neural network 1206 and the training input data, as indicated by a step and module 1012. It should be noted that the term error data 1504 as used in step and module 1012 is a set of error data value for all of the predicted outputs 2106 from the neural network 1206. However, the present invention also contemplates using a global or cumulative error data for evaluating whether the predicted output data values are acceptable.

2066279

After the error data 1504 has been computed (calculated) in the step and module 1012, the neural network 1206 is retrained using the error data 1504 and/or the training input data 1306. The present invention contemplates any method of training the neural network 1306.

After the training step and module 1014 has been completed, the error data 1504 is stored in the historical database 1210 in a step and module 1016. It should be noted that the error data 1504 shown here is the individual data for each output 2106. These stored error data 1504 provide a historical record of the error performance for each output 2106 of the neural network 1206.

The sequence of steps described above is the preferred embodiment used when the neural network 1206 can be effectively trained using a single presentation of the training set created for each new training input data 1306.

However, in using certain training methods or for certain applications, the neural network 1206 may require many presentations of training sets to be adequately (acceptable metric) trained. In this case, two alternate approaches can be used to train the neural network 1206.

In the first approach, the neural network 1206 can save the training sets (that is, the training input data and the associated input data which is retrieved in step and module 308) in a database of training sets, which can then be repeatedly presented to the neural network 1206 to train the neural network. The user might be able to configure the number of training sets to be saved. As new training data becomes available, new training sets are constructed and saved. When the specified number of training sets has been accumulated (in a "stack"), the next training set created based on new lab data would "bump" the oldest training set out of the stack. This oldest training set would be

discarded. Conventional neural network training creates training sets all at once, off-line, and would keep using all the training sets created.

A second (or "stack") approach which can be used is to maintain a time history of input data and training input data in the historical database 1210, and to search the historical database 1210, locating training input data and constructing the corresponding training set by retrieving the associated input data.

It should be understood that the combination of the neural network 1206 and the historical database 1210 containing both the input data and the training input data with their associated timestamps provides a very powerful platform for building, training and using the neural network 1206. The present invention contemplates various other modes of using the data in the historical database 1210 and the neural network 1206 to prepare training sets for training the neural network 1206.

5. Error Acceptable ? Step and Module 310

Referring again to Figure 3, once the neural network 1206 has been trained in step and module 308, a step and module 310 of determining whether an acceptable error? occurs. Figure 11 shows a representative embodiment of the error acceptable? step and module 310.

Referring now to Figure 11, an order pointer 320 indicates that an compute global error using saved global error step and module 1102 occurs. The term global error as used herein means the error over all the outputs 2106 and/or over two or more training sets (cycles) of the neural network 1206. The global error reduces the effects of variation in the error from one training set (cycle) to the next. One

cause for the variation is the inherent variation in lab data tests used to generate the training input data.

Once the global error has been computed (estimated) in the step and module 1102, it is saved in a step and module 1104. The global error may be saved internally in the neural network 1206, or it may be stored in the historical database 1210. Storing the global error in the historical database 1210 provides an historical record of the overall performance of the neural network 1206.

Thereafter, if an appropriate history of global error is available (as would be the case in retraining), a step and module 1106 can be used to determine if the global error is statistically different from zero. This step and module 1106 determines whether a sequence of global error values falls within the expected range of variation around the expected (desired) value of zero, or whether the global error is statistically significantly different from zero. This step and module 1106 can be important when the training input data used to compute the global error has significant random variability. If the neural network 1206 is making accurate predictions, the random variability in the training input data (for example, caused by lab variation) will cause random variation of the global error around zero. This step and module 1106 reduces the tendency to incorrectly classify as not acceptable the predicted outputs of the neural network 1206.

If the global error is not statistically different from zero, then the global error is acceptable, and the present invention moves to an order pointer 122. An acceptable error indicated by order pointer 122 means that the neural network 1206 is trained. This completes step and module 104.

However, if the global error is statistically different from zero, the present invention in the retrain mode moves to

a step and module 1108, which is called training input data statistically valid?. (Note that step and module 1108 is not needed in the training mode of step and module 104. In the training mode, a global error statistically different from zero moves directly to an order pointer 322.)

If the training input data in the retraining mode is not statistically valid, this indicates that the acceptability of the global error cannot be determined, and the present invention moves to the order pointer 122. However, if the training input data is statistically valid, this indicates that the error is not acceptable, and the present invention moves back to the wait training input data interval step and module 304, as indicated in Figure 3.

The steps and modules described here for determining whether the global error is acceptable constitute one example of implementing a global error acceptable metric. It should be understood that different process characteristics, and different sampling frequencies, and different measurement techniques (for process conditions and product properties) may indicate alternate methods of determining whether the error is acceptable. The present invention contemplates any method of creating an error acceptable metric.

Thus, it has been seen that the present invention in step and module 104 configures and trains the neural network 1206 for use in the present invention.

C. Predict Output Data Using Neural Network Step and Module 106

Referring again to Figure 1, the order pointer 122 indicates that there are two parallel paths that the present invention uses after the configure and train neural network step and module 104. One of the paths, which the predict output data using neural network step and module 106

described below is part of, is used for predicting output data using the neural network 1206, for retraining the neural network 1206 using these predicted output data, and for disabling control of the controlled process when the (global) error from the neural network 1206 exceeds a specified error acceptable metric (criterion). The other path is the actual control of the process using the predicted output data from the neural network 1206.

Turning now to the predict output data using neural network step and module 106, this step and module 106 uses the neural network 1206 to produce output data for use in control of the process and for retraining the neural network 1206. Figure 4 shows a representative embodiment of the step and module 106.

Turning now to Figure 4, a wait specified prediction interval step or module 402 utilizes the method or procedure specifed by the user in steps or modules 3106 and 3108 for determining when to retrieve input data. Once the specified prediction interval has elapsed, the present invention moves to a retrieve input data at current time from historical database step or module 404. The input data is retrieved at the current time. That is, the most recent value available for each input data value is retrieved from the historical database 1210.

The neural network 1206 then predicts output data from the retrieved input data, as indicated by a step and module 406. This output data is used for process control, retraining and control purposes as discussed below in subsequent sections. Prediction is done using any presently known or future developed approach. For example, prediction can be done as specified above in Section I.B.

D. Retrain Neural Network Step or Module

Referring again to Figure 1, once the predicted output data has been produced by the neural network 1206, a retrain neural network step or module 108 is used.

Retraining of the neural network 1206 occurs when new training input data becomes available. Figure 5 shows a representative embodiment of the retrain neural network step or module 108.

Referring now to Figure 5, an order pointer 124 shows that a new training input data? step or module 306 determines if new training input data has become available. Figure 9 shows a representative embodiment of the new training input data? step or module 306. Step or module 306 was described above in connection with Figure 3; for this reason, it is not described again here.

As indicated by an order pointer 126, if new training data is not present, the present invention returns to the predict output data using neural network step or module 106, as shown in Figure 1.

If new training input data is present, the neural network 1206 is retrained, as indicated by a module or step 308. A representative example of module or step 308 is shown in Figure 10. Since training of the neural network is the same as retraining, and has been described in connection with Figure 3, module or step 308 is not discussed in detail here.

Once the neural network 1206 has been retrained, an order pointer 128 causes the present invention to move to an enable/disable control step or module 110 discussed below.

2066279

E. Enable/Disable Control Module or Step 110

Referring again to Figure 1, once the neural network 1206 has been retrained, as indicated by the step or module 108, the present invention moves to an enable/disable control step or module 110. The purpose of the enable/disable control step or module 110 is to prevent the control of the process using output data (predicted values) produced by the neural network 1206 when the error is not unacceptable ("poor").

A representative example of the enable/disable control step or module 110 is shown in Figure 6. Referring now to Figure 6, the function of module 110 is to enable control of the controlled process if the error is acceptable, and to disable control if the error is unacceptable. As shown in Figure 6, an order pointer 128 moves the present invention to an error acceptable? step or module 310. If the error between the training input data and the predicted output data is unacceptable, control of the controlled process is disabled by a disable control step and module 604. The disable control step and module 604 sets a flag (indicator) which can be examined by the control process using output data step and module 112 indicating that the output data should not be used for control.

Figure 30 shows a representative embodiment of the enable control step and module 602. Referring now to Figure 30, an order pointer 142 causes the present invention first to move to an output data indicates safety or operability problems? step or module 3002. If the output data does not indicate a safety or operability problem, this indicates that the process 1212 can continue to operate safely. This is indicated by the fact that the present invention moves to the enable control using output data step or module 3006.

In contrast, if the output data does indicate a safety or operability problem, the present invention recommends that the process being controlled be shut down, as indicated by a recommend process shutdown step and module 3004. This recommendation to the operator of the process 1212 can be made using any suitable approach. An example is a screen display or an alarm indicator. This safety feature allows the present invention to prevent the controlled process 1212 from reaching a critical situation.

If the output data does not indicate safety or operability problems in step and module 3002, or after the recommendation to shut down the process has been made in step and module 3004, the present invention moves to the enable control using output data step and module 3006. This step and module 3006 sets a flag (indicator) which can be examined by step and module 112, indicating that the output data should be used to control the process.

Thus, it can be appreciated that the enable/disable control step or module 110 provides the function to the present invention of (1) allowing control of the process 1212 using the output data in step or module 112, (2) preventing the use of the output data in controlling the process 1212, but allowing the process 1212 to continue to operate, or (3) shutting down the process 1212 for safety reasons.

F. Control Process Using Output Data Step or Module 112

Referring again to Figure 1, the order pointer 122 indicates that the control of the process using the output data from the neural network 1206 runs in parallel with the prediction of output data using the neural network 1206, the retraining of the neural network 1206, and the enable/disable control of the process 1212.

2066279

Figure 7 shows a representative embodiment of the control process using output data step and module 112. Referring now to Figure 7, the order pointer 122 indicates that the present invention first moves to a wait controller interval step or module 702. The interval at which the controller operates can be any preselected value. This interval can be a time value, an event, or the occurrence of a data value. Other interval control methods or procedures can be used.

Once the controller interval has occurred, as indicated by the order pointer, the present invention moves to a control enabled? step or module 704. If control has been disabled by the enable/disable control step or module 110, the present invention does not control the process 1212 using the output data. This is indicated by the order pointer marked "NO" from the control enabled ? step or module 704.

If control has been enabled, the present invention moves to the retrieve output data from historical database step or module 706. This step or module shows that the output data 1218 (see Figure 12) produced by the neural network 1206 and stored in the historical database 1210 is retrieved (1214) and used by the controller 1202 to compute controller output data 1208 for control of the process 1212.

This control by the controller 1202 of the process 1212 is indicated by an effectively control process using controller to compute controller output step or module 708 of Figure 7.

Thus, it can be appreciated that the present invention effectively controls the process using the output data from the neural network 1206. It should be understood that the control of the process 1212 can be any presently known or future developed approach, including the architecture shown in Figures 15 and 16.

Alternatively, when the output data from the neural network 1206 is determined to be unacceptable, the process 1212 can continue to be controlled 1202 by the controller without the use of the output data.

V. Preferred Structure (Architecture)

Discussed above in Section III has been the preferred method of operation of the present invention. Discussed in this Section are the preferred structures (architecture) of the present invention. However, it should be understood that in the description set forth above, the modular structure (architecture) of the present invention was also discussed in connection with the operation. Thus, certain portions of the structure of the present invention have inherently been described in connection with the description set forth above in Section III.

The preferred embodiment of the present invention comprises one or more software systems. In this context, software system is a collection of one or more executable software programs, and one or more storage areas, for example, RAM or disk. In general terms, a software system should be understood to comprise a fully functional software embodiment of a function, which can be added to an existing computer system to provide new function to that computer system.

Software systems generally are constructed in a layered fashion. In a layered system, a lowest level software system is usually the computer operating system which enables the hardware to execute software instructions. Additional layers of software systems may provide, for example, historical database capability. This historical database system provides a foundation layer on which additional software systems can be built. For example, a neural network software

system can be layered on top of the historical database. Also, a supervisory control software system can be layered on top of the historical database system.

A software system is thus understood to be a software implementation of a function which can be assembled in a layered fashion to produce a computer system providing new functionality. Also, in general, the interface provided by one software system to another software system is well-defined. It should be understood in the context of the present invention that delineations between software systems are representative of the preferred implementation. However, the present invention may be implemented using any combination or separation of software systems.

Figure 12 shows a preferred embodiment of the structure of the present invention. Referring now to Figure 12, the process 1212 being controlled receives the raw materials 1222 and produces the product 1216. Sensors 1226 (of any suitable type) provide sensor signals 1221, 1224, which are supplied to the historical database 1210 for storage with associated timestamps. It should be noted that any suitable type of sensor 1226 can be employed which provides sensor signals 1221, 1224.

The historical database 1210 stores the sensor signals 1224 that are supplied to it with associated timestamps as provided by a clock 1230. In addition, as described below, the historical database 1210 also stores output data 1218 from the neural network 1206. This output data 1218 also has associated timestamps provided by the neural network 1206.

Any suitable type of historical database 1210 can be employed. A historical database is generally discussed in Hale and Sellars, "Historical Data Recording for Process Computers," 77 Chem. Eng'g Progress 38 AICLE, New York, (1981).

2066279

The historical database 1210 that is used must be capable of storing the sensor input data 1224 with associated timestamps, and the predicted output data 1218 from the neural network 1206 with associated timestamps. Typically, the historical database 1210 will store the sensor data 1224 in a compressed fashion to reduce storage space requirements, and will store sampled (lab) data 1304 in uncompressed form.

Often, the historical database 1210 will be present in a chemical plant in the existing process control system. The present invention can utilize this historical database to achieve the improved process control obtained by the present invention.

A historical database is a special type of database in which at least some of the data is stored with associated time stamps. Usually the time stamps can be referenced in retrieving (obtaining) data from a historical database.

The historical database 1210 can be implemented as a stand alone software system which forms a foundation layer on which other software systems, such as the neural network 1206, can be layered. Such a foundation layer historical database system can support many functions in a process control environment. For example, the historical database can serve as a foundation for software which provides graphical displays of historical process data for use by a plant operator. An historical database can also provide data to data analysis and display software which can be used by engineers for analyzing the operation of the process 1212. Such a foundation layer historical database system will often contain a large number of sensor data inputs, possibly a large number of laboratory data inputs, and may also contain a fairly long time history for these inputs.

It should be understood, however, that the present invention requires a very limited subset of the functions of

2066279

the historical database 1210. Specifically, the present invention requires the ability to store at least one training data value with the timestamp which indicates an associated input data value, and the ability to store such an associated input data value. In certain circumstances where, for example, a historical database foundation layer system does not exist, it may be desirable to implement the essential historical database functions as part of the neural network software. By integrating the essential historical database capabilities into the neural network software, the present invention can be implemented in a single software system. It should be understood that the various divisions among software systems used to describe the present invention are only illustrative in describing the best mode as currently practiced. Any division or combination among various software systems of the steps and elements of the present invention may be used.

The historical database 1210, as used in the present invention, can be implemented using a number of methods. For example, the historical database can be built as a random access memory (RAM) database. The historical database 1210 can also be implemented as a disk-based database, or as a combination of RAM and disk databases. If an analog neural network 1206 is used in the present invention, the historical database 1210 could be implemented using a physical storage device. The present invention contemplates any computer or analog means of performing the functions of the historical database 1210.

The neural network 1206 retrieves input data 1220 with associated timestamps. The neural network 1206 uses this retrieved input data 1220 to predict output data 1218. The output data 1218 with associated timestamps is supplied to the historical database 1210 for storage.

A representative embodiment of the neural network 1206 is described above in Section I. It should be understood that neural networks, as used in the present invention, can be implemented in any way. For example, the preferred embodiment uses a software implementation of a neural network 1206. It should be understood, however, that any form of implementing a neural network 1206 can be used in the present invention, including physical analog forms. Specifically, as described below, the neural network may be implemented as a software module in a modular neural network control system.

It should also be understood with regard to the present invention that software and computer embodiments are only one possible way of implementing the various elements in the systems and methods. As mentioned above, the neural network 1206 may be implemented in analog or digital form and also, for example, the controller 1202 may also be implemented in analog or digital form. It should be understood, with respect to the method steps as described above for the functioning of the systems as described in this section, that operations such as computing (which imply the operation of a digital computer) may also be carried out in analog equivalents or by other methods.

Returning again to Figure 12, the output data 1214 with associated timestamps stored in the historical database 1210 is supplied by a path 1214 to the controller 1202. This output data 1214 is used by the controller 1202 to generate controller output data 1208 sent to an actuator(s) 1228 used to control a controllable process state 2002 of the process 1212. Representative examples of controller 1202 are discussed below.

The shaded box shown in Figure 12 indicates that the neural network 1206 and the historical database 1210 may, in a variant of the present invention, be implemented as a

single software system. This single software system could be delivered to a computer installation in which no historical database previously existed, to provide the functions of the present invention. Alternately, a neural network configuration function (or program) 1204 could also be included in this software system.

Two additional aspects of the architecture and structure shown in Figure 12 are as follows. First, it should be noted that the controller 1202 may also be provided with input data 1221 from sensors 1220. This input data is provided directly to controller 1202 from these sensor(s).

Second, the neural network configuration module 1204 is connected in a bi-directional path configuration with the neural network 1206. The neural network configuration module 1204 is used by the user (developer) to configure and control the neural network 1206 in a fashion as discussed above in connection with the step and module 104 (Figure 1), or in connection with the user interface discussion contained below.

Turning now to Figure 13, an alternate preferred embodiment of the structure and architecture of the present invention is shown. Only differences between the embodiment of Figure 12 and that of Figure 13 are discussed here. These differences are as follows.

A laboratory ("lab") 1307 is supplied with samples 1302. These samples 1302 could be physical specimens or some type of data from an analytical test or reading. Regardless of the form, the lab takes this material/data and utilizes it to produce actual measurements 1304, which are supplied to the historical database 1210 with associated timestamps. The values 1304 are stored in the historical database 1210 with their associated timestamps.

lThus, the historical database 1210 also now contains actual test results or actual lab results in addition to sensor input data. It should be understood that a laboratory is illustrative of a source of actual measurements 1304 which are useful as training input data. Other sources are encompassed by the present invention. Laboratory data can be electronic data, printed data, or data exchanged over any communications link.

The second difference in this embodiment is that the neural network 1206 is supplied with the lab data 1304 and associated timestamps stored in the historical database 1210.

Another addition to the architecture of Figure 12 is error data 1504 (Figure 15) supplied by the neural network 1206 with associated timestamps to the historical database 1210 for storage.

Thus, it can be appreciated that the embodiment of Figure 13 allows the present invention to utilize lab data 1304 as training input data 1306 to train the neural network.

Turning now to Figure 14, a representative embodiment of the controller 1202 is shown. The embodiment utilizes a regulatory controller 1406 for regulatory control of the process 1212. Any type of regulatory controller is contemplated which provides such regulatory control. There are many commercially available embodiments for such a regulatory controller. Typically, the present invention would be implemented using regulatory controllers already in place. In other words, the present invention can be integrated into existing process control systems.

In addition to the regulatory controller 1406, the embodiment shown in Figure 14 also includes a supervisory controller 1408. The supervisory controller 1408 computes supervisory controller output data, computed in accordancd with the predicted output data 1214. In other words, the

2066279

supervisory controller 1408 utilizes the predicted output data 1214 from the neural network 1206 to produce supervisory controller output data 1402.

The supervisory controller output data 1402 is supplied to the regulatory controller 1406 for changing the regulatory controller setpoint 1404 (or other parameter of regulatory controller 1406). In other words, the supervisory controller output data 1402 is used for changing the regulatory controller setpoint 1404 so as to change the regulatory control provided by the regulatory controller 1406.

Any suitable type of supervisory controller 1408 can be employed by the present invention, including commercially available embodiments. The only limitation is that the supervisory controller 1408 be able to use the output data 1408 to compute the supervisory controller output data 1402 used for changing the regulatory controller setpoint (parameter) 1404.

The present invention contemplates the supervisory controller 1408 being in a software and hardware system which is physically separate from the regulatory controller 1406. For example, in many chemical processes, the regulatory controller 1406 is implemented as a digital distributed control system (DCS). These digital distributed control systems provide a very high level of robustness and reliability for regulating the process 1212. The supervisory controller 1408, in contrast, may be implemented on a host-based computer, such as a VAX (VAX is a trademark of DIGITAL EQUIPMENT CORPORATION, Maynard, Massachusetts).

Referring now to Figure 15, a more detailed embodiment of the present invention is shown. In this embodiment, the supervisory controller 1408 is separated from the regulatory controller 1406. The three shaded boxes shown in Figure 15 suggest various ways in which the functions of the

supervisory controller 1408, the neural network configuration program 1204, the neural network 1206 and the historical database 1210 can be implemented. For example, the box labeled 1502 shows how the supervisory controller 1408 and the neural network 1206 can be implemented together in a single software system. This software system may take the form of a modular system as described below in Figure 16. Alternately, the neural network configuration program 1204 may be included as part of the software system. These various software system groupings are indicative of various ways in which the present invention can be implemented. However, it should be understood that any combination of functions into various software systems can be used to implement the present invention.

Referring now to Figure 16, a representative embodiment 1602 of the neural network 1206 combined with the supervisory controller 1408 is shown. This embodiment is called a modular supervisory controller approach. The modular architecture that is shown illustrates that the present invention contemplates the use of various types of modules which can be implemented by the user (developer) in configuring neural network(s) 1206 in combination with supervisory control functions so as to achieve superior process control operation.

Several modules that can be implemented by the user of the present invention are shown in the embodiment of Figure 16. Specifically, in addition to the neural network module 1206, the modular embodiment of Figure 16 also includes a feedback control module 1602, a feedforward control module 1604, an expert system module 1606, a cusum (cumulative summation) module 1608, a Shewhart module 1610, a user program module 1612, and a batch event module 1614. Each of these can be selected by the user. The user can implement

more than one of each of these in configuring the present invention. Moreover, additional types of modules can be utilized.

The intent of the embodiment shown in Figure 16 is to illustrate three concepts. First, the present invention can utilize a modular approach which will ease user configuration of application of the present invention. Second, the modular approach allows for much more complicated systems to be configured since the modules act as basic building blocks which can be manipulated and used independently of each other.

Third, the modular approach shows that the present invention can be integrated into other process control systems. In other words, the present invention can be implemented into the system and method of the United States patents and patent applications previously referenced.

Specifically, this modular approach allows the neural network capability of the present invention to be integrated with the expert system capability described in the above-noted patents and patent applications. As described above, this enables the neural network capabilities of the present invention to be easily integrated with other standard control functions such as statistical tests and feedback and feedforward control. However, even greater function can be achieved by combining the neural network capabilities of the present invention, as implemented in this modular embodiment, with the expert system capabilities of the above-noted patent applications, also implemented in the modular embodiment. This easy combination and use of standard control functions, neural network functions, and expert system functions allows a very high level of capability to be achieved in solving process control problems.

2066279

The modular approach to building neutral networks gives two principal benefits. First, the specification needed from the user is greatly simplified so that only data is required to specify the configuration and function of the neutral network. Secondly, the modular approach allows for much easier integration of neutral network function with other related control functions, such as feedback control, feedforward control, etc.

In contrast to a programming approach to building a neutral network, a modular approach provides a partial definition beforehand of the function to be provided by the neutral network module. The predefined function for the module determines the procedures that need to be followed to carry out the module function, and it determines any procedures that need to be followed to verify the proper configuration of the module. The particular function will define the data requirements to complete the specification of the neutral network module. The specifications for a modular neural network would be comprised of configuration information which defines the size, connectivity and behavior of the neutral network in general, and the data interactions of the neutral network which define the source and location of data that will be used and created by the network.

Two approaches can be used to simplify the user configuration of neutral networks. First, a limited set of procedures can be prepared and implemented in the modular neutral network software. These predefined functions will by nature define the specifications needed to make these procedures work as a neutral network module. For example, the creation of a neutral network module which is fully connected, has one hidden or middle layer, and has no feedback would require the specification of the number of inputs, the number of middle error elements, and number of

outputs. It would not require the specification for the connections between the inputs, the ouputs and elements. Thus, the user input required to specify such a module is greatly simplified. This predefined procedure approach is the preferred method of implementing the modular neural network.

A second approach could be used to provide modular neutral network function is to allow a limited set of natural language expressions to be used to define the neutral network. In such an implementation, the user or developer would be permitted to enter, through typing or other means, natural language definitions for the neutral network. For example, the user may enter the text which might read, for example, "I want a fully connected feedforward neutral network." These user inputs can be parsed searching for specification combinations of terms, or their equivalents, which would allow the specific configuration information to be extracted from the restricted natural language input.

By parsing the total user input provided in this method, the complete specification for a neutral network module could be obtained. Once this information is known, two approaches could be used to generate a runnable module.

The first approach would be to search for a predefined procedure matching the configuration information provided by the restricted natural language input. This would be useful where users tend to specify the same basic neutral network functions for many problems.

A second approach could provide for much more flexible creation of neutral network function. In this approach, the specifications obtained by parsing the natural language input could be used to generate a neutral network procedure by actually generating runnable or compilable code. In this approach, the neutral network functions would be defined in

relatively small increments as opposed to the approach of providing a complete predefined neutral network function. This approach may combine, for example, a small function which is able to obtain input data and populate a set of inputs. By combining a number of such small functional pieces and generating code which reflects and incorporates the user specifications, a complete neutral network procedure could be generated.

This approach could optionally include the ability to query the user for specifications which have been neglected or omitted in the restricted natural language input. Thus, for example, if the user neglected to specify the number of outputs in the network, the user could be prompted for this information and the system could generate an additional line of user specification reflecting the answer to the query.

The parsing and code generation in this approach use pre-defined, small sub-functions of the overall neural network function. A given key word (term) corresponds to a certain sub-function of the overall neural network function. Each sub-function has a corresponding set of key words (terms) and associated key words and numeric values. Taken together, each key word and associated key words and values constitute a symbolic specification of the neural network sub-function. The collection of all the symbolic specifications make up a symbolic specification of the entire neural network function.

The parsing step processes the substantially natural language input. It removes the unnecessary natural language words, and groups the remaining key words and numeric values into symbolic specifications of neural network subfunctions. One way to implement parsing is to break the input into sentences and clauses bounded by periods and commas, and restrict the specification to a single subfunction per

clause. Each clause is searched for key words, numeric values, and associated key words. The remaining words are discarded. A given key word (term) corresponds to a certain sub-function of the overall neural network function.

Or, key words can have relational tag words, like "in," "with," etc., which can indicate the relation of one key word to another. For example, in the specification "3 nodes in the hidden layer," the word "in" relates "3 nodes" to "hidden layer," so that a hidden layer sub-function specification is indicated. Using such relational tag words, multiple sub-function specification could be processed in the same clause.

Key words can be defined to have equivalents. For example, the user might be allowed, in an embodiment of this aspect of the invention, to specify the transfer function (activation function) used in the elements (nodes) in the network. Thus the key word might be "activation function" and an equivalent might be "transfer function." This keyword corresponds to a set of pre-defined subfunctions which implement various kinds of transfer functions in the neural network elements. The specific data that might be allowed in combination with this term might be, for example, the term "sigmoidal" or the word "threshold." These specific data, combined with the key word, indicate which of the sub-functions should be used to provide the activation function capability in the neural network when it is constructed.

Another example might be key word "nodes," which might have an equivalent "nodes" or "elements." The associated data would be an integer number which indicates the number of nodes in a given layer. In this particular case, it might be advantageous to look for the numeric data in combination with the word or term "in" and the key word "hidden layer," etc. In combination, these might specify the number of nodes in the middle layer. Thus, it can be seen that various levels

of flexibility in the substantially natural language specification can be provided. Increasing levels of flexibility require more detailed and extensive specification of key words and associated data with their associated key words.

In contrast, the key word "fully connected" might have no associated input. By itself, it conveys the entire meaning.

The neural network itself is constructed, using this method, by processing the specifications, as parsed from the hsubstantially natural language input, probably in a pre-defined order, and generating the fully functional procedure code for the neural network from the procedural sub-function code fragments.

The other major advantage of a modular approach is the ease of integration with other functions in the application (problem) domain. For example, in the process control domain, it may be desirable or productive to combine the functions of a neutral network with other more standard control functions such as statistical tests, feedback control, etc. The implementation of neutral networks as modular neutral networks in a larger control system can greatly simplify this kind of implementation.

The incorporation of modular neutral networks into a modular control system is beneficial because it makes it easy to create and use neutral network predictions in a control application. However, the application of modular neutral networks in a control system is different from the control functions that are typically found in a control system. For example, the control functions described in some of the United States patents and patent applications referenced above generally rely on the current information for their actions, and they do not generally define their

function in terms of past data. In order to make a neutral network function effectively in a modular control system, some means is needed to train and operate the neutral network using the data which is not generally available by retrieving current data values. The systems and methods of the present invention, as described above, provide this essential capability which allow a modular neutral network function to be implemented in a modular control system.

A modular neutral network has several characteristics which significantly ease its integration with other control functions. First, the execution of neutral network functions, prediction and/or training are easily coordinated in time with other control functions. The timing and sequencing capabilities of a modular implementation of a neutral network provide this capability. Also, when implemented as a modular function, neutral networks can make their results readily accessible to other control functions that may need them. This can be done, for example, without needing to store the neutral network outputs in an external system such as a historical database.

Modular neutral networks can run either synchronized or unsynchronized with other functions in the control system. Any number of neutral networks can be created within the same control application, or in different control applications, within the control system. This may significantly facilitate the use of neutral networks to make predictions of output data where several small neutral networks may be more easily or rapidly trained than a single large neutral network. Modular neutral networks also provide a consistent specification and user interface so that a user trained to use the modular neutral network control system can address many control problems without learning new software.

2066279

An extension of the modular concept is the specification of data using pointers. Here again, the user (developer) is offered the easy specification of a number of data retrieval or data storage functions by simply selecting the function desired and specifying the data needed to implement the function. For example, the retrieval of a time-weighted average from the historical database is one such predefined function. By selecting a data type such a time-weighted average, the user (developer) need only specify the specific measurement desired, and the starting and ending time boundaries and the predefined retrieval function will use the appropriate code or function to retrieve the data. This significantly simplifies the user's access to data which may reside in a number of different process data systems. By contrast, without the modular approach, the user would have to be skilled in the programming techniques needed to write the calls to retrieve the data from the state assistance.

A further development of the modular approach of the present invention is shown in Figure 17. Figure 17 shows the neural network 1206 in a modular form.

Referring now to Figure 17, a specific software embodiment of the modular form of the present invention is shown. In this modular embodiment, a limited set of neural network module types 1702 is provided. Each neural network module type 1702 allows the user to create and configure a neural network module implementing a specific type of neural network. Different types of neural networks may have different connectivity, different numbers of layers of elements, different training methods and so forth. For each neural network module type, the user may create and configure neural network modules. Three specific instances of neural network modules are shown as 1702', 1702'', and 1702'''.

2065279

In this modular software embodiment, neural network modules are implemented as data storage areas which contain a procedure pointer 1710', 1710'', 1710''' to procedures which carry out the functions of the neural network type used for that module. The neural network procedures 1706' and 1706'' are contained in a limited set of neural network procedures 1704. The procedures 1706', 1706'' correspond one to one with the neural network types contained in the limited set of neural network types 1702.

In this modular software embodiment, many neural network modules may be created which use the same neural network procedure. In this case, the multiple modules each contain a procedure pointer to the same neural network procedure 1706' or 1706''. In this way, many modular neural networks can be implemented without duplicating the procedure or code needed to execute or carry out the neural network functions.

Referring now to Figure 18, a more specific software embodiment of the modular neural network is shown. This embodiment is of particular value when the neural network modules are implemented in the same modular software system as modules performing other functions such as statistical tests or feedback control.

Because neural networks can use a large number of inputs and outputs with associated error values and training input data values, and also because neural networks can require a large number of weight values which need to be stored, neural network modules may have significantly greater storage requirements than other module types in the control system. In this case, it is advantageous to store neural network parameters in a separate neural network parameter storage area 1806. This structure means that modules implementing functions other than neural network functions need not reserve unused storage sufficient for neural networks.

In this modular software embodiment, each instance of a modular neural network 1702' and 1702" contains two pointers. The first pointers 1710' and 1710'' are the procedure pointer described above in reference to Figure 17. Each neural network module also contains a second pointer, parameter pointers 1802' anad 1802'' which point to a storage area 1806', 1806'' for network parameters in a neural network parameter storage area 1804. Only neural network modules need contain the parameter pointers 1802', 1802'' to the neural network parameter storage area 1804. Other module types such as control modules which do not require such extensive storage need not have the storage allocated via the parameter pointer 1802.

Figure 22 shows representative aspects of the architecture of the neural network 1206. The representation in Figure 22 is particularly relevant in connection with the modular neural network approach shown in Figures 16, 17 and 18 discussed above.

Referring now to Figure 22, the components to make and use a representative embodiment of the neural network 1206 are shown in an exploded format.

The neural network 1206 must contain a neural network model. As stated above, the present invention contemplates all presently available and future developed neural network models and architectures. As shown in Figure 22, the neural network model 2202 can have a fully connected 2220 aspect, or a no feedback 2222 aspect. These are just examples. Other aspects or architectures for the neural network model 2202 are contemplated.

The neural network 1206 must have access to input data and training input data and access to locations in which it can store output data and error data. The preferred embodiment of the present invention uses an on-line approach.

In this approach, the data is not kept in the neural network 1206. Instead, data pointers are kept in the neural network which point to data storage locations in a separate software system. These data pointers, also called data specifications, can take a number of forms and can be used to point to data used for a number of purposes.

For example, input data pointer 2204 and output data pointer 2206 must be specified. As shown in the exploded view, the pointer can point to or use a particular data source system 2224 for the data, a data type 2226, and a data item pointer 2228.

Neural network 1206 must also have a data retrieval function 2208 and a data storage function 2210. Examples of these functions are callable routines 2230, disk access 2232, and network access 2234. These are merely examples of the aspects of retrieval and storage functions.

Neural network 1206 must also have prediction timing and training timing. These are specified by prediction timing control 2212 and training timing control 2214. One way to implement this is to use a timing method 2236 and its associated timing parameters 2238. Referring now to Figure 24, examples of timing method 2236 include a fixed time interval 2402, new data entry 2404, after another module 2406, on program request 2408, on expert system request 2410, when all training data updates 2412, and batch sequence methods 2414. These are designed to allow the training and function of the neural network 1206 to be controlled by time, data, completion of modules, or other methods or procedures. The examples are merely illustrative in this regard.

Figure 24 also shows examples of the timing parameters 2238. Such examples include the time interval 2416, the module specification 2420, and the sequence specification 2422. Another example is the data item specification

(pointer) 2418. As is shown in Figure 24, examples of the data items specification include specifying the data source system 2224, the data type 2226, and the data item pointer 2228 which have been described above.

Referring again to Figure 22, training data coordination, as discussed previously, may also be required in many applications. Examples of approaches that can be used for such coordination are shown. One is to use all current values as representative by reference numeral 2240. Another is to use current training input data values and the input data at the earliest training input data time, as indicated by reference numeral 2242. Another approach is to use the current training input data values with the input data from the latest train time, as indicated by reference numeral 2244. Again, these are merely examples, and should not be construed as limiting in terms of the type of coordination of training data that can be utilized by the present invention.

The neural network 1206 also needs to be trained, as discussed above. As stated previously, any presently available or future developed training method is contemplated by the present invention. The training method also may be somewhat dictated by the architecture of the neural network model that is used. Examples of aspects of training methods include back propagation 2246, generalized delta 2248, and gradient descent 2250, all of which are well known in the art. In this regard, reference is made to the article series entitled "Neural Networks Primer," by Maureen Caudill, AI Expert, December 1987 (Part I), February 1988 (Part II), June 1988 (Part III), August 1988 (Part IV), and November 1988 (Part V).

Referring now to Figure 23, examples of the data source system 2224, the data type 2226, and the data item pointer 2228 are shown for purposes of illustration.

With respect to data source system 2224, examples are an historical database 1210, a distributed control system 1202, a programmable controller 2302, and a networked single loop controller 2304. These are merely illustrative.

Any data source system can be utilized by the present invention. It should also be understood that such source system could either be a storage device or an actual measuring or calculating device. All that is required is that a source of data be specified to provide the neural network 1206 with the input data 1220 that is needed to produce the output data 1218. The present invention contemplates more than one data source system used by the same neural network 1206.

The neural network 1206 needs to know the data type that is being specified. This is particularly important in an historical database 1210 since it can provide more than one type of data. Several examples are shown in Figure 23 as follows: current value 2306, historical value 2308, time weighted average 2310, controller setpoint 2312, and controller adjustment amount 2314. Other types are contemplated.

Finally, the data item pointer 2228 must be specified. The examples shown are a loop number 2316, a variable number 2318, a measurement number 2320, and a loop tag I.D. 2322. Again, these are merely examples for illustration purposes, since the present invention contemplates any type of data item pointer 2228.

It is thus seen that neural network 1206 can be constructed so as to obtain desired input data 1220 or to provide output data 1218 in any intended fashion. In the preferred embodiment of the present invention, this is all done through menu selection by the user (developer) using a software based system on a computer platform.

2066279

The construction of the controller 1202 is shown in Figure 25 in an exploded format. Again, this is merely for purposes of illustration. First, the controller 1202 must be implemented on some hardware platform 2502. Examples of hardware platforms 2502 include pneumatic single loop controller 2414, electronic single loop controller 2516, networked single looped controller 2518, programmable loop controller 2520, distributed control system 2522, and programmable logic controller 2524. Again, these are merely examples for illustration. Any type of hardware platform 2502 is contemplated by the present invention.

In addition to the hardware platform 2502, the controller 1202, 1406, 1408 needs to implement or utilize an algorithm 2504. Any type of algorithm 2504 can be used. Examples shown include: proportional (P) 2526; proportional, integral (PI) 2528; proportional, integral, derivative (PID) 2530; internal model 2532; adaptive 2534; and, non-linear 2536. These are merely illustrative of feedback algorithms. However, the present invention also contemplates feedforward or other algorithm approaches.

The controller 1202 also inherently includes parameters 2506. These parameters are utilized by the algorithm 2504. Examples shown include setpoint 1404, proportional gain 2538, integral gain 2540, derivative gain 2542, output high limit 2544, output low limit 2546, setpoint high limit 2548, and setpoint low limit 2550.

The controller 1202 also needs some means for timing its operation. One way to do this is to use a timing means 2508. Timing means 2508, for example, can use a timing method 2236 with associated timing parameters 2238, as previously described. Again, these are merely illustrative.

The controller 1202 also needs to utilize one or more input signals 2510, and to provide one or more output signals

2066279

2512. These can take the form of pressure signals 2552, voltage signals 2554, amperage (current) signals 2556, or digital values 2558. In other words, input and output signals can be in either analog or digital format.

VI. User Interface

The present invention utilizes a template and menu driven user interface 2600, 2700 which allows the user to configure, reconfigure and operate the present invention. This approach makes the present invention very user friendly. It also eliminates the need for the user to perform any computer programming, since the configuration, reconfiguration and operation of the present invention is carried out in a template and menu format not requiring any actual computer programming expertise or knowledge.

The system and method of the present invention utilizes templates. These templates define certain specified fields that must be addressed by the user in order to configure, reconfigure and operate the present invention. The templates tend to guide the user in using the present invention.

Representative examples of templates for the menu driven system of the present invention are shown in Figures 26-29. These are merely for purposes of illustration.

The preferred embodiment of the present invention uses a two-template specification 2600, 2700 for a neural network module. Referring now to Figure 26, the first template 2600 in this set of two templates is shown. This template 2600 specifies general characteristics of how the neural network 1206 will operate. The portion of the screen within a box labeled 2620, for example, shows how timing options are specified for the neural network module 1206. As previously described, more than one timing option may be provided. This template 2600 provides a training timing option under the

label "train" and a prediction timing control specification under the "label run." The timing methods shown in boldface type are chosen from a pop-up menu of various timing methods that are implemented in the preferred embodiment. The parameters needed for the timing method which is chosen are entered in the shaded blocks under heading "Time Interval and Key Block." These parameters are specified only for timing methods for which they are required. Not all timing methods require parameters, and not all timing methods that require parameters require all the parameters.

In a box labeled 2606 bearing the heading "Mode and Store Predicted Outputs," the prediction and training functions of the neural network module can be controlled. By putting a check in the box next to either the train or the run designation under "Mode," the training and/or prediction functions of the neural network module 1206 are enabled. By putting a check in the box next to the "when training" and "when running" labels, the storage of predicted output data 1218 can be enabled when the neural network 1206 is training and when the neural network 1206 is predicting (running), respectively.

The size of the neural network 1206 is specified in a box labeled 2622 bearing the heading "network size." In this embodiment of a neural network module 1206, there are three layers only, and the user may specify how many elements or nodes are to be used in each layer. In the preferred embodiment, the number of inputs, outputs and middle nodes is limited to some predefined value.

The coordination of input data with training data is controlled using a checkbox labeled 2608. By checking this box, the user can specify that input data 1220 is to be retrieved such that the timestamps on the input data 1220 correspond with the timestamps on the training input data

1306. The training or learning constant can be entered in a field 2610. This constant determines how aggressively the weights in the neural network 1206 are adjusted when there is an error 1504 between the output data 1218 and the training input data 1306.

The user, by pressing a keypad softkey labeled "dataspec page" 2624, may call up the second template 2700 in the neural network module specification. This template 2700 is shown in Figure 27. This template 2700 allows the user to specify (1) the data inputs 1220, 1306, and (2) the outputs 1218, 1504 that are to be used by the neural network module. A data specification box 2702, 2704, 2706, and 2708 is provided for each of the network inputs 1220, network training inputs 1306, the network outputs 1218, and the summed error output, respectively. These correspond to the input data, the training input data, the output data, and the error data. These four boxes use the same data specification methods.

Within each data specification box, the data pointers and parameters are specified. In the preferred embodiment, the data specification comprises a three-part data pointer as described above. In addition, various time boundaries and constraint limits can be specified depending on the data type specified.

In Figure 28, an example of a pop-up menu is shown. In this figure, the specification for the data system for the network input number 1 is being specified as shown by the highlighted field reading "DMT PACE." The box in the center of the screen is a pop-up menu 2802 of choices which may be selected to complete the data system specification. The templates in the preferred embodiment of the present invention utilize such pop-up menus 2802 whereever applicable.

2066279

Figure 29 shows the various elements which make up the data specification block. These include a data title 2902, an indication as to whether the block is scrollable 2906, and an indication of the number of the specification in a scrollable region 2904. The box also contains arrow pointers indicating that additional data specifications exist in the list either above or below the displayed specification. These pointers 2922 and 2932 are displayed as a small arrow when other data is present. Otherwise, they are blank.

The items making up the actual data specification are: a data system 2224, a data type 2226, a data item pointer or number 2228, a name and units label for the data specification 2908, a label 2924, a time boundary 2926 for the oldest time interval boundary, a label 2928, a time specification 2930 for the newest time interval boundary, a label 2910, a high limit 2912 for the data value, a label 2914, a low limit value 2916 for the low limit on the data value, a label 2918, and a value 2920 for the maximum allowed change in the data value.

The data specification shown in Figure 29 is representative of the preferred mode of implementing the present invention. However, it should be understood that various other modifications of the data specification could be used to give more or less flexibility depending on the complexity needed to address the various data sources which may be present. The present invention contemplates any variation on this data specification method.

Although the foregoing refers to particular preferred embodiments, it will be understood that the present invention is not so limited. It will occur to those of ordinarily skill in the art that various modifications may be made to the

2066279

disclosed embodiments, and that such modifications are intended to be within the scope of the present invention.

5

CLAIMS:

1. A computer neural network process control system adapted for predicting output data provided to a controller used to control a process for producing a product having at least one product property, the computer neural network process control system comprising:

(a) an historical database comprising,

(1) storing means for storing a first training input data with an associated first timestamp, and for storing a first input data indicated by said associated first timestamp, and

(2) retrieving means, responsive to said storing means, for retrieving said first training input data and said first input data indicated by said associated first timestamp; and

(b) a trained neural network, trained in accordance with said first training input data and said first input data,comprising,

(1) a first data pointer and a second data pointer, and

(2) predicting means, using said trained neural network, for predicting the output data which is stored in accordance with said first data pointer, the output data predicted in accordance with second input data specified by said second data pointer, wherein the output data is used to control the process.

2. The computer neural network process control system of claim 1, further comprising sensor means, responsive to said historical database, for providing said first input data.

3. The computer neural network process control system of claim 2, further comprising clock means for providing an associated second timestamp to said historical database.

4. The computer neural network process control system of claim 1, wherein said second input data of said predicting means includes data indicative of past output of data.

5. The computer neural network process control system of claim 1, wherein said neural network comprises a software system adapted to run on a computer platform.

6. The computer neural network process control system of claim 1, further comprising output means for storing the output data in said historical database.

7. The computer neural network process control system of claim 1, wherein said first data pointer and said second data pointer are three-part pointers with additional qualifiers.

8. A computer neural network process control system adapted for predicting output data provided to a controller used to control a process for producing a product having at least one product property, the computer neural network process control system comprising:

(a) a database comprising,

(1) storing means for storing, in independently addressable storage, at least one data value,

(2) retrieving means, responsive to said storing means, for independently retrieving each of said at least one data value(s); and

(b) a neural network comprising,

(1) a data pointer specifying one of said at least one data value(s) in said database to be used as a first input data, and

(2) predicting means for predicting the output data, connected to request, from said retrieving means, said data value indicated by said data pointer, and connected to receive said first input data from said retrieving means.

9. The computer neural network process control system of claim 8, wherein said data pointer comprises means for storing a specification of a data location.

10. The computer neural network process control system of claim 8, wherein said data pointer comprises means for storing a specification of a data type.

11. The computer neural network process control system of claim 8, wherein said data pointer comprises means for storing a specification of a data item number.

12. The computer neural network process control system of claim 8, wherein said data pointer comprises means for storing a specification of an oldest time interval boundary.

13. The computer neural network process control system of claim 8, wherein said data pointer comprises means for storing a specification of a newest time interval boundary.

14. The computer neural network process control system of claim 8, wherein said data pointer comprises means for storing a specification of one specific point in time.

15. The computer neural network process control system of claim 8, wherein said data pointer comprises means for storing a specification(s) of a high limit value and/or a low limit value.

16. The computer neural network process control system of claim 8, further comprising:

(c) a user interface for presenting a template to a user for entering said data pointer.

17. The computer neural network process control system of claim 16, wherein said template comprises user-alterable fields for entering a data system specification, a data type specification, and/or a data item specification, and said data pointer comprises means for storing said data system specification, said data type specification, and/or said data item specification.

18. The computer neural network process control system of claim 15, wherein said predicting means further comprises means for scaling a value in accordance with said high limit value and/or said low limit value.

19. The computer neural network process control system of claim 15, wherein said predicting means further comprises means for clamping an output data in accordance with said high limit value and/or said low limit value.

20. The computer neural network process control system of claim 8, further comprising:

(c) a controller, connected to use the output data as its controller input data, and connected to control the process in accordance with a controller parameter.

21. The computer neural network process control system of claim 20, wherein said data pointer comprises means for storing a specification(s) of a high limit value and/or a low limit value.

22. The computer neural network process control system of claim 21, further comprising comparing means for disabling the operation of said controller when the value of an input, training input, or output is above said high limit value and/or below said low limit value.

23. The computer neural network process control system of claim 8, wherein said network further comprises:

(3) a second data pointer specifying a storage location in said database to be used to store a first output data.

24. The computer neural network process control system of claim 8, wherein said neural network further comprises:

(3) training means for training said neural network in accordance with training input data; and

(4) a second data pointer specifying one of said at least one data value(s) in said database to be used as a first training input data.

2066279

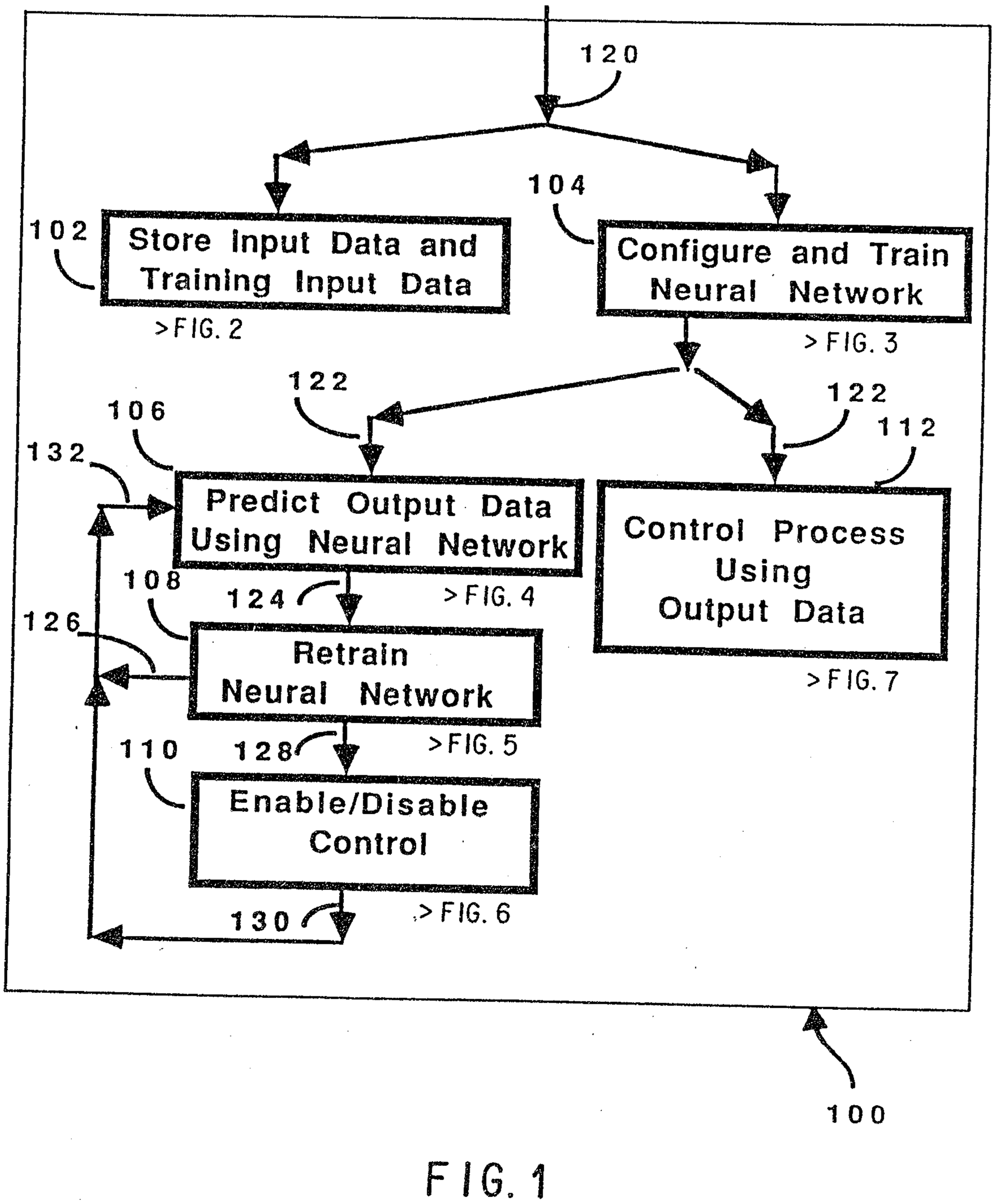


FIG. 1

2066279

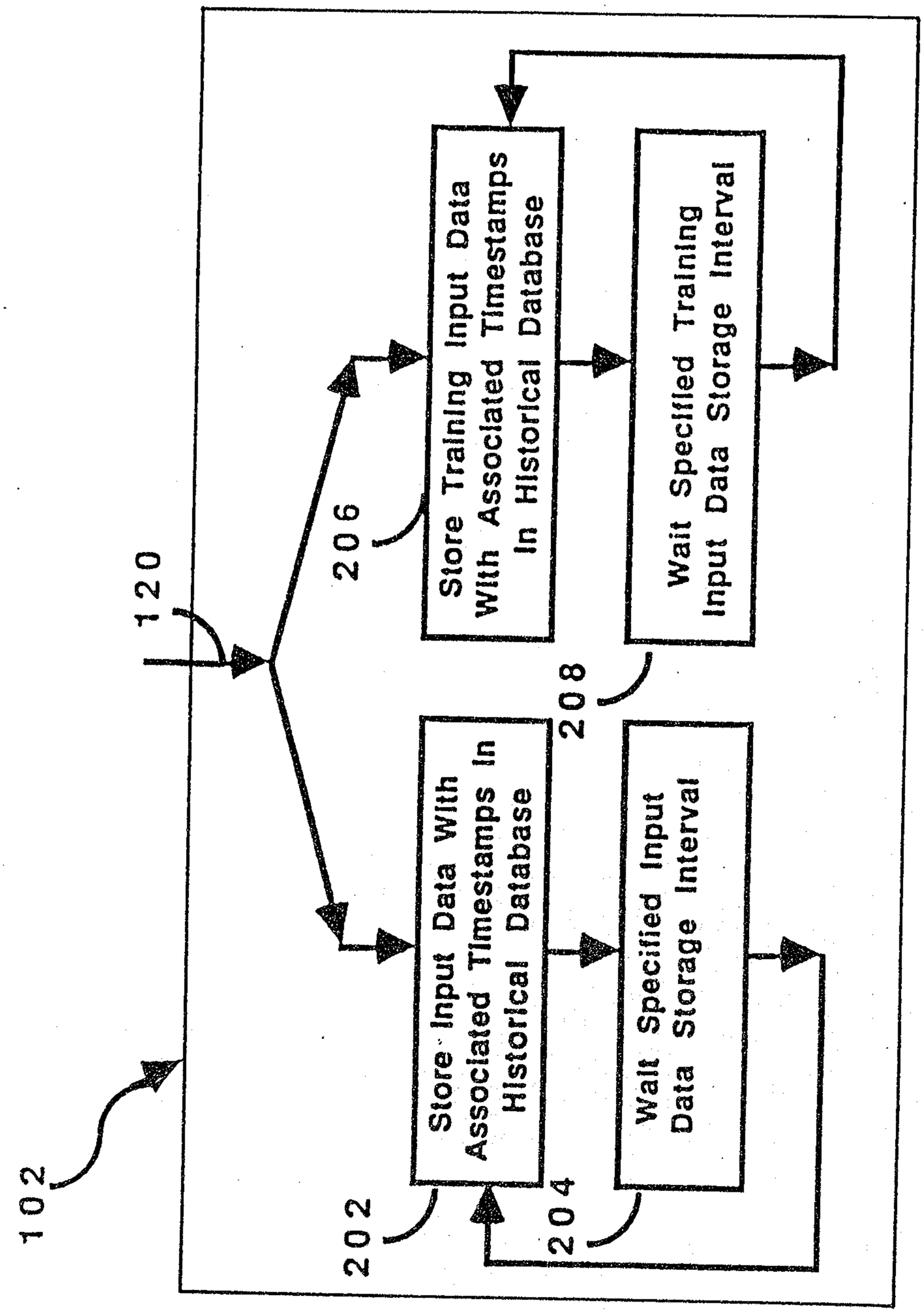


FIG. 2

SUBSTITUTE SHEET

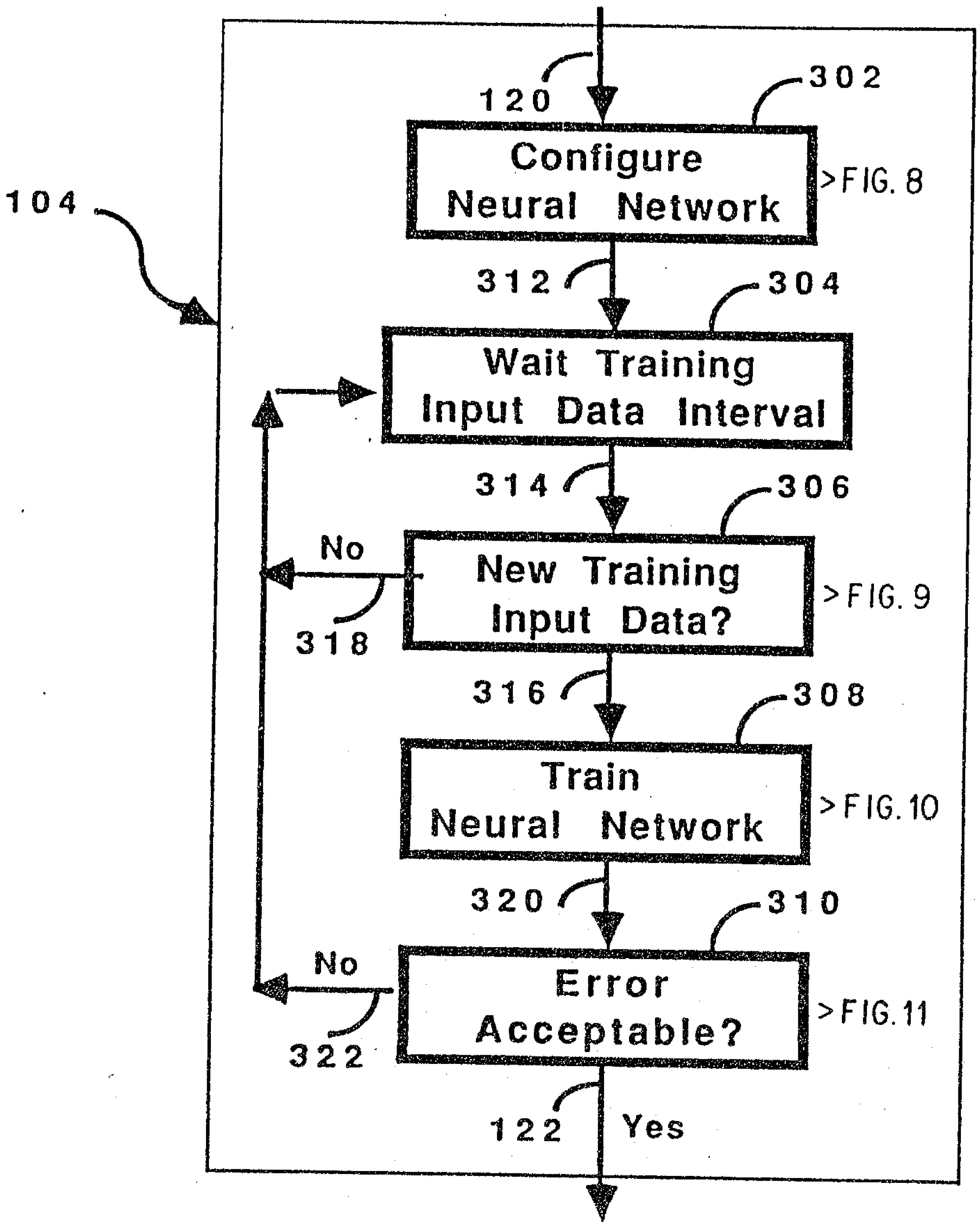

104
120
302
Configure Neural Network
>FIG. 8
312
304
Wait Training Input Data Interval
314
306
No
New Training Input Data?
>FIG. 9
318
316
308
Train Neural Network
>FIG. 10
320
310
No
Error Acceptable?
>FIG. 11
322
122
Yes

FIG. 3

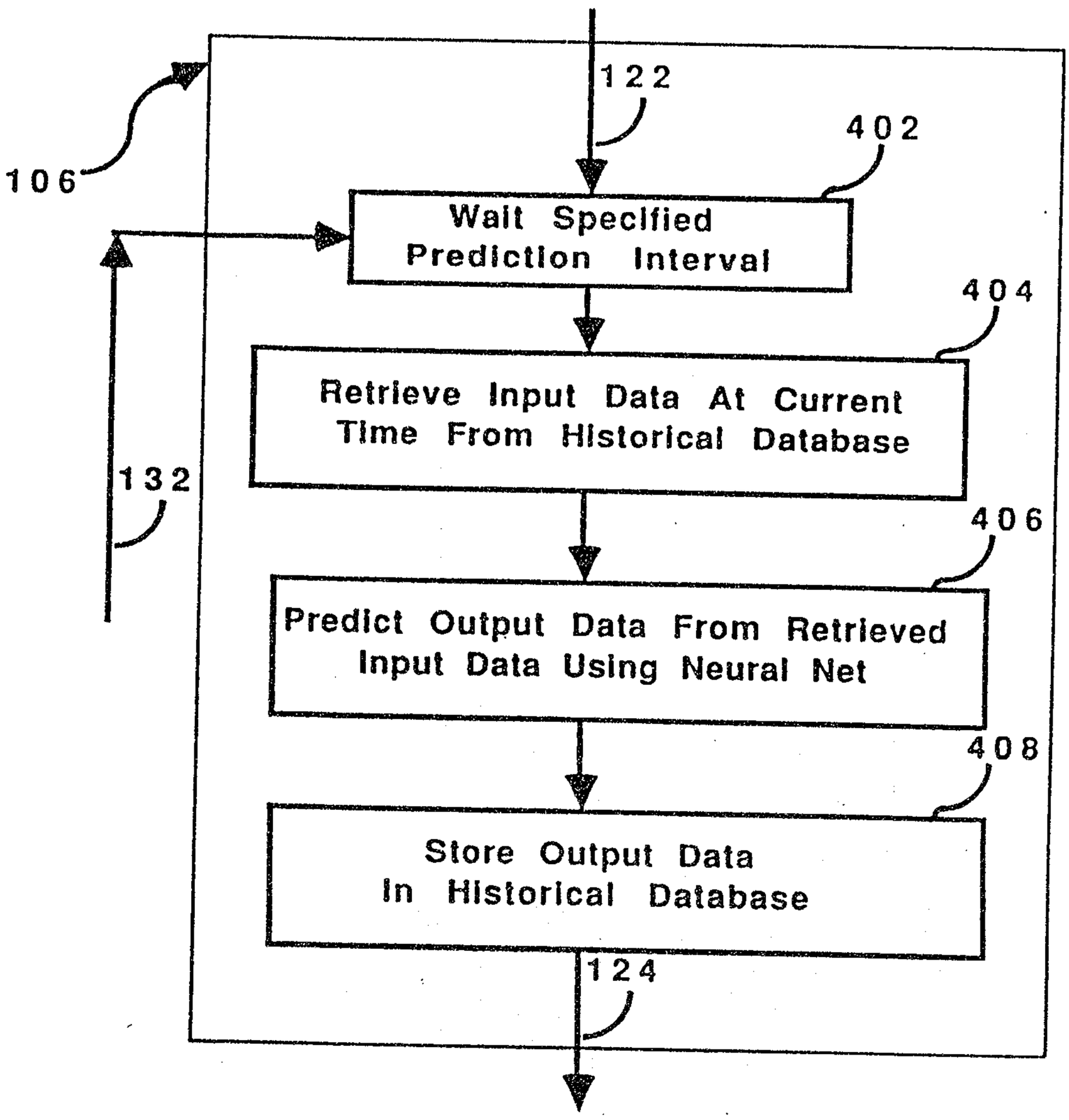

122
402
106
Wait Specified Prediction Interval
404
Retrieve Input Data At Current Time From Historical Database
132
406
Predict Output Data From Retrieved Input Data Using Neural Net
408
Store Output Data In Historical Database
124

FIG. 4

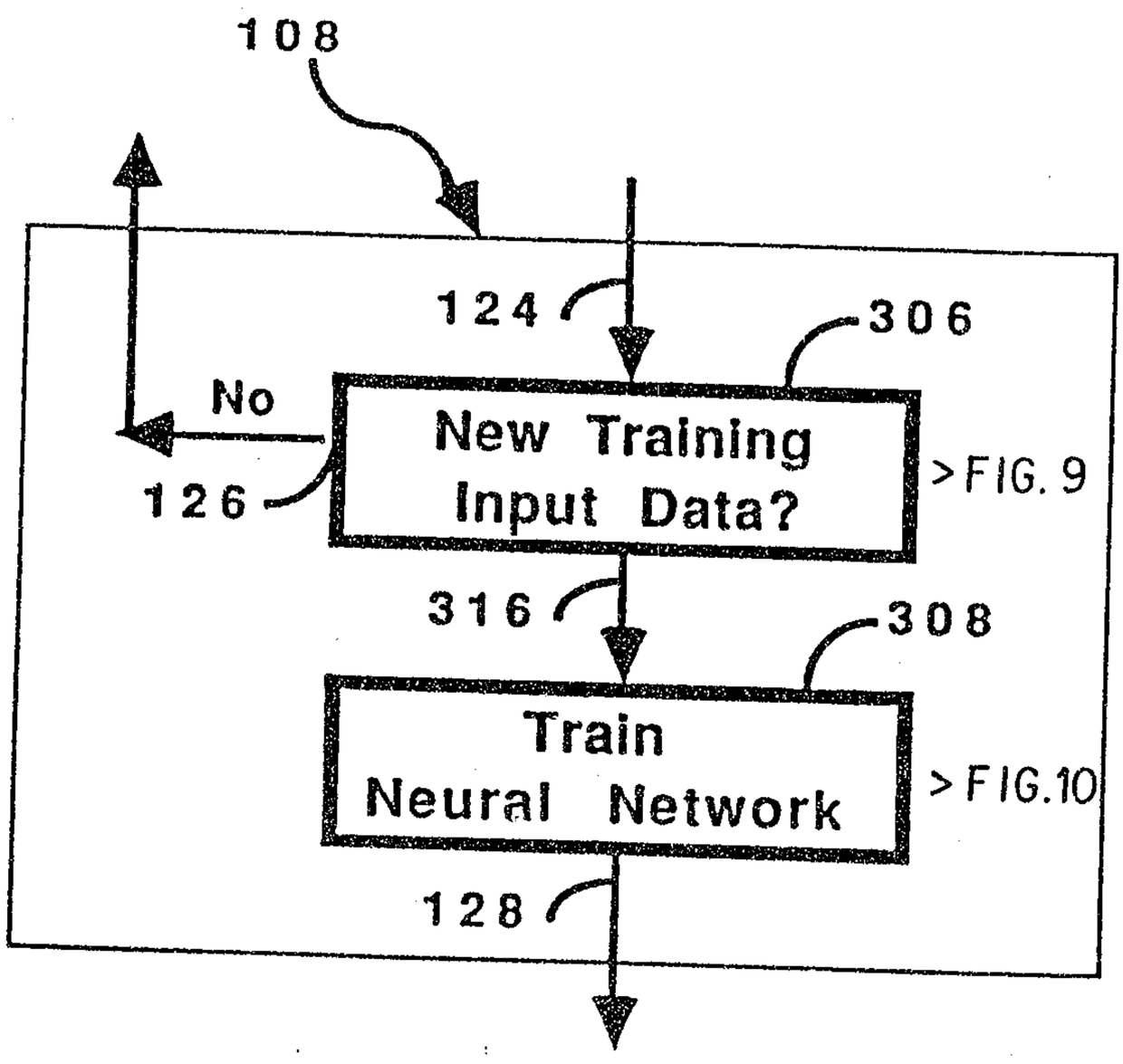

108
124
306
No
New Training Input Data?
126
> FIG. 9
316
308
Train Neural Network
> FIG.10
128

FIG. 5

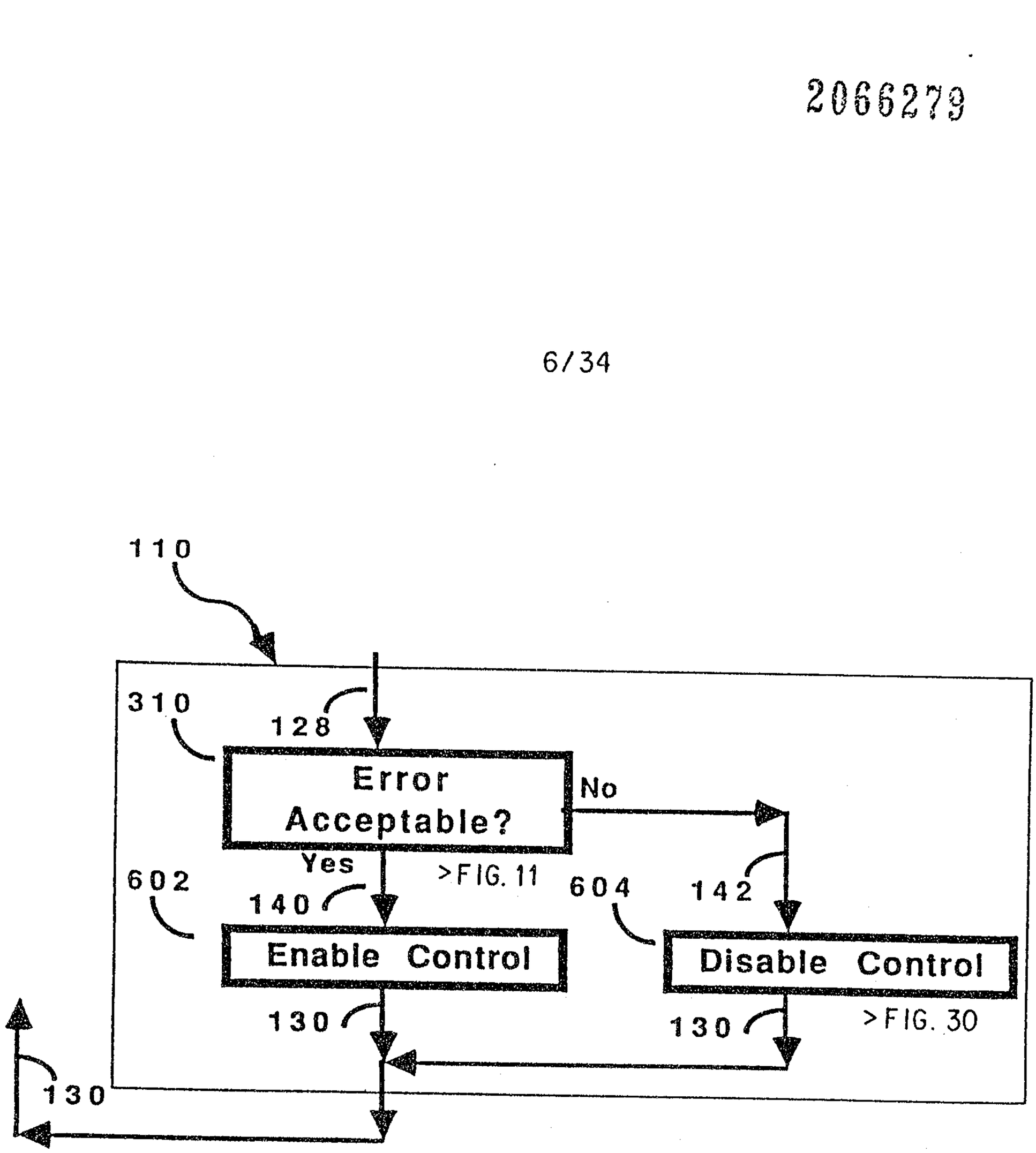

110
310
128
Error Acceptable?
No
Yes
>FIG. 11
602
140
Enable Control
604
142
Disable Control
130
130
>FIG. 30
130

FIG. 6

2066279

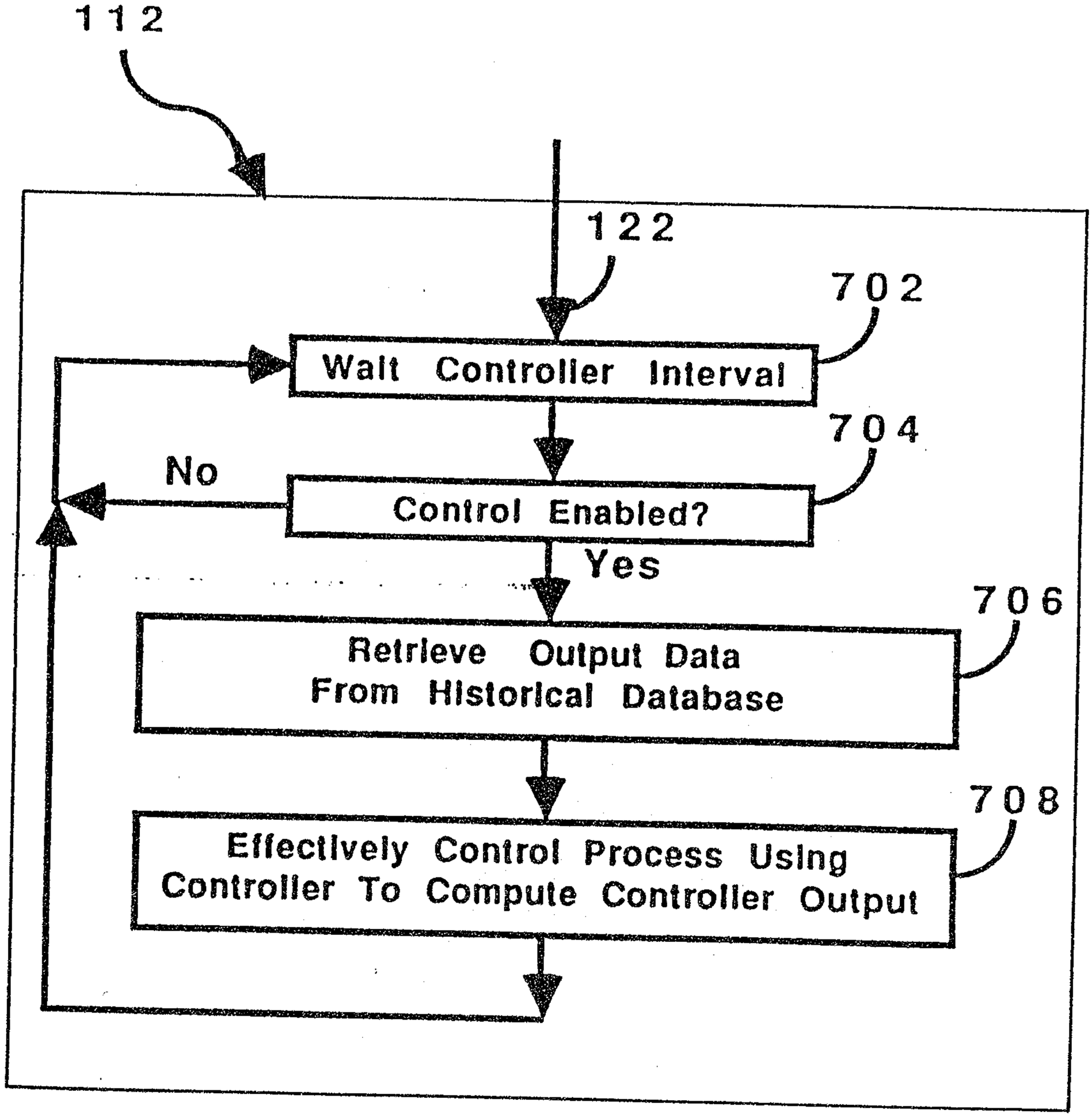


FIG. 7

SUBSTITUTE SHEET

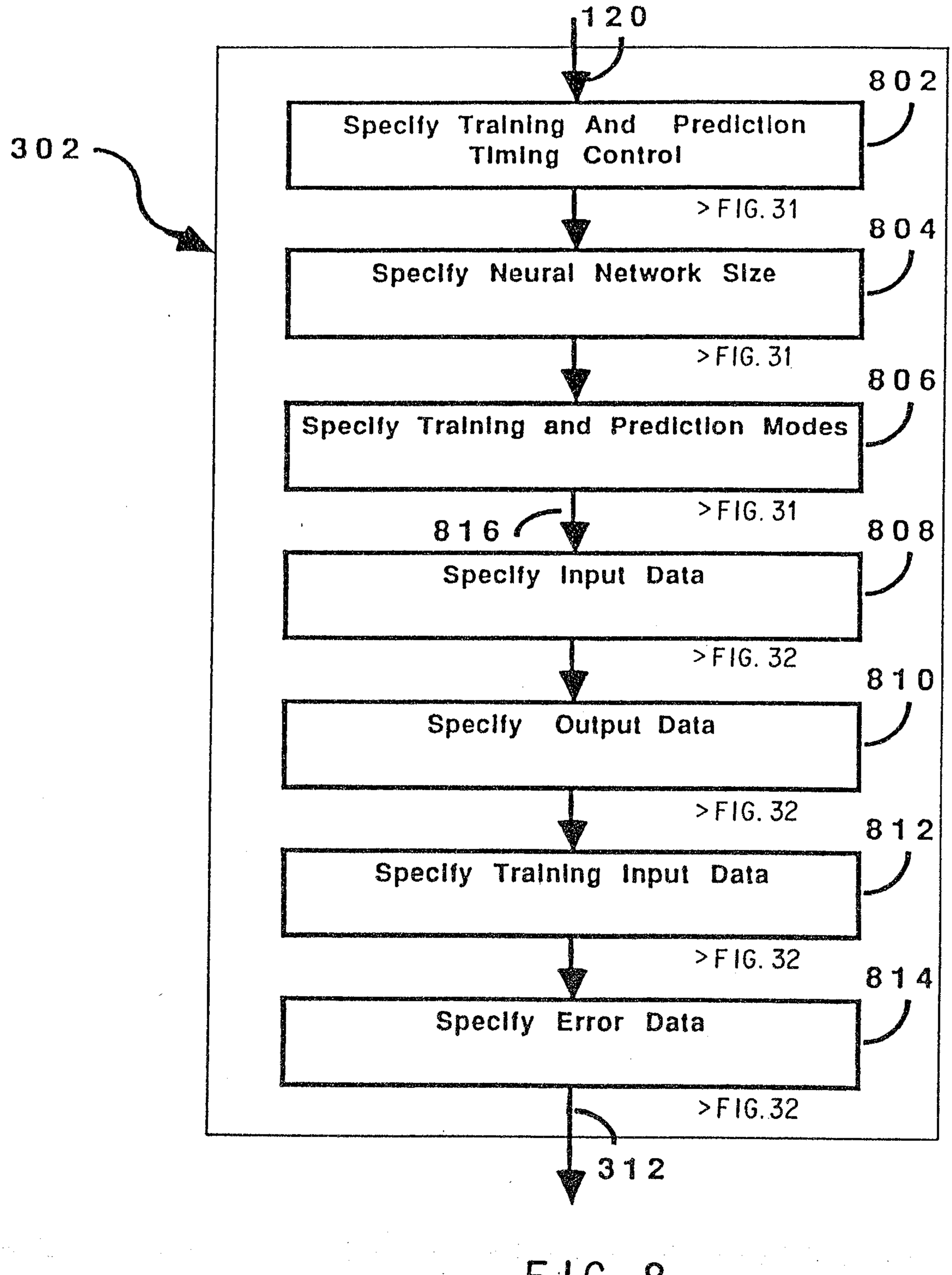

120
302
802
Specify Training And Prediction Timing Control
> FIG. 31
804
Specify Neural Network Size
> FIG. 31
806
Specify Training and Prediction Modes
> FIG. 31
816
808
Specify Input Data
> FIG. 32
810
Specify Output Data
> FIG. 32
812
Specify Training Input Data
> FIG. 32
814
Specify Error Data
> FIG. 32
312

FIG. 8

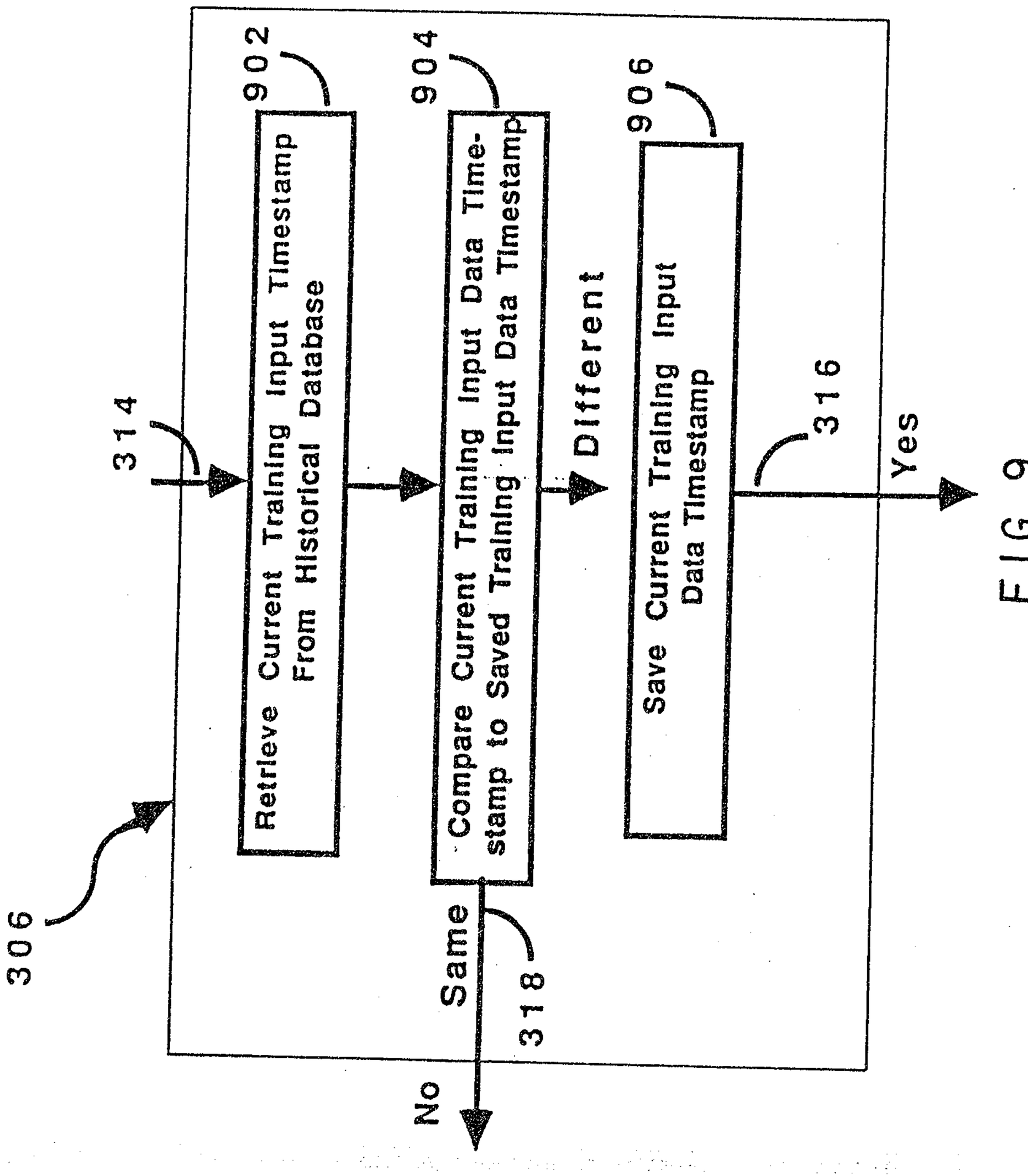

306
314
Retrieve Current Training Input Timestamp From Historical Database
902
Compare Current Training Input Data Time-stamp to Saved Training Input Data Timestamp
904
Same
Different
906
Save Current Training Input Data Timestamp
316
Yes
318
No

FIG. 9

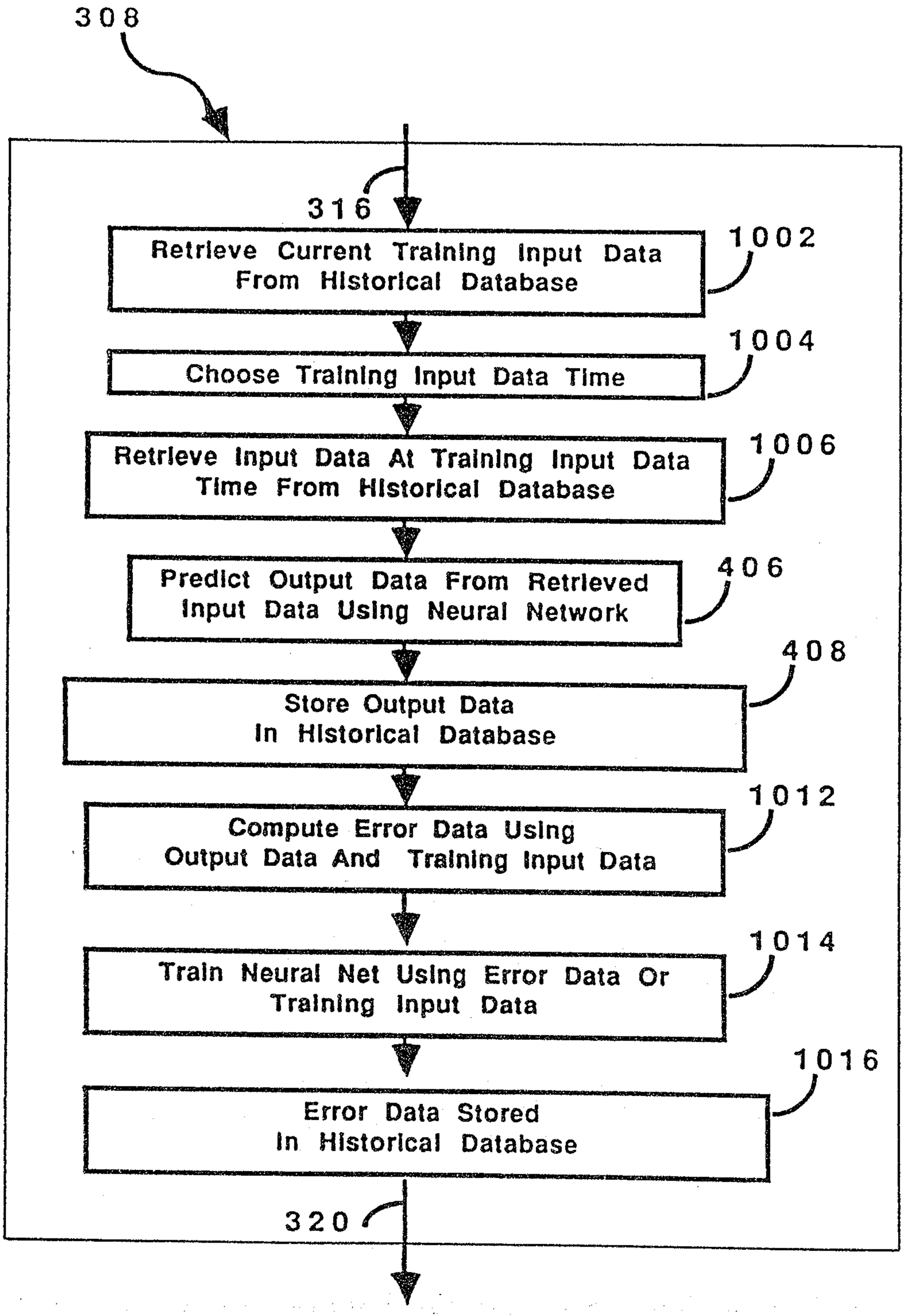

308
316
Retrieve Current Training Input Data From Historical Database
1002
Choose Training Input Data Time
1004
Retrieve Input Data At Training Input Data Time From Historical Database
1006
Predict Output Data From Retrieved Input Data Using Neural Network
406
Store Output Data In Historical Database
408
Compute Error Data Using Output Data And Training Input Data
1012
Train Neural Net Using Error Data Or Training Input Data
1014
Error Data Stored In Historical Database
1016
320

FIG. 10

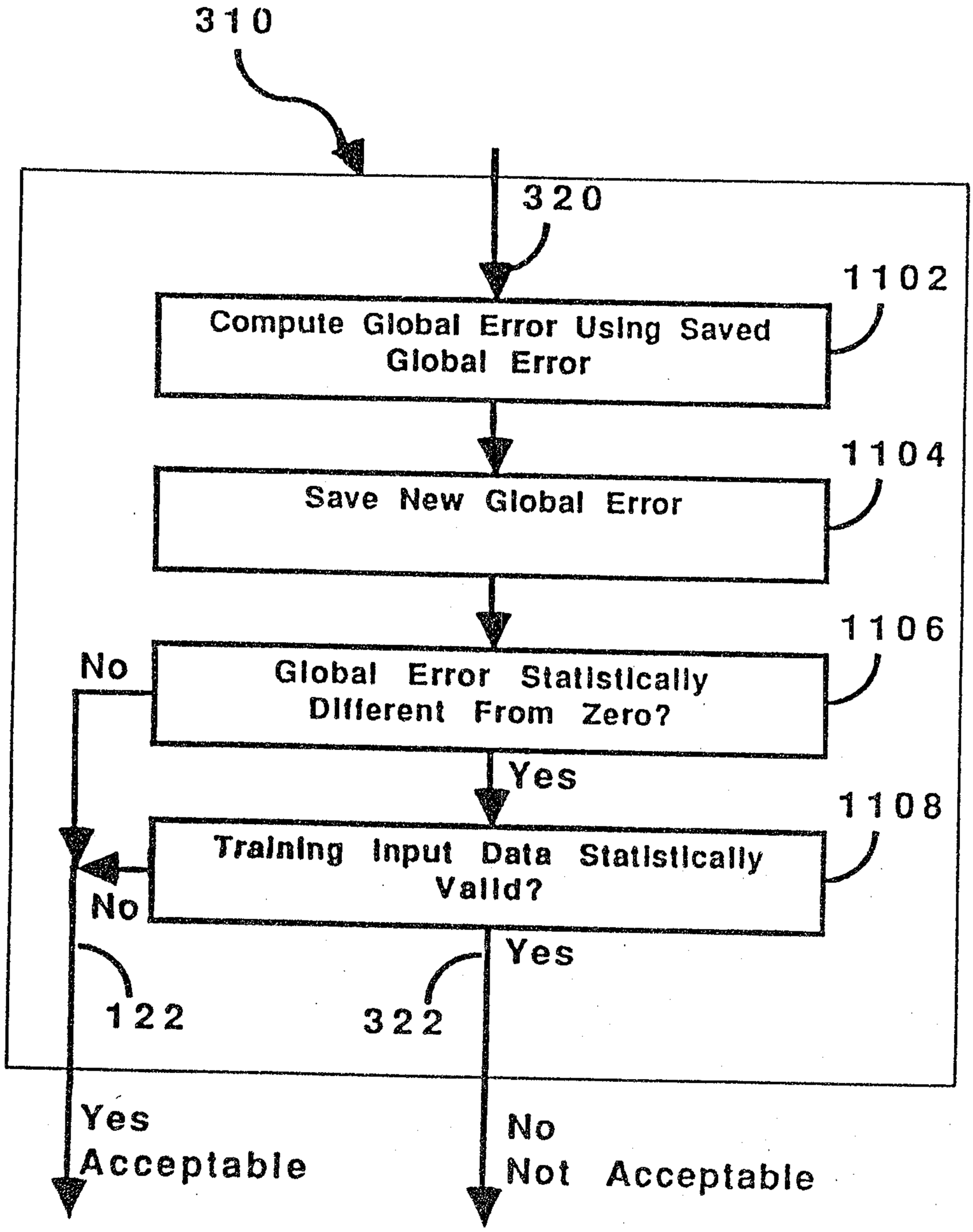

310
320
1102
Compute Global Error Using Saved Global Error
1104
Save New Global Error
1106
No
Global Error Statistically Different From Zero?
Yes
1108
Training Input Data Statistically Valid?
No
Yes
122
322
Yes
Acceptable
No
Not Acceptable

FIG. 11

2066279

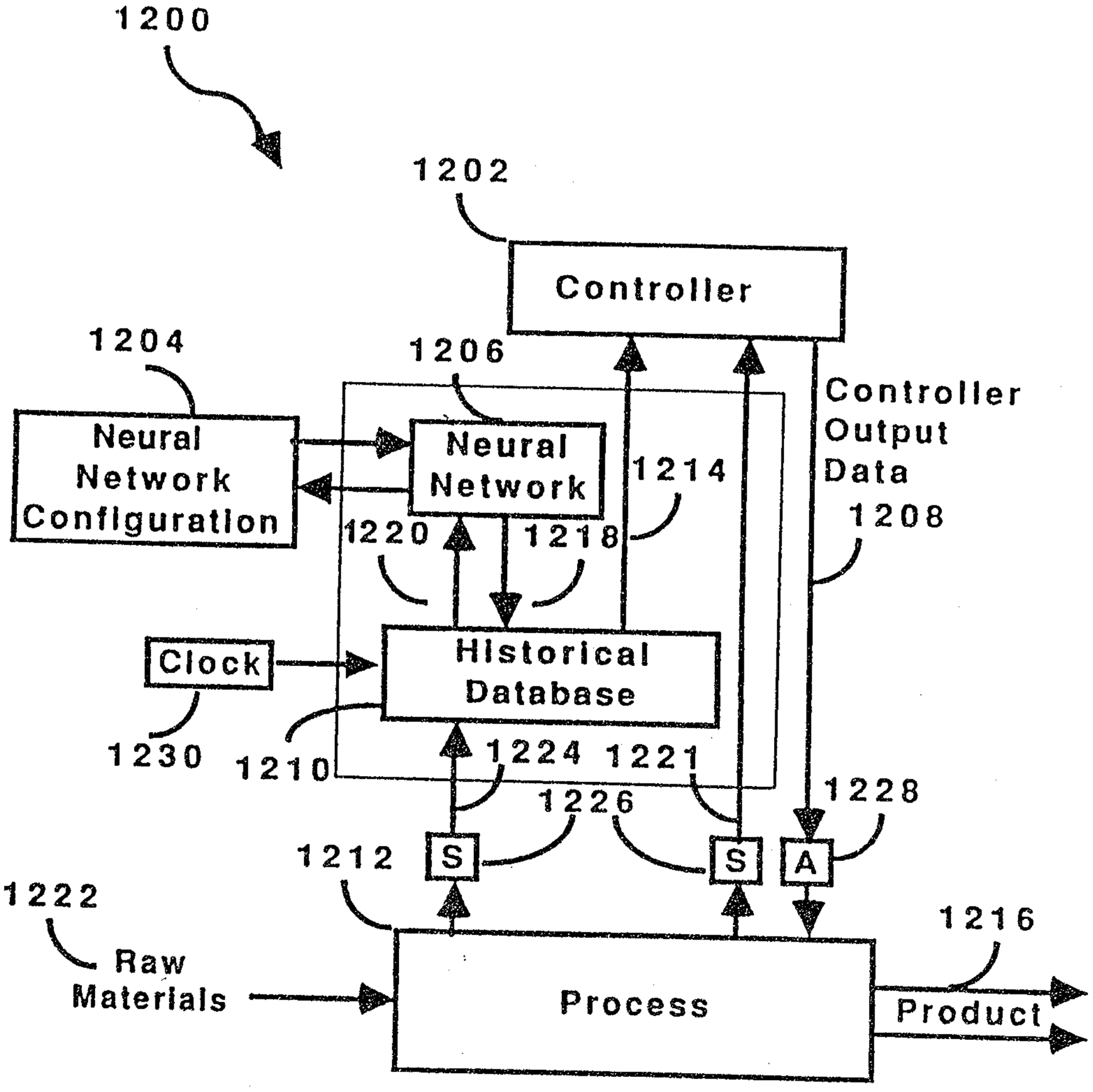


FIG. 12

2066279

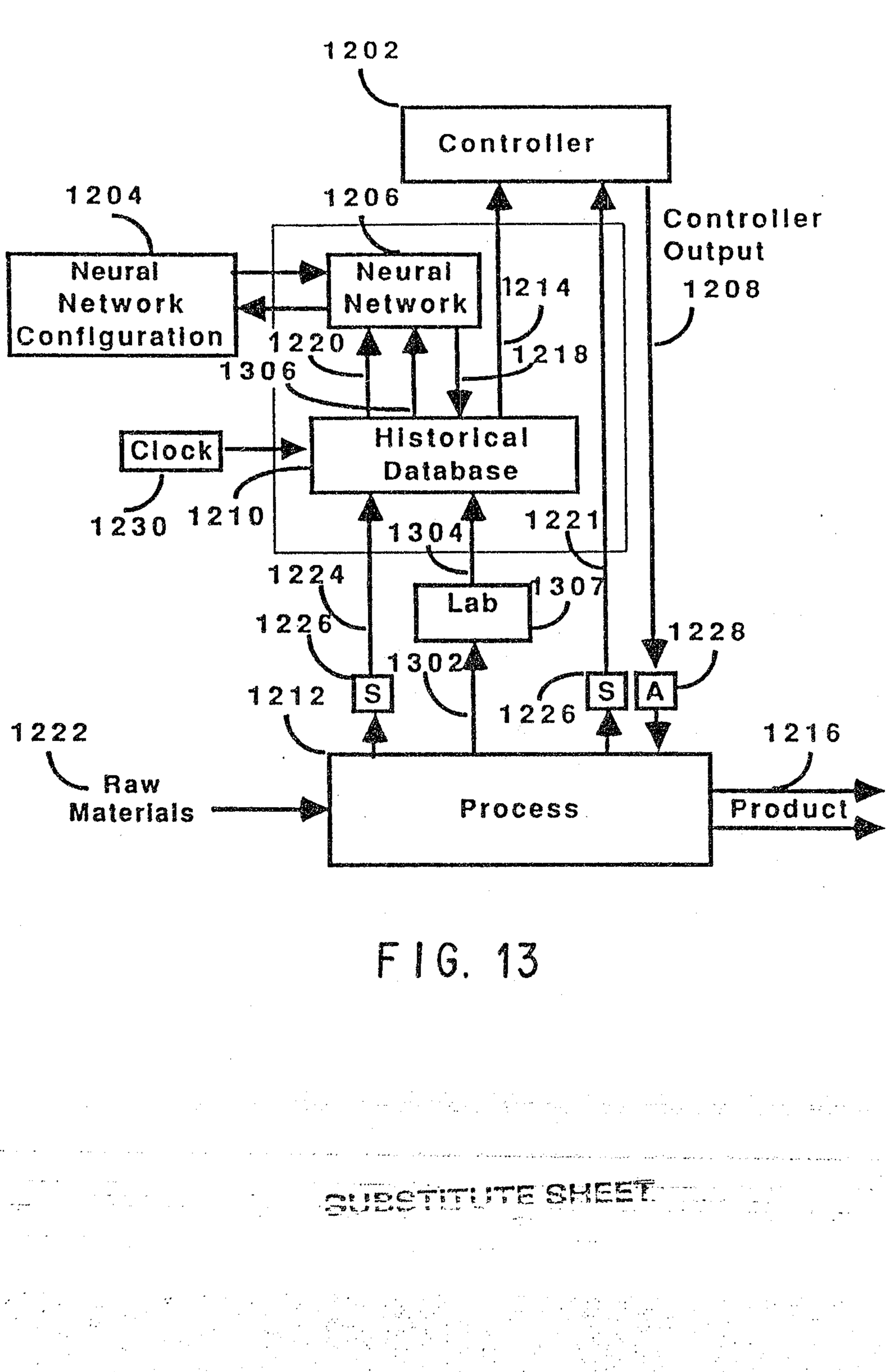


FIG. 13

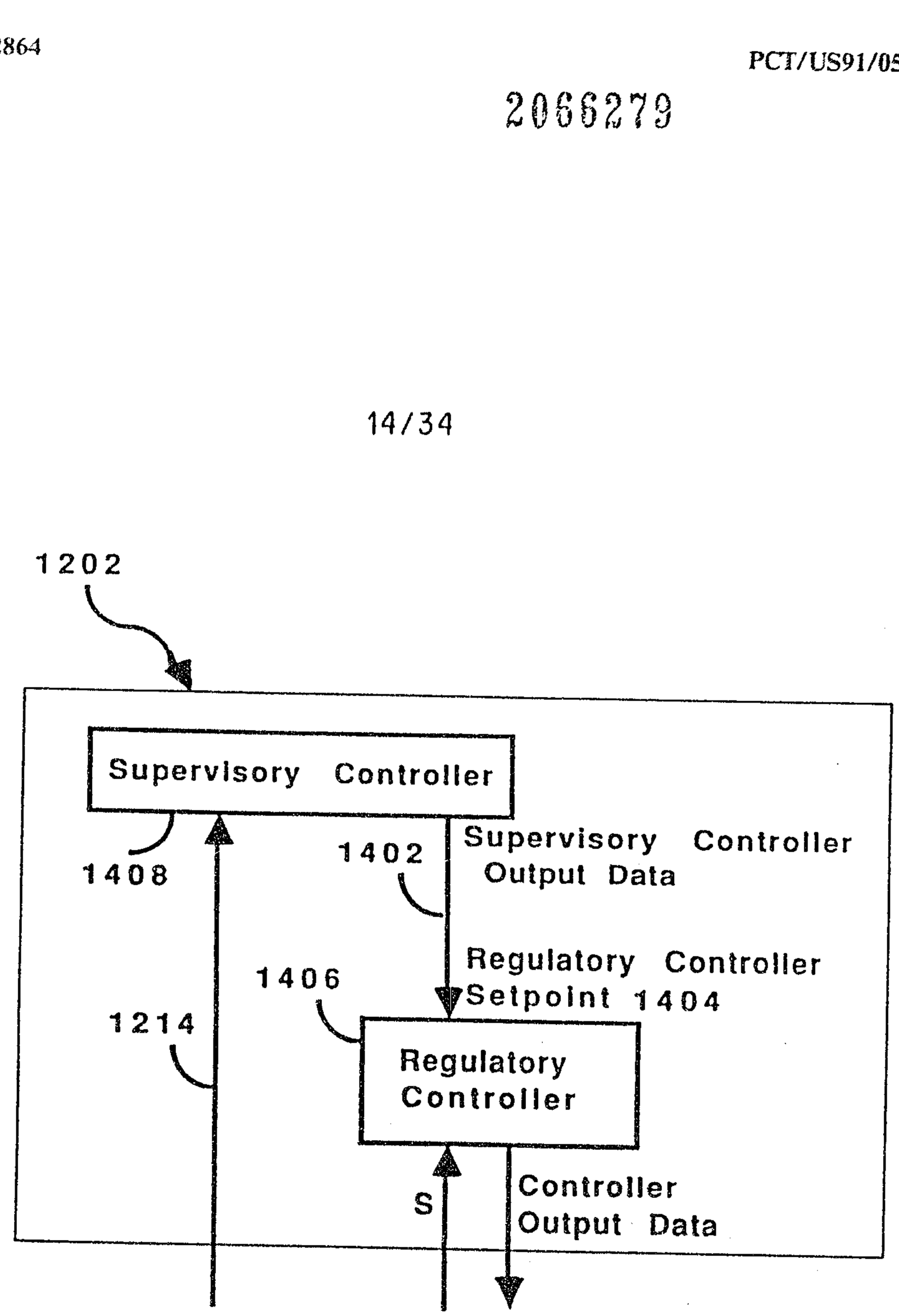

1202
Supervisory Controller
1408
1402
Supervisory Controller Output Data
Regulatory Controller Setpoint 1404
1406
1214
Regulatory Controller
S
Controller Output Data

FIG. 14

2066279

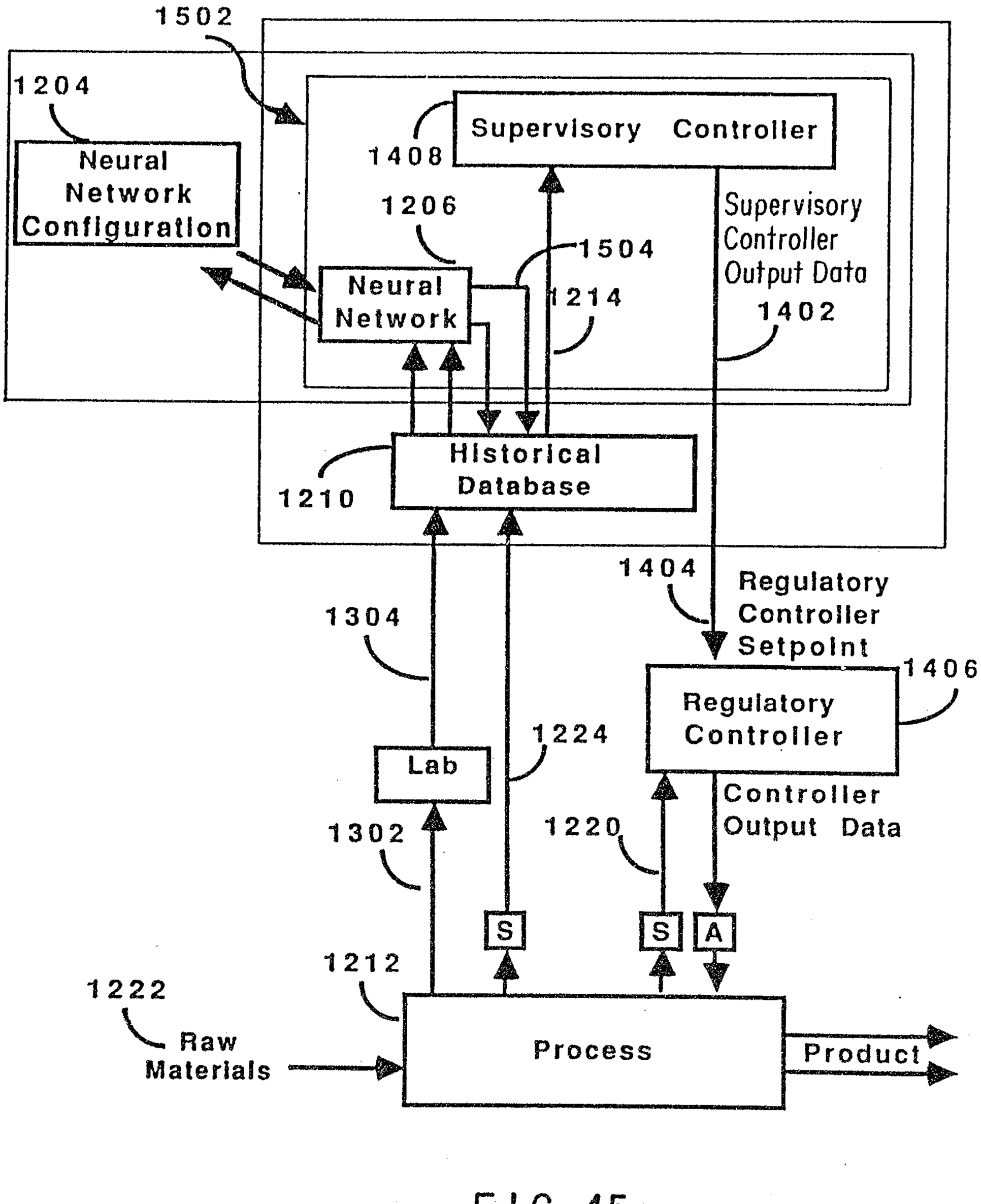


FIG. 15

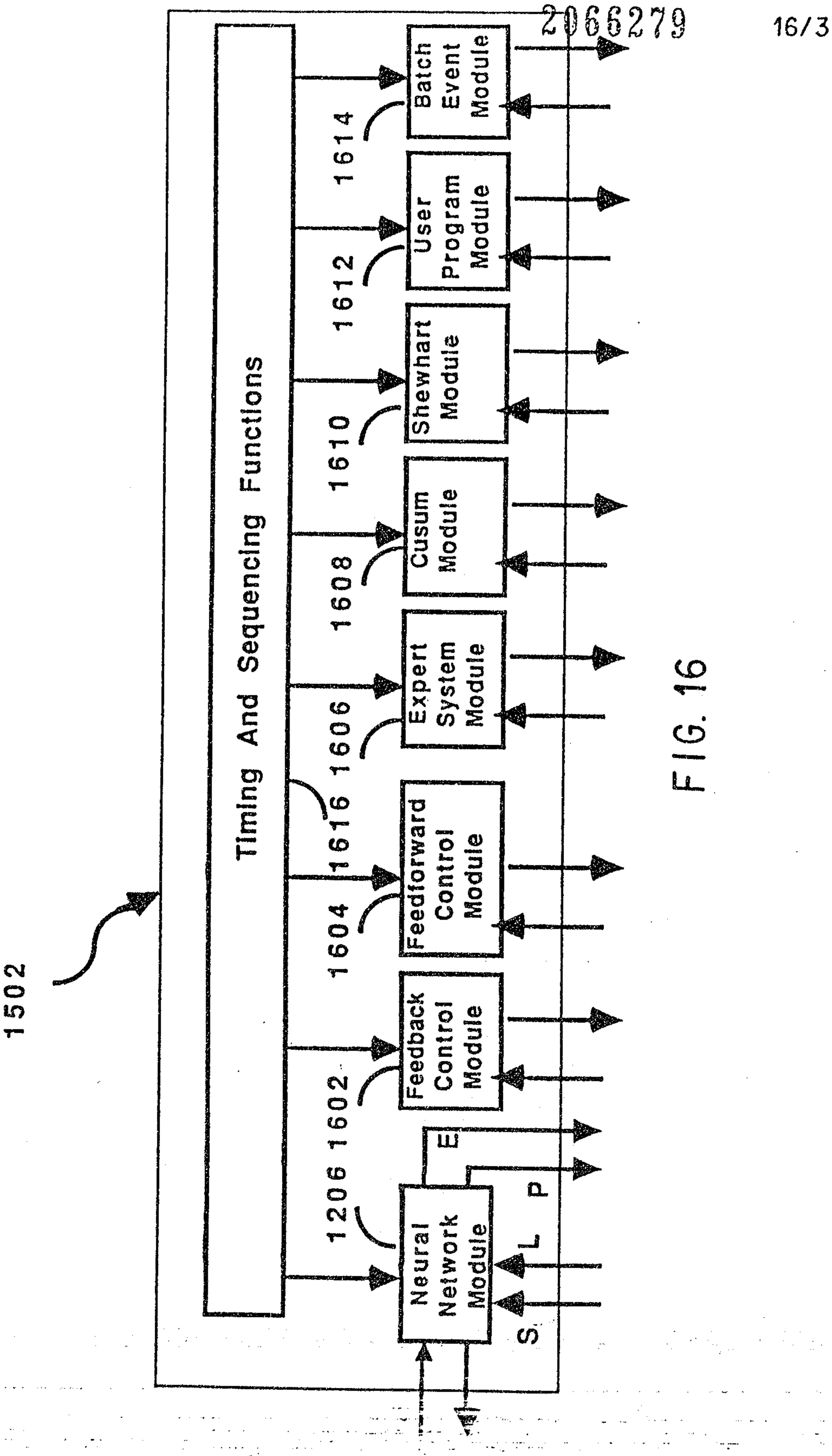

1502
Timing And Sequencing Functions
1614
Batch Event Module
1612
User Program Module
1610
Shewhart Module
1608
Cusum Module
1606
Expert System Module
1616
Feedforward Control Module
1604
Feedback Control Module
1602
1206
Neural Network Module
E
P
L
S

FIG. 16

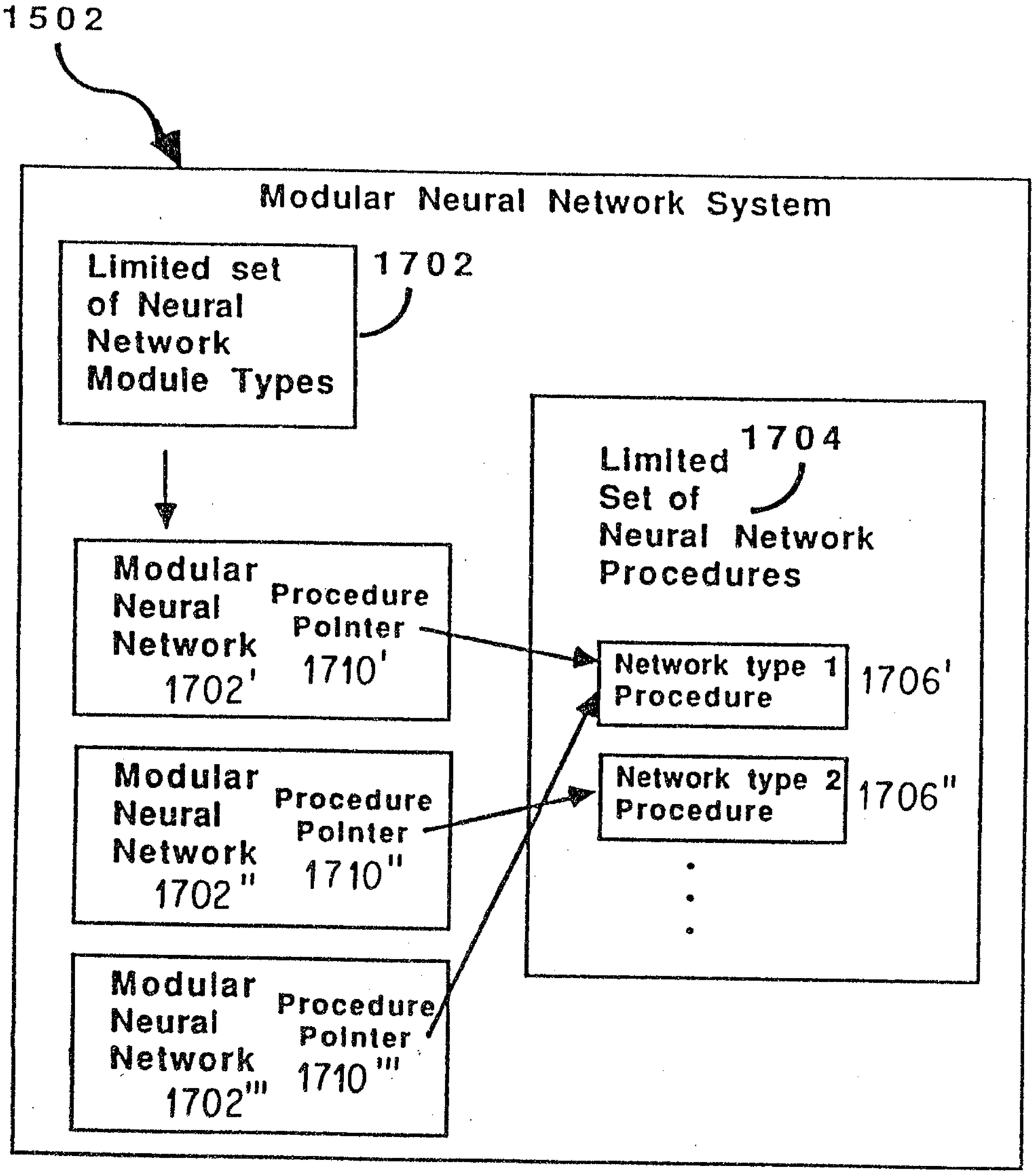

1502
Modular Neural Network System
Limited set of Neural Network Module Types
1702
Modular Neural Network 1702'
Procedure Pointer 1710'
Modular Neural Network 1702"
Procedure Pointer 1710"
Modular Neural Network 1702'''
Procedure Pointer 1710'''
Limited Set of Neural Network Procedures
1704
Network type 1 Procedure
1706'
Network type 2 Procedure
1706"

FIG. 17

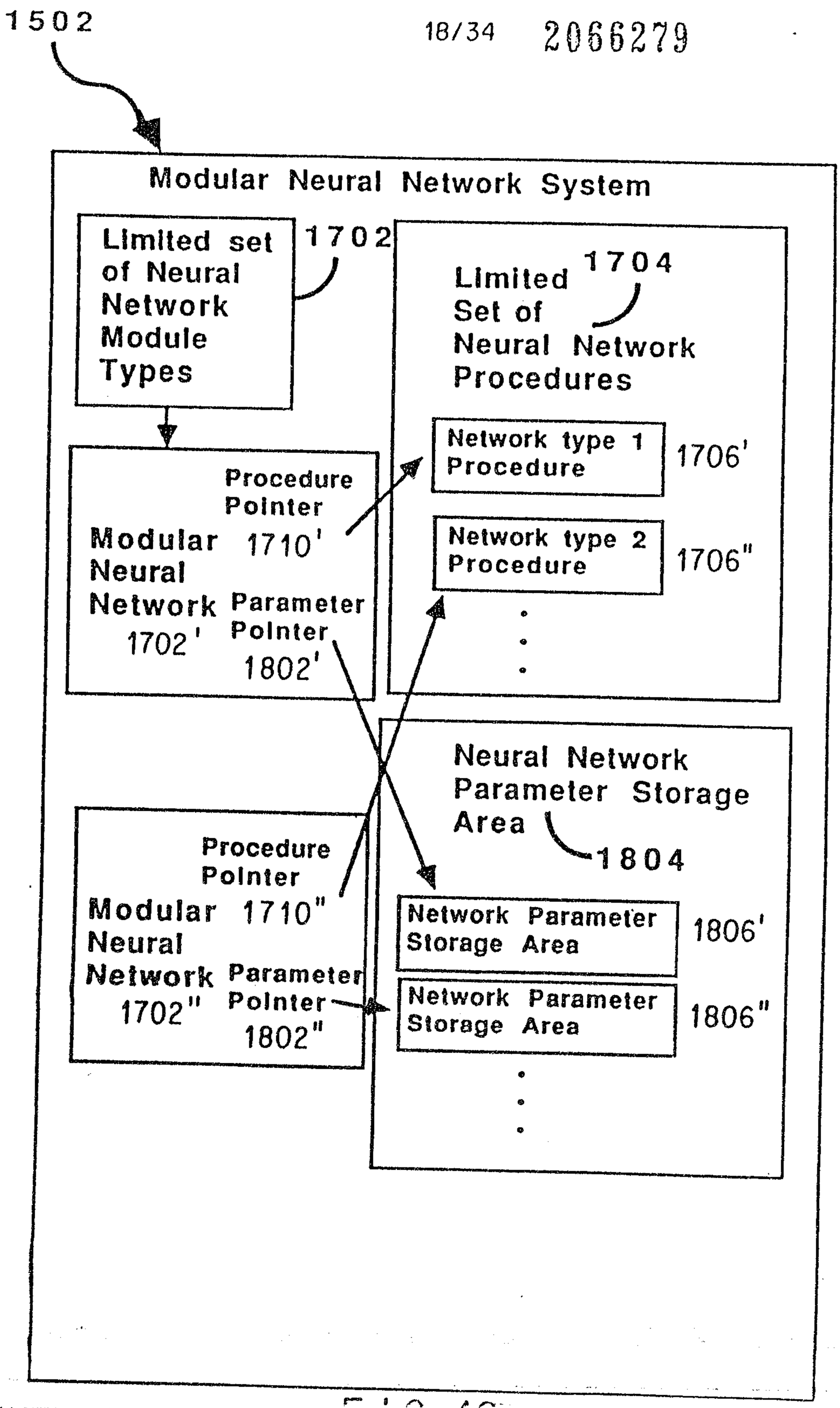

1502
Modular Neural Network System
Limited set of Neural Network Module Types
1702
Limited Set of Neural Network Procedures
1704
Network type 1 Procedure
1706'
Network type 2 Procedure
1706"
Procedure Pointer
1710'
Modular Neural Network
1702'
Parameter Pointer
1802'
Neural Network Parameter Storage Area
1804
Procedure Pointer
1710"
Modular Neural Network
1702"
Parameter Pointer
1802"
Network Parameter Storage Area
1806'
Network Parameter Storage Area
1806"

FIG. 18

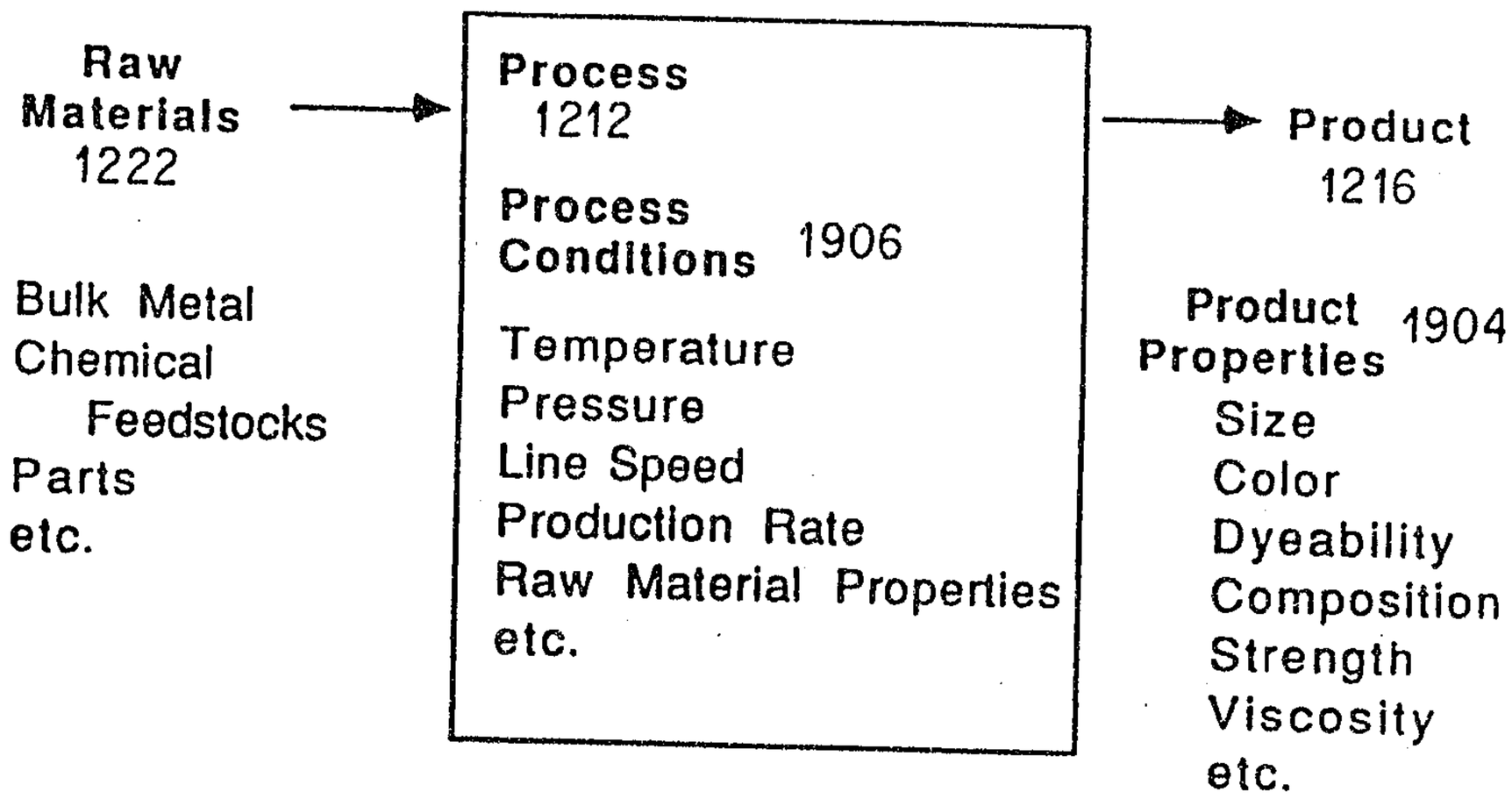

Raw Materials 1222
Bulk Metal
Chemical
Feedstocks
Parts
etc.
Process 1212
Process Conditions 1906
Temperature
Pressure
Line Speed
Production Rate
Raw Material Properties
etc.
Product 1216
Product Properties 1904
Size
Color
Dyeability
Composition
Strength
Viscosity
etc.

FIG. 19

2066279

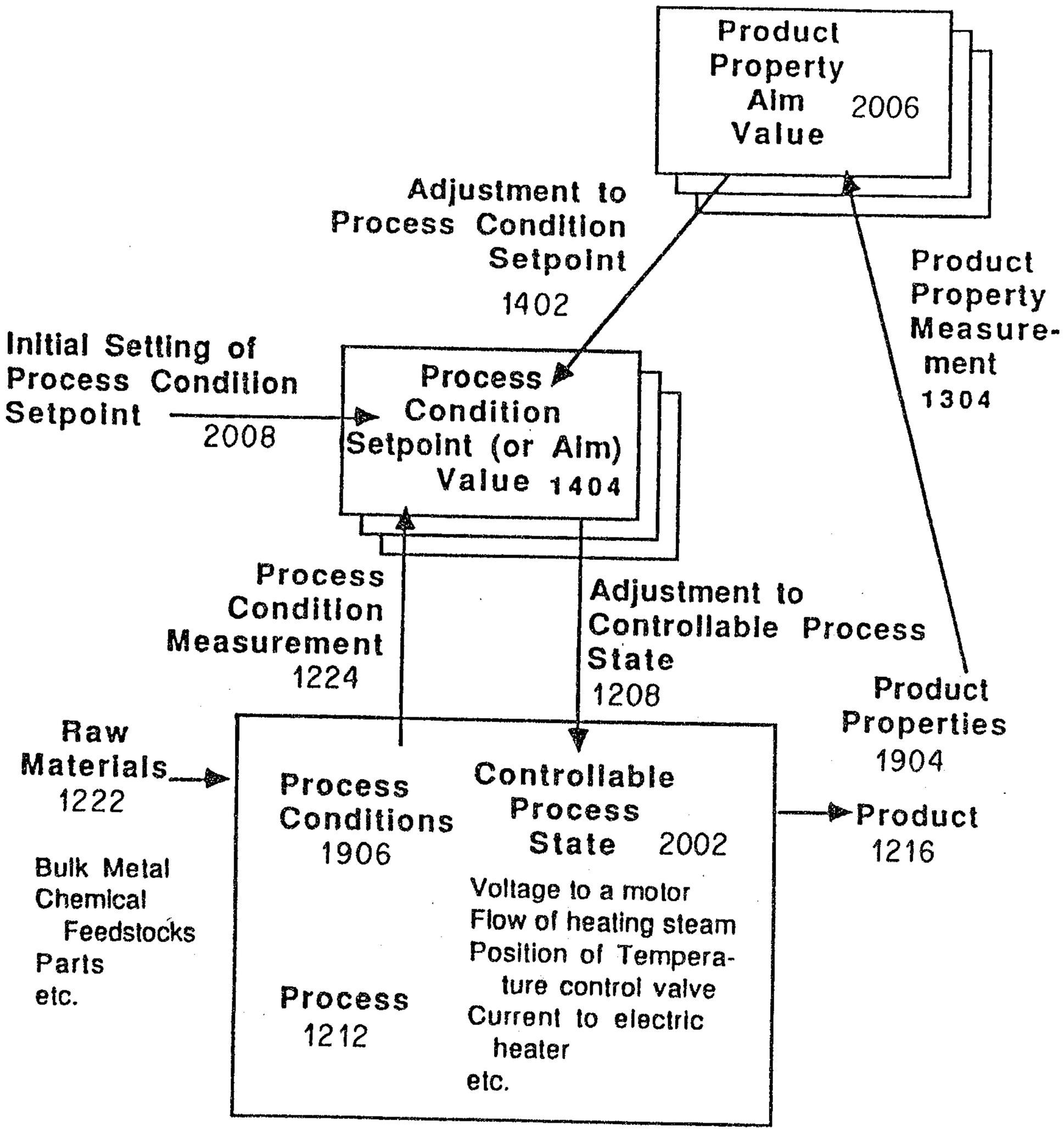


FIG. 20

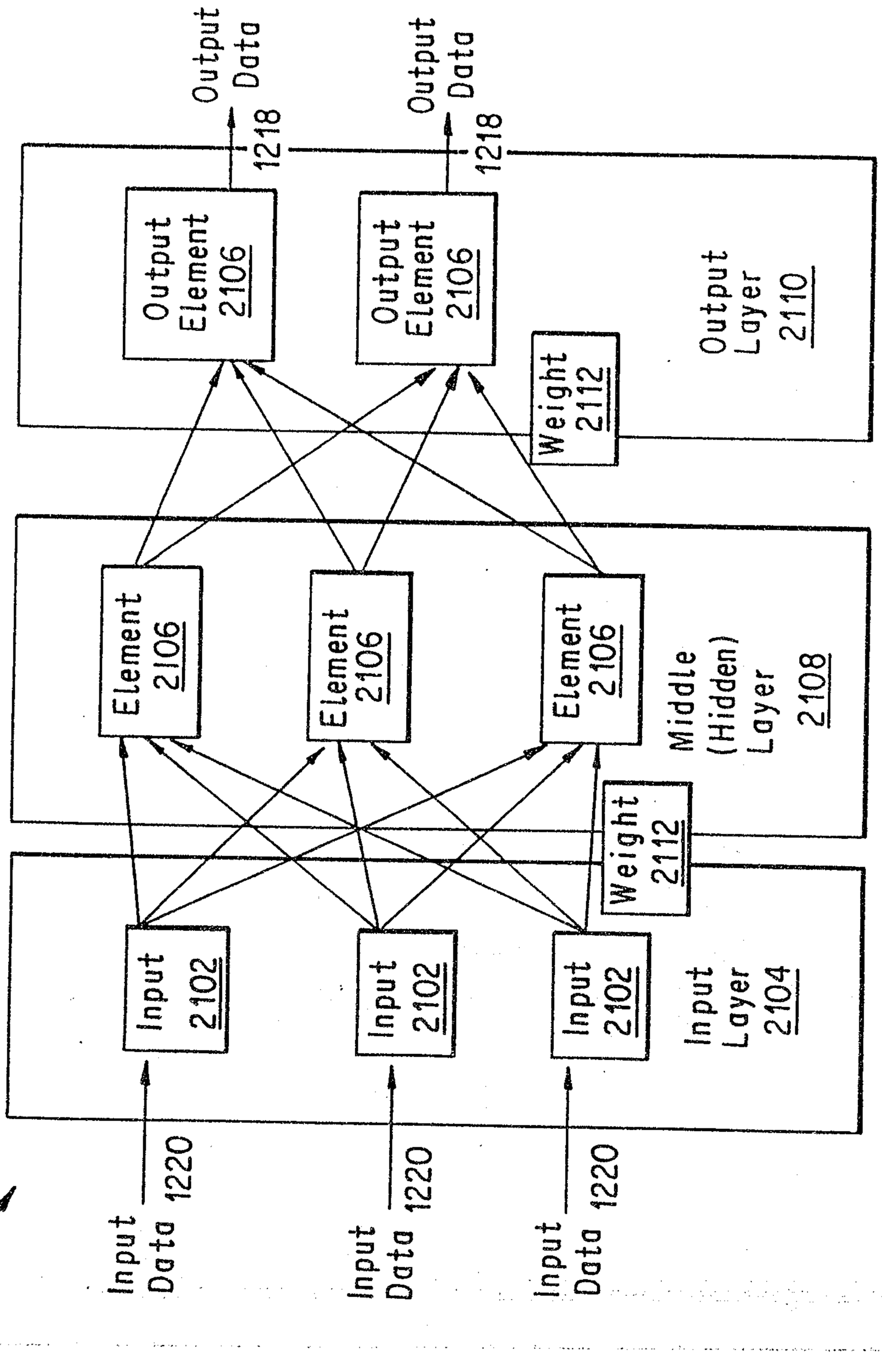

Output Data 1218
Output Data 1218
Output Element 2106
Output Element 2106
Output Layer 2110
Weight 2112
Element 2106
Element 2106
Element 2106
Middle (Hidden) Layer 2108
Weight 2112
Input 2102
Input 2102
Input 2102
Input Layer 2104
Input Data 1220
Input Data 1220
Input Data 1220
1206

FIG. 21

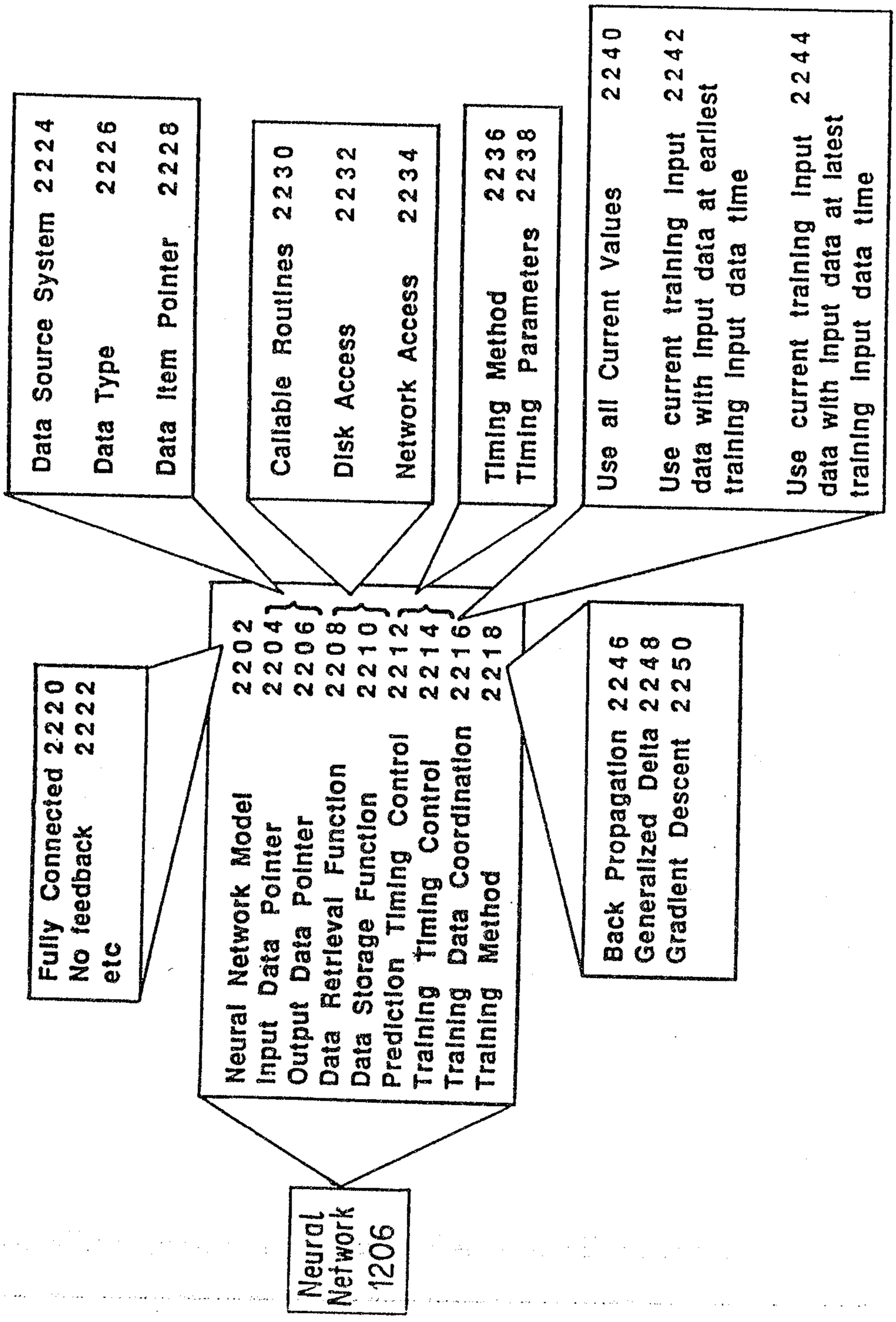

Neural Network 1206
Neural Network Model 2202
Input Data Pointer 2204
Output Data Pointer 2206
Data Retrieval Function 2208
Data Storage Function 2210
Prediction Timing Control 2212
Training Timing Control 2214
Training Data Coordination 2216
Training Method 2218
Fully Connected 2220
No feedback 2222
etc
Data Source System 2224
Data Type 2226
Data Item Pointer 2228
Callable Routines 2230
Disk Access 2232
Network Access 2234
Timing Method 2236
Timing Parameters 2238
Use all Current Values 2240
Use current training Input data with Input data at earliest training Input data time 2242
Use current training Input data with Input data at latest training Input data time 2244
Back Propagation 2246
Generalized Delta 2248
Gradient Descent 2250

FIG. 22

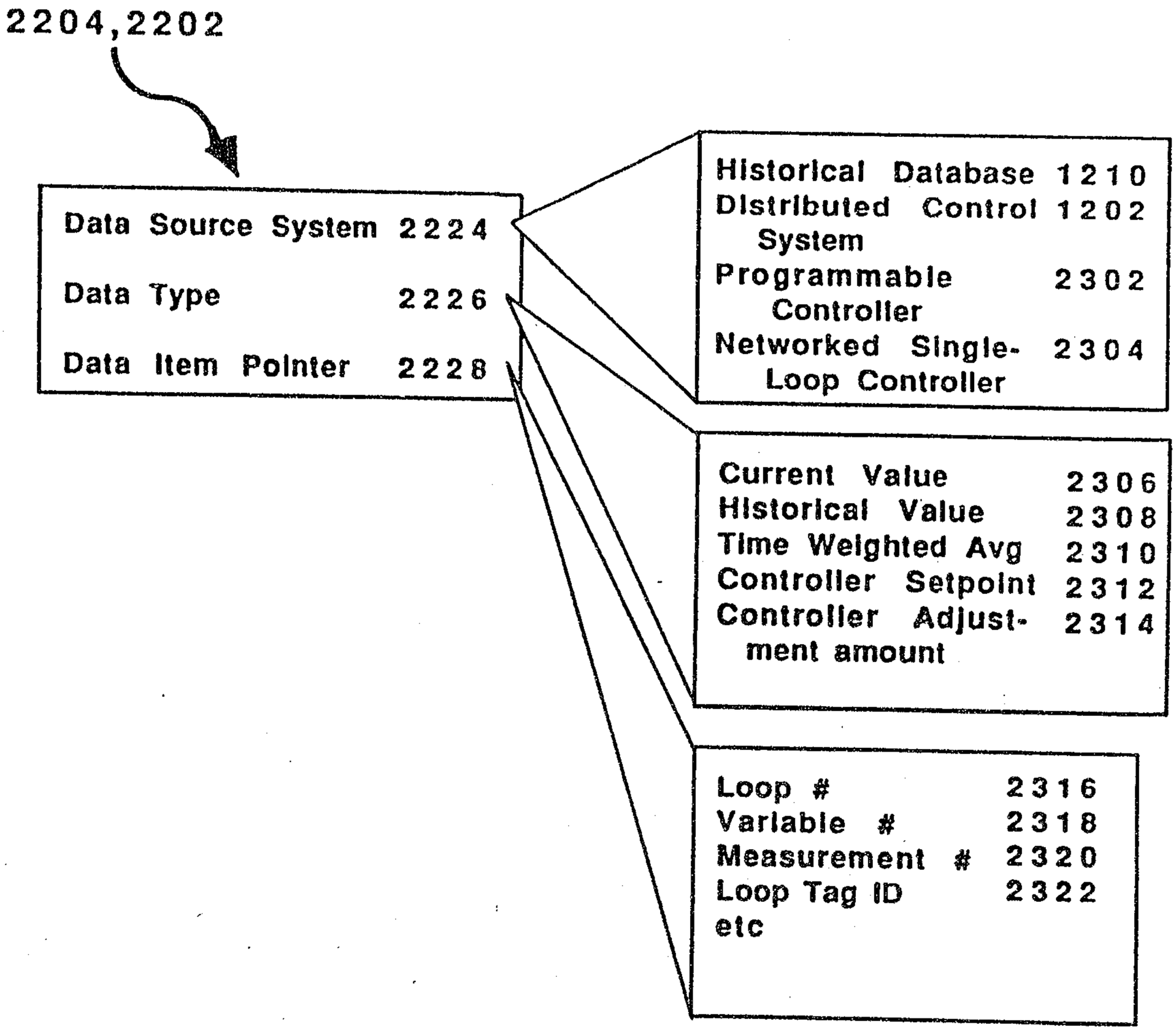

2204,2202
Data Source System 2224
Data Type 2226
Data Item Pointer 2228
Historical Database 1210
Distributed Control System 1202
Programmable Controller 2302
Networked Single-Loop Controller 2304
Current Value 2306
Historical Value 2308
Time Weighted Avg 2310
Controller Setpoint 2312
Controller Adjustment amount 2314
Loop # 2316
Variable # 2318
Measurement # 2320
Loop Tag ID 2322
etc

FIG. 23

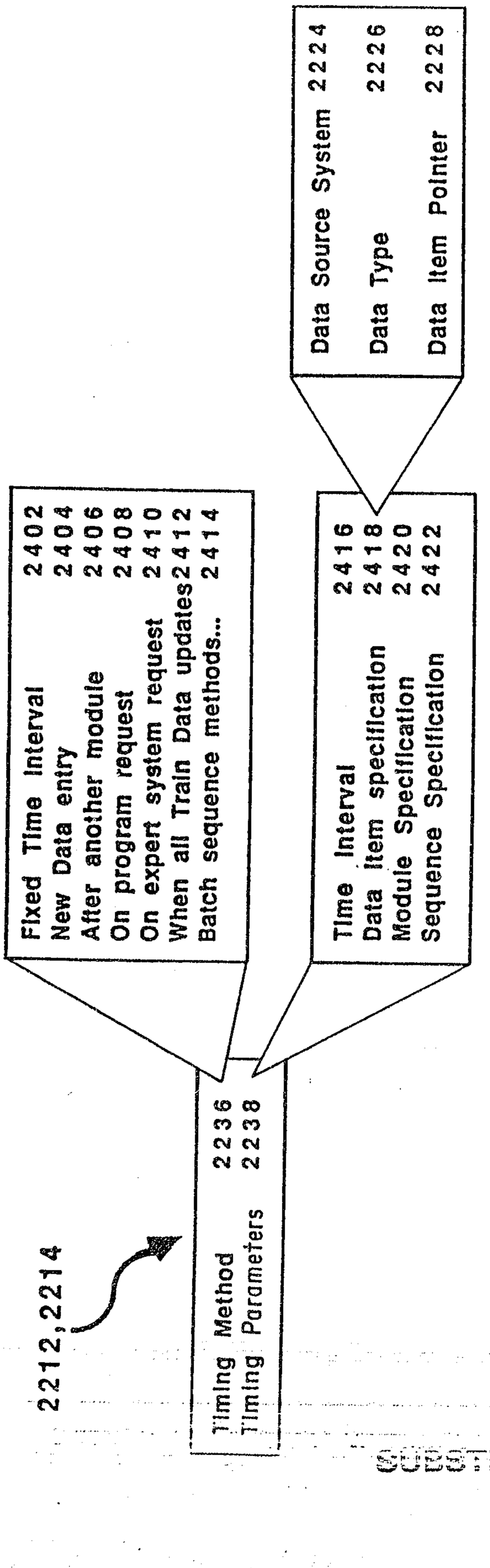

Data Source System 2224
Data Type 2226
Data Item Pointer 2228
Fixed Time Interval 2402
New Data entry 2404
After another module 2406
On program request 2408
On expert system request 2410
When all Train Data updates 2412
Batch sequence methods... 2414
Time Interval 2416
Data Item specification 2418
Module Specification 2420
Sequence Specification 2422
2212,2214
Timing Method 2236
Timing Parameters 2238

FIG. 24

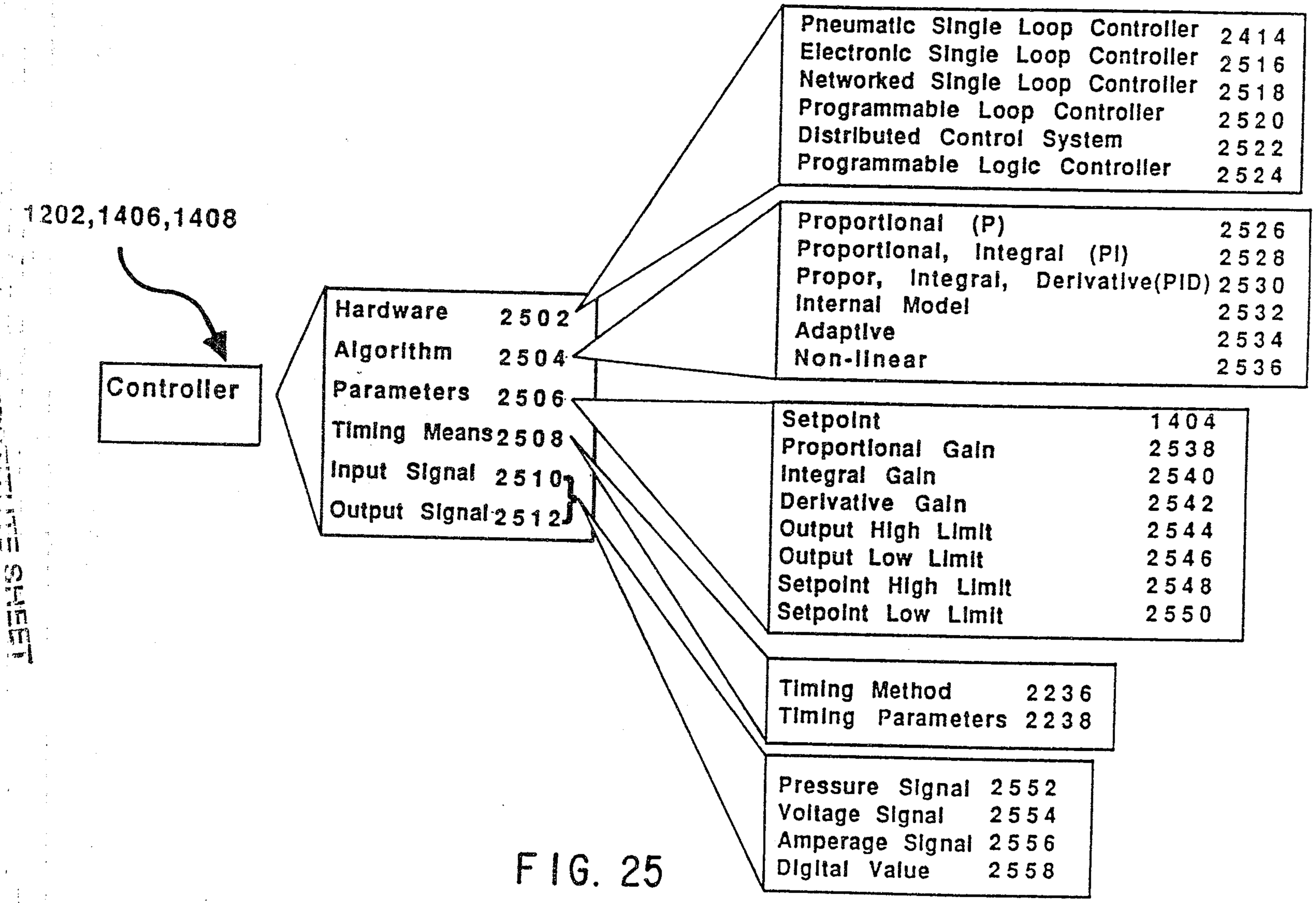
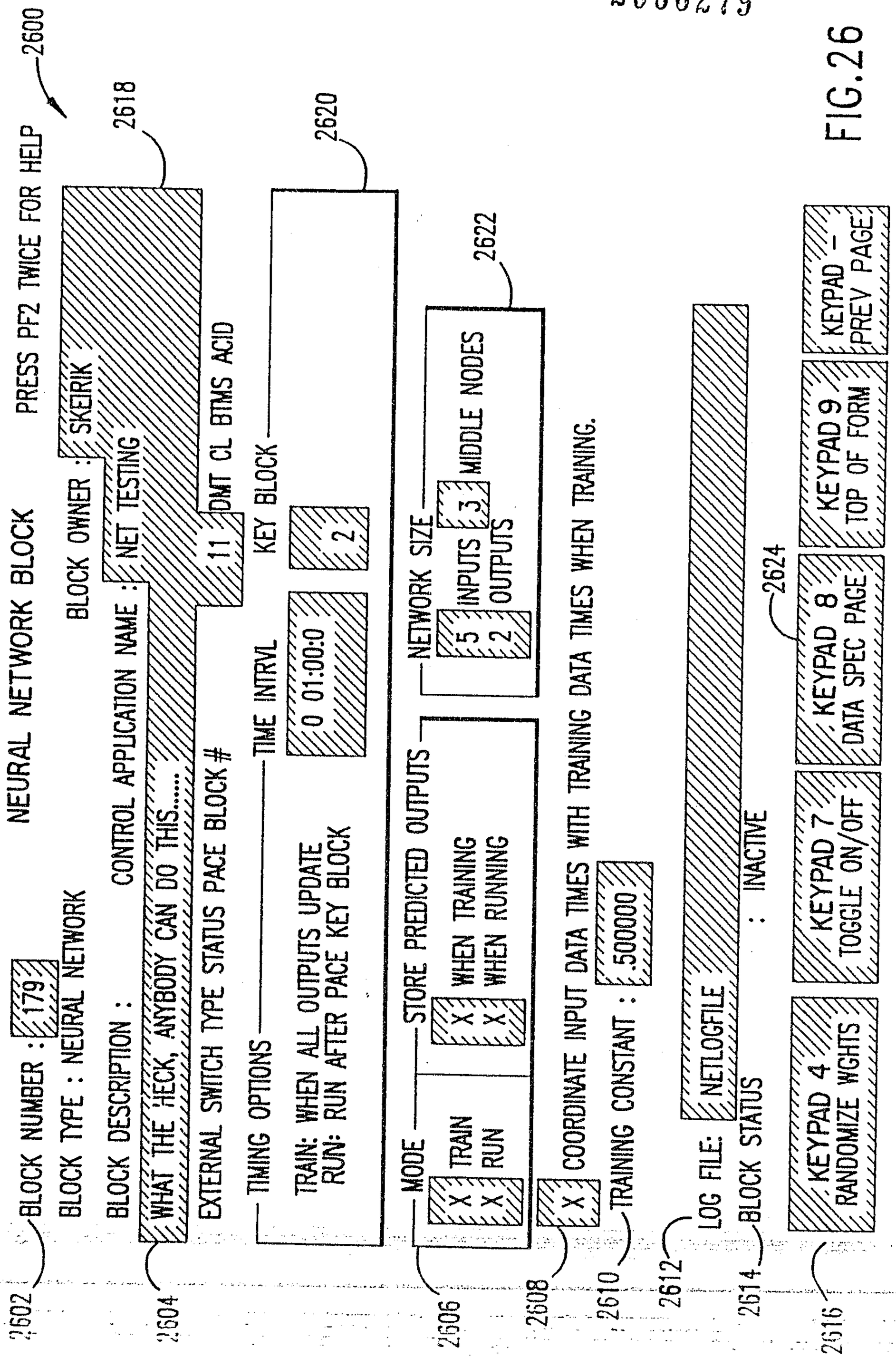


FIG. 25

SUBSTITUTE SHEET

2066279

FIG.26



SUBSTITUTE SHEET

2700

NEURAL NETWORK BLOCK PAGE 2 – NETWORK INPUTS AND OUTPUTS

NETWORK INPUTS (SCROLLABLE)

1 DMT VANTAGE TIME WT AVERAGE VAR # 1105 XYLENE COL FEED MPPH
OLDEST TIME 0 01:00:00 NEWEST TIME 0 00:10:00
HIGH LIMIT 150.0 LOW LIMIT: 50. MAX CHANGE 25.0

2702

NETWORK TRAINING INPUTS (SCROLLABLE)

1 DMT VANTAGE CURRENT VALUE VAR # 2081 XYL COL BTMS MFB PPM
HIGH LIMIT 2500.0 LOW LIMIT: 1500.0 MAX CHANGE 250.0

2704

NETWORK OUTPUTS (SCROLLABLE)

1 DMT VANTAGE CUR VALUE UNCOMPRSD VAR # 9055 NN PRED XYL BTM MFB PPM
HIGH LIMIT 2500.0 LOW LIMIT: 1000.0 MAX CHANGE 250.0

2706

SUMMED ERROR OUTPUT (NOT SCROLLABLE)

DMT VANTAGE CUR VALUE UNCOMPRSD VAR # 9056 NN XYL COL MODEL ERROR
HIGH LIMIT 1.0 LOW LIMIT: 0.00 MAX CHANGE 1.0

2708

USE UP/DOWN ARROW TO SCROLL THRU INPUTS.
USE GOLD UP/DOWN ARROW TO MOVE BETWEEN BOXES.

KEYPAD – PREV PAGE

FIG.27

SUBSTITUTE SHEET

NEURAL NETWORK BLOCK PAGE 2 – NETWORK INPUTS AND OUTPUTS

NETWORK INPUTS(SCROLLABLE)

1 DMT PACE TIME WT AVERAGE VAR # 1105 XYLENE COL FEED MPPH

OLDEST TIME 0 01:00:00 NEWEST TIME 0 00:10:00

HIGH LIMIT 150.0 LOW LIMIT: 50. MAX CHANGE 25.0

V

NETWORK TRAINING INPUTS (SCROLLABLE)

1 DMT VANTAGE CURRENT VALUE VAR # 2081 XYL COL BTMS MFB PPM

USE ARROW KEYS AND RETURN
TO SELECT ONE CHOICE.

DMT PACE
DMT VANTAGE
TPA PLANT PCS
DMT PLANT PCS
CRYSTAR VANTAGE
CRYSTAR TDC

2802

USE KEYPAD 0 TO EXIT

HIGH LIMIT 1.0 LOW LIMIT: 0.00 MAX CHANGE 1.0

USE UP/DOWN ARROW TO SCROLL THRU INPUTS.
USE GOLD UP/DOWN ARROW TO MOVE BETWEEN BOXES.

KEYPAD – PREV PAGE

FIG.28

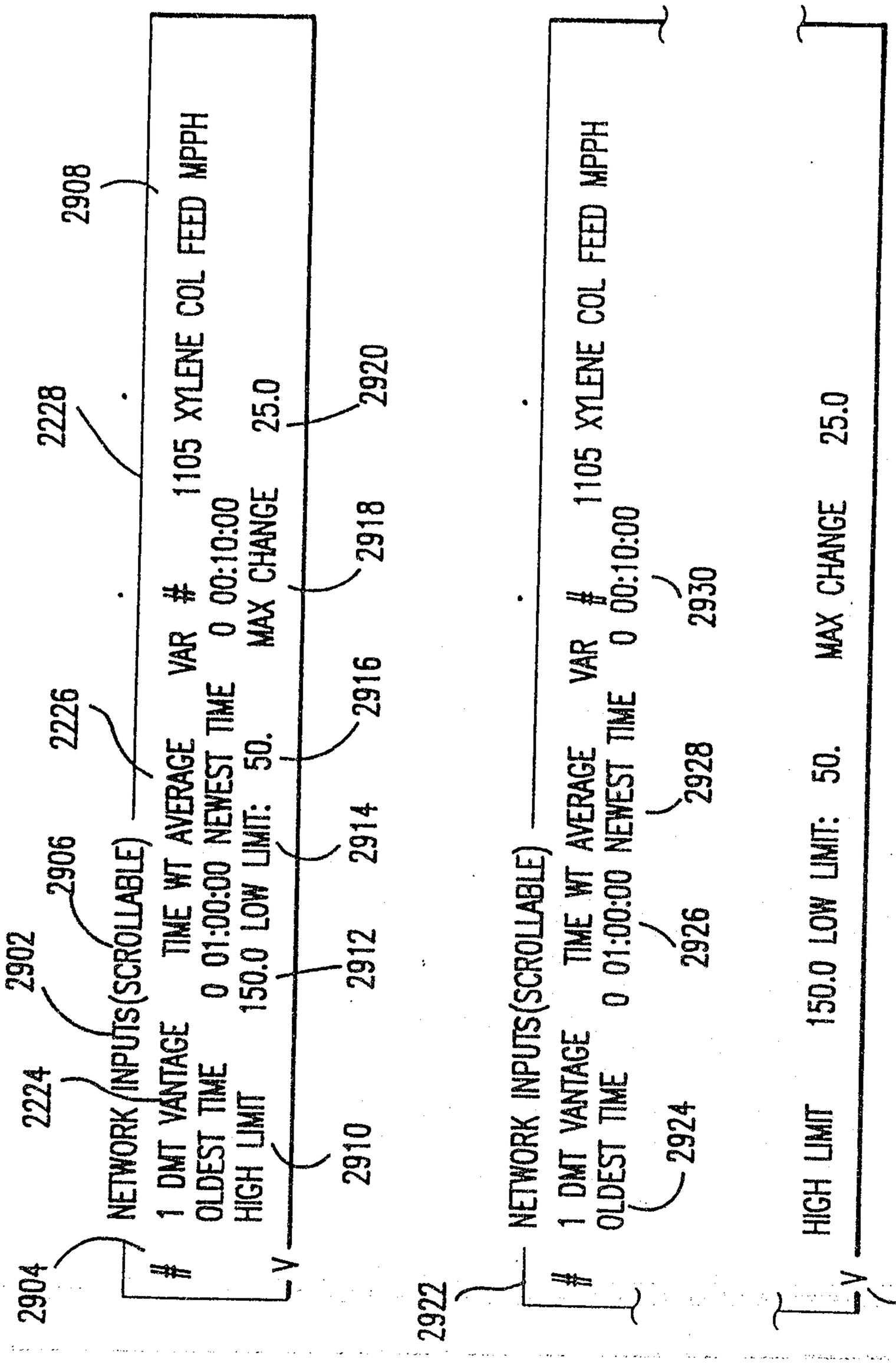

2904
NETWORK INPUTS(SCROLLABLE)
2224
2902
2906
2226
2228
2908
1 DMT VANTAGE
TIME WT AVERAGE
VAR # 1105 XYLENE COL FEED MPPH
OLDEST TIME 0 01:00:00 NEWEST TIME 0 00:10:00
HIGH LIMIT 150.0 LOW LIMIT: 50.
MAX CHANGE 25.0
V
2910
2912
2914
2916
2918
2920
2922
NETWORK INPUTS(SCROLLABLE)
1 DMT VANTAGE
TIME WT AVERAGE
VAR # 1105 XYLENE COL FEED MPPH
OLDEST TIME 0 01:00:00 NEWEST TIME 0 00:10:00
2924
2926
2928
2930
MAX CHANGE 25.0
HIGH LIMIT 150.0 LOW LIMIT: 50.
V
2932

FIG.29

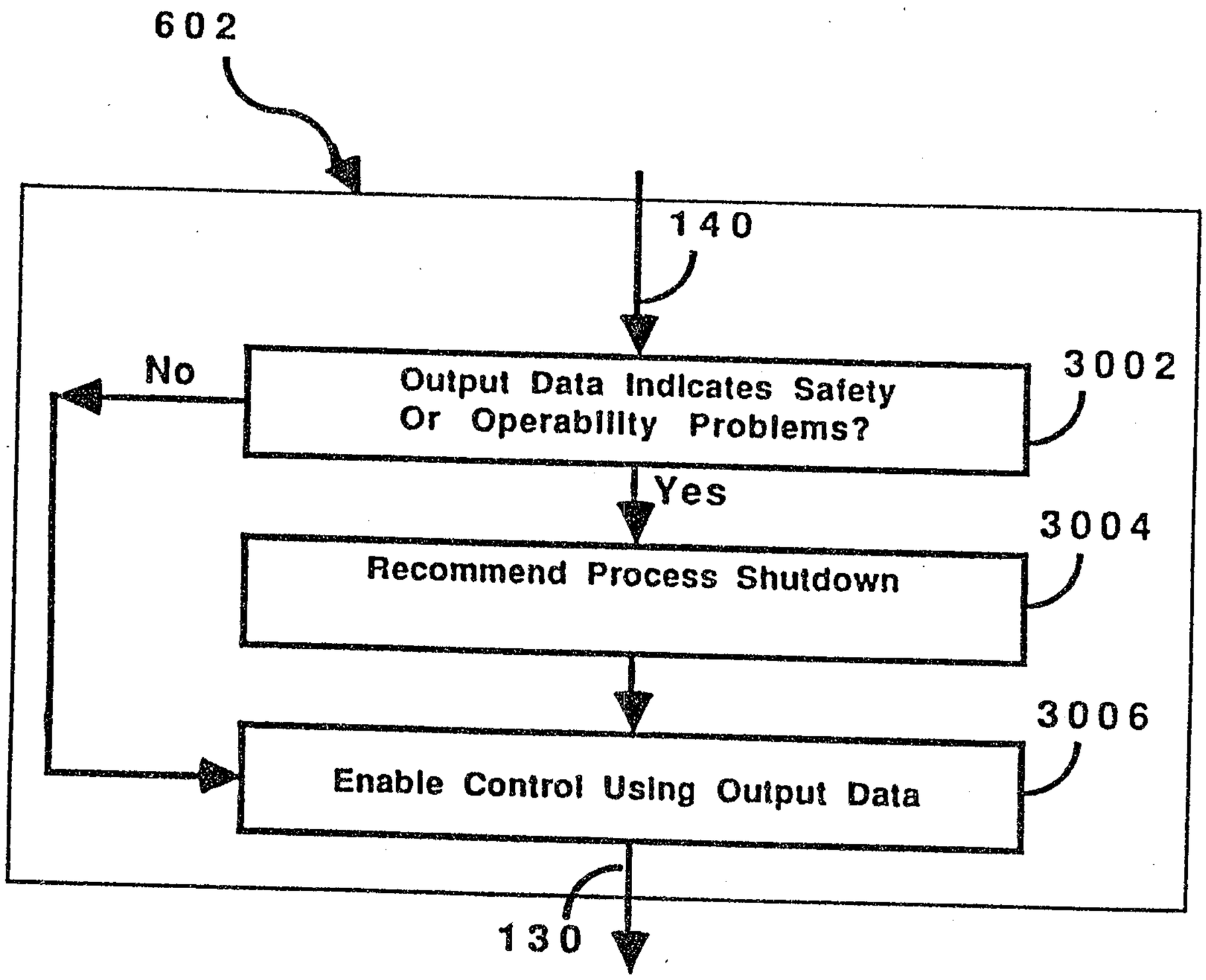

602
140
No
Output Data Indicates Safety Or Operability Problems?
3002
Yes
3004
Recommend Process Shutdown
3006
Enable Control Using Output Data
130

FIG. 30

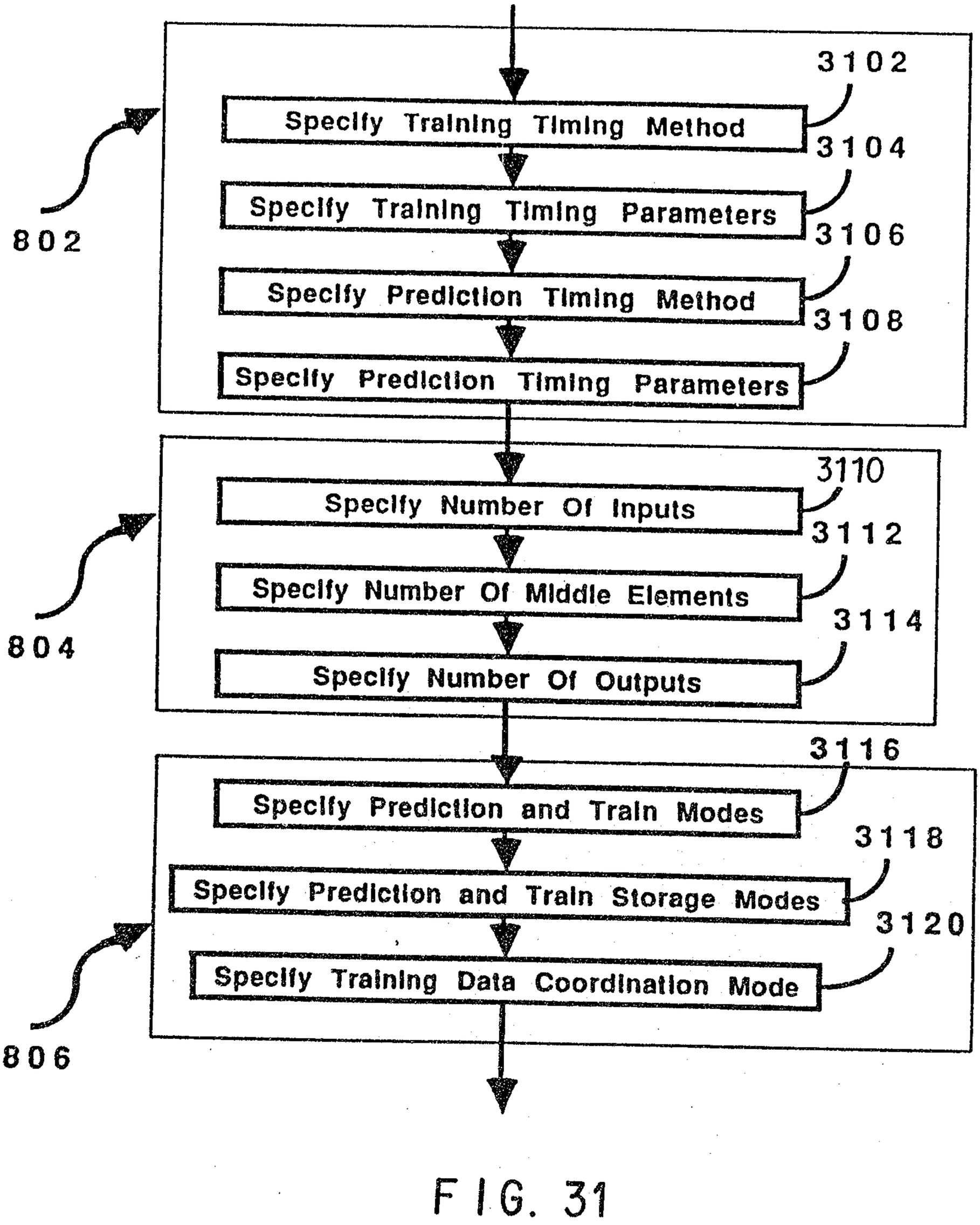

802
Specify Training Timing Method
3102
Specify Training Timing Parameters
3104
Specify Prediction Timing Method
3106
Specify Prediction Timing Parameters
3108
804
Specify Number Of Inputs
3110
Specify Number Of Middle Elements
3112
Specify Number Of Outputs
3114
806
Specify Prediction and Train Modes
3116
Specify Prediction and Train Storage Modes
3118
Specify Training Data Coordination Mode
3120

FIG. 31

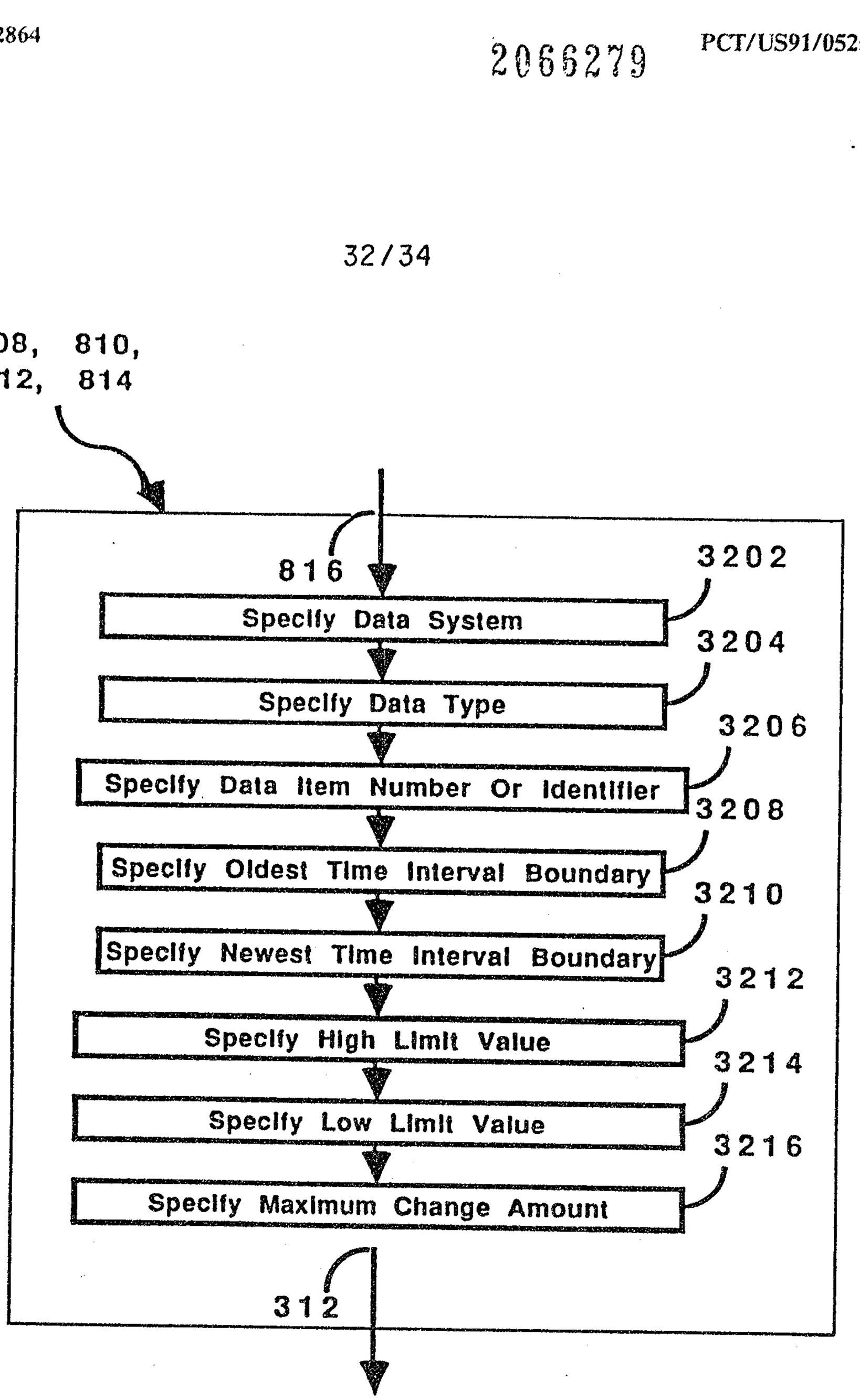

808, 810,
812, 814
816
3202
Specify Data System
3204
Specify Data Type
3206
Specify Data Item Number Or Identifier
3208
Specify Oldest Time Interval Boundary
3210
Specify Newest Time Interval Boundary
3212
Specify High Limit Value
3214
Specify Low Limit Value
3216
Specify Maximum Change Amount
312

FIG. 32

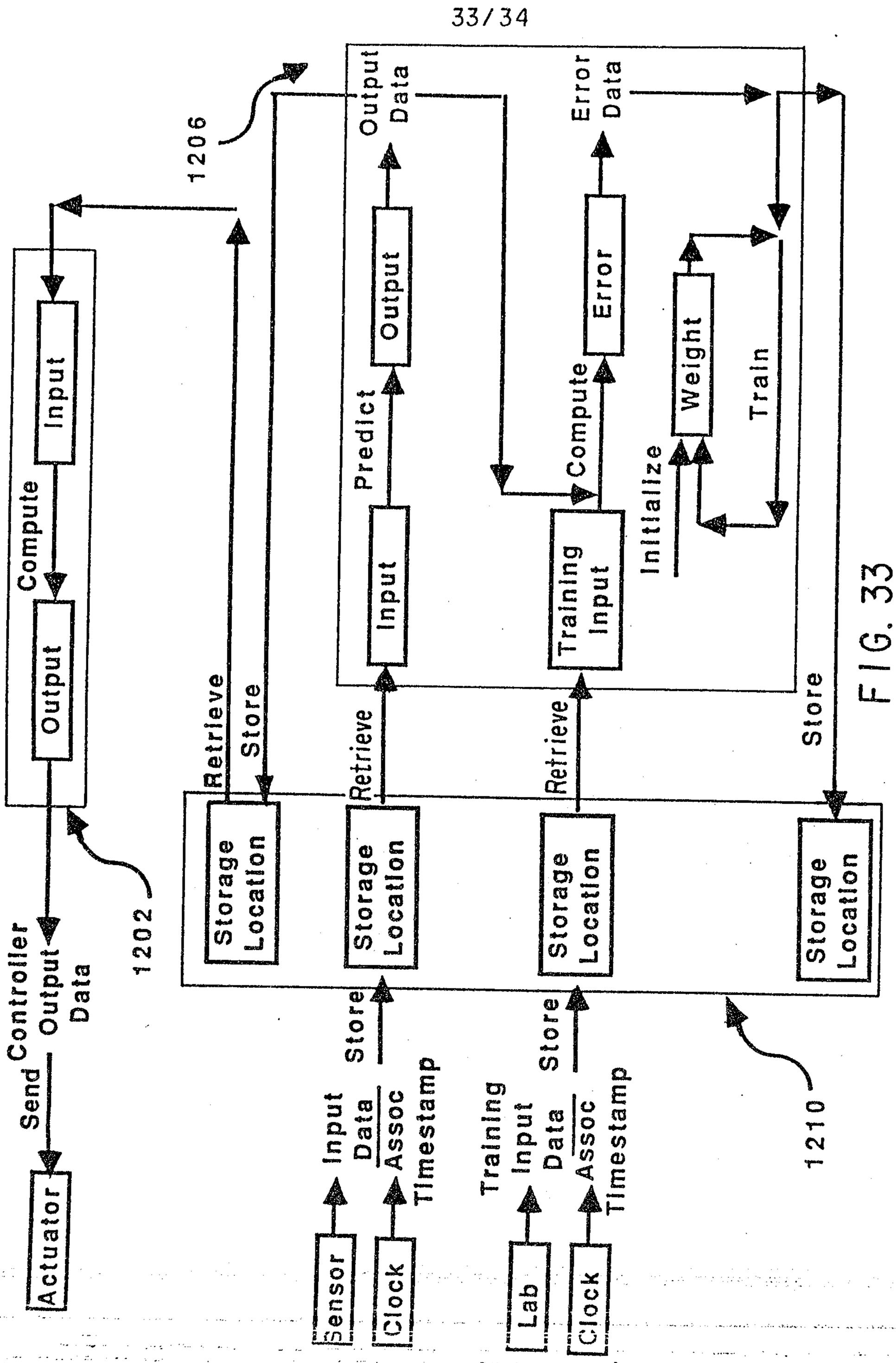

1206
Output Data
Error Data
Compute
Input
Output
Predict
Input
Output
Compute
Error
Training Input
Initialize
Weight
Train
Retrieve
Store
Retrieve
Retrieve
Store
Storage Location
Storage Location
Storage Location
Storage Location
1202
Send Controller Output Data
Actuator
Store
Input Data Assoc Timestamp
Sensor
Clock
Store
Training Input Data Assoc Timestamp
Lab
Clock
1210

FIG. 33

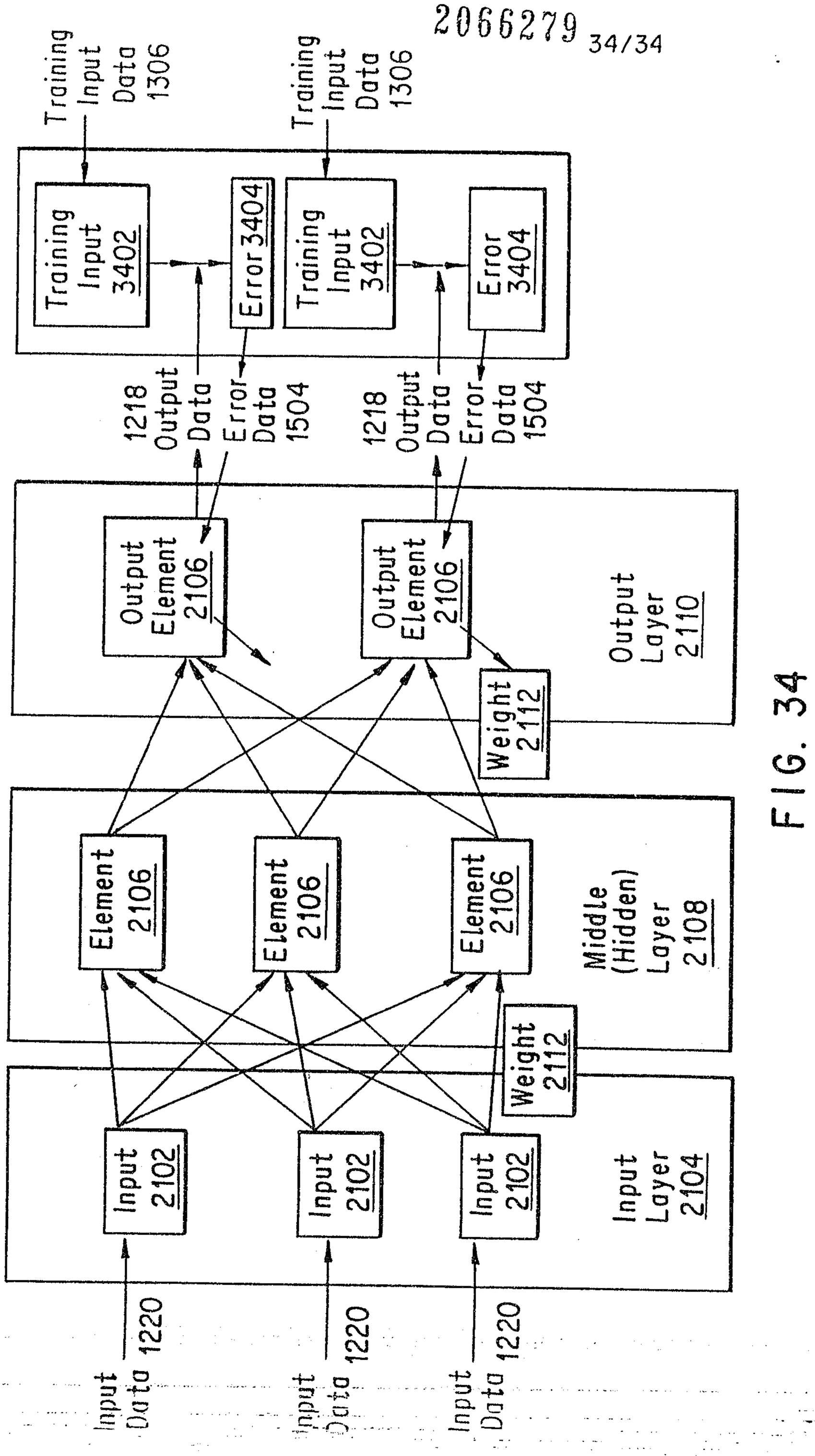
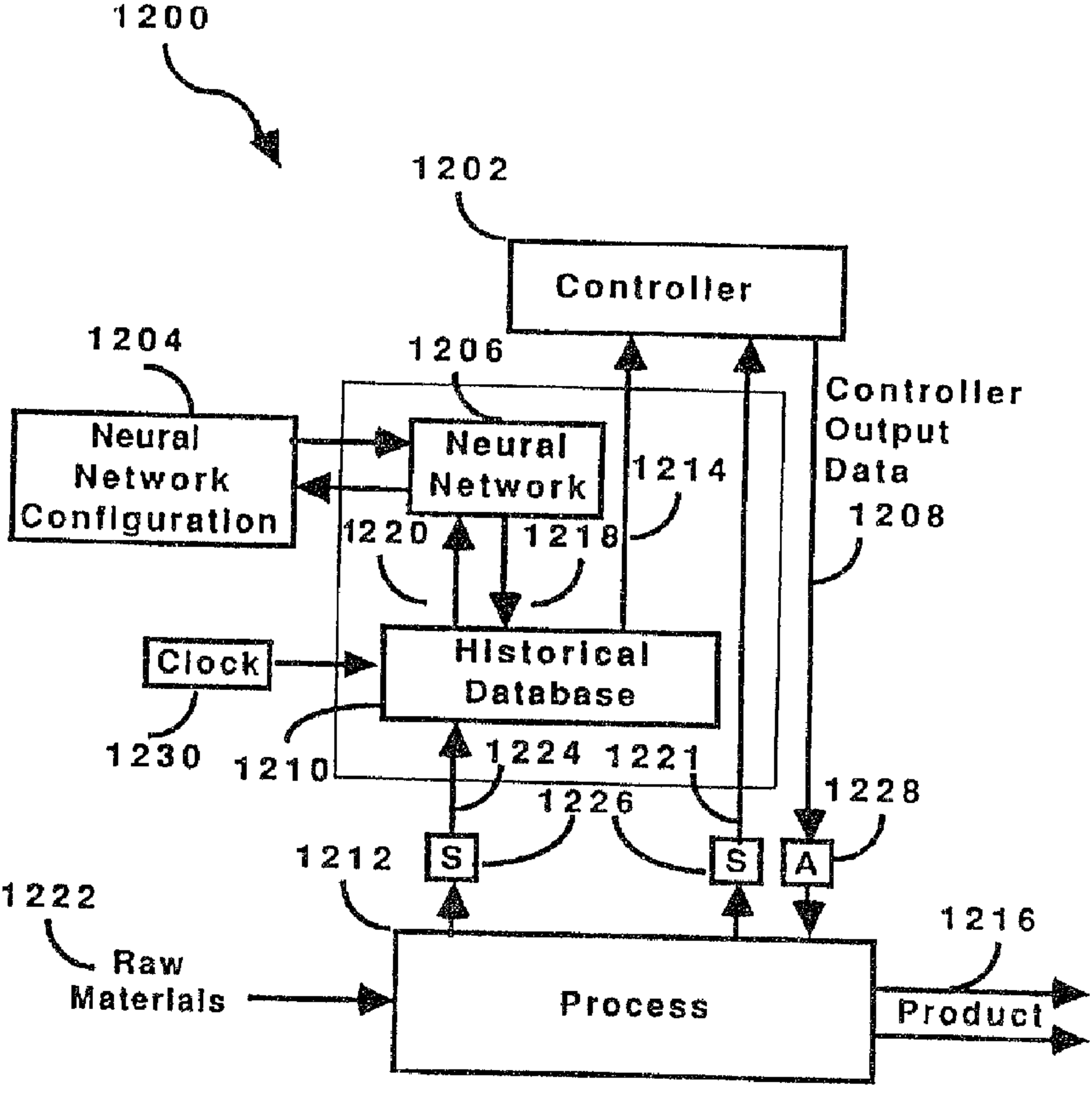

Training Input Data 1306
Training Input Data 1306
Training Input 3402
Error 3404
Training Input 3402
Error 3404
1218 Output Data
Error Data 1504
1218 Output Data
Error Data 1504
Output Element 2106
Output Element 2106
Output Layer 2110
Weight 2112
Element 2106
Element 2106
Element 2106
Middle (Hidden) Layer 2108
Weight 2112
Input 2102
Input 2102
Input 2102
Input Layer 2104
1206
Input Data 1220
Input Data 1220
Input Data 1220

FIG. 34


1200
1202
Controller
1204
1206
Neural Network Configuration
Neural Network
1214
Controller Output Data
1208
1220
1218
Clock
Historical Database
1230
1210
1224
1221
1226
1228
1212
S
S
A
1222
Raw Materials
1216
Process
Product