IV－21

Applicability of Neural Network Model for Estimating Traffic States on Freeway

NASSER POURMOALLEM, TAKASHI NAKATSUJI, and TAKASHI FUJIWARA
Faculty of Engineering, Hokkaido University

ABSTRACT. This paper describes a scheme in which a Neural-Kalman filtering model is used to a real-time traffic flow model for a freeway . A neural network model was integrated into a Kalman filtering model for to estimating traffic states on freeway. The Cremer model, which is a macroscopic traffic flow model combined with a Kalman filter, is revised using a neural network model. In this case, by using two different the multilayer neural networks, the derivative of state equations and observation equations that play an important role in correcting the estimates of state variables were easily obtained. This Neural-Kalman filtering model was applied to a road section on the Metropolitan Expressway in Tokyo and it was examined how precisely the method could work as compared with the original Cremer model.

Key Words: *Macroscopic traffic flow model, Cremer model, Kalman filter model, Multilayer Neural Network model, Neural-Kalman filtering model.*

1. INTRODUCTION

A Kalman filter is a powerful tool for analyzing a dynamic system[1]. But it requires us to define a set of state and observation equations in advance. Unfortunately, in many dynamic systems, it is often difficult to define such functions theoretically. The Cremer model[1], in which the Payne-type macroscopic model[2] is combined with a Kalman filter, is one of such dynamic models for estimating traffic states on a freeway. In the model, the state equations that represent flow dynamics of state variables, such as density and space mean speed, play important roles, as well as the observation equations that relate the observed variables, such as flow rate and time mean speed, to the state variables. So far, both state and observation equations used to be described theoretically using analytical functions. However, we often have some difficulty in describing them.

Some neural network models have some promising abilities: They can accurately describe non-linear phenomena; they can organize their structures flexibly according to observed data; also, they can deal with any kind of numbers, not only quantitative numbers, but also qualitative numbers, even fuzzy numbers. If a neural network model is applied to this dynamic estimation problem, it can easily establish a steady non-linear input-output relationship between the state and observation variables. It requires no preliminary knowledge of both equations. What we have to do is to prepare input and output patterns. Although the neural network model is one of statistical regression approaches, it can flexibly identify the system even when input and output data are too few or too many. Another superior characteristic of the neural network model is that it is easy to produce the derivative of the system equations since the neural network model is described by the combination of elementary functions.

This paper aims to investigate the ability of a neural network model for estimating traffic states on a freeway. Intending to extend the Cremer model, we examined how accurately a multilayer neural network model could describe the dynamic system. First, we present the fundamental theoretical backgrounds. Next, we explain how we can deal with the above-mentioned problems using a multilayer neural model. Then, we present numerical experiments. We investigated how effective the Neural-Kalman filtering model was. We applied the method to a road section on the Metropolitan Expressway in Tokyo and compared the results with those estimated by the original Cremer model. Finally, we conclude with a brief summary.

2. THEORETICAL BACKGROUNDS

2.1. Cremer Model[1]

2.1.1. Macroscopic Traffic Flow Model[2]

Macroscopic traffic flow simulation models describe traffic phenomena aggregatively based on traffic variables, such as density, average speed, and flow rate. We divided a road section on a freeway into several segments, as shown in Fig. 1.

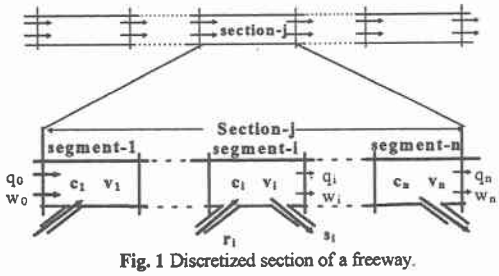

Fig. 1 Discretized section of a freeway.

The Payne-type model that decries the traffic flow dynamic is employed in the Cremer Model. Using the aggregative flow variables, the dynamic equations are defined as follows:

$$c_i(k+1) = c_i(k) + \frac{\Delta t}{\Delta l_i}\left[q_{i-1} - q_i + r_i - s_i\right]_{(k)} \tag{1}$$

$$v_i(k+1) = v_i(k) + \frac{\Delta t}{\tau}\left[V(c_i) - v_i\right]_{(k)} + \frac{\Delta t}{\Delta l_i}\left[v_i\left(v_{i-1} - v_i\right)\right]_{(k)} + \frac{\nu\,\Delta t}{\tau\,\Delta l_i}\left[\frac{c_i - c_{i+1}}{c_i + \kappa}\right]_{(k)} \tag{2}$$

where $c_i(k)$ is density of segment i at time k, $v_i(k)$ is space mean speed, and $q_i(k)$ is flow rate. $r_i(k)$ and $s_i(k)$ are possible entrance and exit ramp flow rates. Δl_i is segment length and Δt is time interval of simulation. τ is time constant, ν is sensitivity factor, and κ is density constant. $V(c_i(k))$ in Eq. (2) is the steady-state speed, which is defined by a density-speed characteristic curve:

$$V\left(c_i(k)\right) = v_f\left[1 - \left(\frac{c_i(k)}{c_{max}}\right)^{l-1}\right]^{\frac{1}{m-1}} \tag{3}$$

where v_f is free speed, c_{max} is jam density, l and m are sensitivity factors.

In the original Cremer model, flow rate $q_i(k)$ and time mean speed $w_i(k)$ are defined at the boundaries of segments. And they are estimated as a linearly-weighted average of products of density and space mean speed on the adjacent segments:

$$q_i(k) = \left[\alpha c_i v_i + (1-\alpha) c_{i+1} v_{i+1}\right]_{(k)} \tag{4}$$

$$w_i(k) = \left[\alpha v_i + (1-\alpha) v_{i+1}\right]_{(k)} \tag{5}$$

where α is the weighting factor ranging $0<\alpha<1$.

2.1.2. Kalman Filter[3]

Choosing density and space mean speed as the state variables vector x_k, and flow rate and time mean speed as the observation vector y_k, we can rewrite Eqs. (1) and (2) as a state equation and Eqs. (4) and (5) as an observation equation:

$$x_{k+1} = f(x_k) + \xi_k \tag{6}$$

$$y_k = g(x_k) + \zeta_k \tag{7}$$

where ξ_k and ζ_k are 2n-dimensional and 2m-dimensional noise vectors, respectively, where n and m are numbers of segments and observation points. Since Eqs. (6) and (7) are non-linear, we have to linearize them. They result in

$$\Delta x_{k+1} = \Phi_k \Delta x_k + \xi_k \tag{8}$$

$$\Delta y_k = \Psi_k \Delta x_k + \varsigma_k \tag{9}$$

where Φ_k, is a 2n×2n-dimensional matrix and Ψ_k is a 2m×2n-dimension al matrix defined as:

$$\Phi_k = \frac{\partial f}{\partial x} \tag{10}$$

$$\Psi_k = \frac{\partial g}{\partial x} \tag{11}$$

Calculating Φ_k and Ψ_k step by step, we can correct the state variables every time we obtain the observed data y_k

$$\hat{x}_k = \tilde{x}_k + K_k\left[y_k - \tilde{y}_k\right] \tag{12}$$

where

$$\tilde{x}_k = f\left(\hat{x}_{k-1}\right) \tag{13}$$

$$\tilde{y}_k = f\left(\tilde{x}_k\right) \tag{14}$$

The vector $\tilde{x}_k$ is referred to as the one-step predictor of x_k, and $\hat{x}_k$ as the filtered estimate of x_k. K_k is Kalman gain matrix at time k.

2.2. Multiple Neural Network Model[4]

We used a multiple neural network model as shown in Fig. 2, which consists of three layer; an input layer, an intermediate layer, and an output layer.

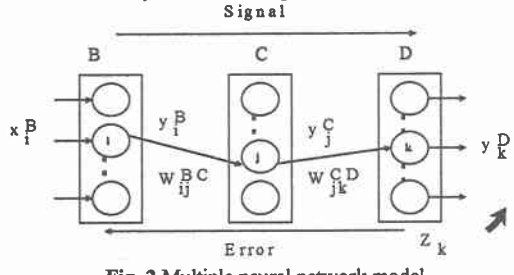

Fig. 2 Multiple neural network model.

x_i^B represents an input signal and y_k^D an output signal. The strengths of the connections W_{ij}^{BC} and W_{jk}^{BC} are called synaptic weights. The output signal y_k^D is calculated as

$$y_k^D = h\left(\sum_j W_{jk}^{CD} h\left(\sum_i W_{ij}^{BC} h\left(x_i^B\right)\right)\right) \tag{15}$$

where h(x) is the action that describes the non-liner behaviors of neurons. We used the sigmoid function as the activation function. For adjusting synaptic weights, we need target signals that are given by theoretical equations or observed data. We repeated the back-propagation operations until the following average squared sum of the between output signal y_k^D and target single z_k became sufficiently small:

$$J = \frac{1}{N_D}\sum_k^{N_D}\left(y_k^D - z_k\right)^2 \tag{16}$$

It should be noted here that it is very to produce the derivative of an output single y_k^D to an input single x_i^B. It follows:

$$\frac{\partial y_k^D}{\partial x_i^B} = y_k^D\left(1-y_k^D\right)\sum_j^j W_{jk}^{CD} y_j^C\left(1-y_j^C\right)W_{ij}^{BC} \qquad (17)$$

As will be shown later, this derivative constitutes components of matrices Φ_k and Ψ_k in Eqs. (10) and (11).

For describing the state equations of Eqs. (1) and (2) using a neueral network model, we emulated the basic structure of them. Traffic variables on the right sides of the equations were used as input signals, such as $v_{i-1}(k-1)$, $c_i(k-1)$, $v_i(k-1)$, $c_{i+1}(k-1)$, $r_i(k)$, $s_i(k)$, and $V(c_i(k))$, while variables on the left sides, such as $c_i(k)$, and $v_i(k)$, were used as output signals. Although the average speed $V(c_i(k))$ in Eq. (3) is dependent on $c_i(k)$, we treated it as an input signal because it contains some independently-determined model parameters. We produced such a neural network for each segment. Preparing a lot of target signals for extensive traffic conditions using Eqs. (1) and (2) in advance, we can adjust the synaptic weights so that the difference between the output signals and the target signals are minimized. The neural models for the state equations were used only for estimating the matrix Φ_k in Eq.(10) because Eqs. (1) and (2) could accurately estimate traffic states. Using Eq.(17), we can easily produce the components of matrix Φ_k.

For the observation equations, we adopted a similar neural structure, traffic variables on the right sides of the equations were used as input signals, such as ..., $c_i(k)$, $v_i(k)$, $c_{i+1}(k)$, $v_{i+1}(k)$,..., while variables on the left sides, such as $q_i(k)$, and $w_i(k)$. Since the real equations are unknown, we assumed that the observation variables at a point were related to the state variables of a few adjacent segments. That is, the neural network model inputs the density and the space mean speed of a few adjacent segments in both upstream and downstream. Then it outputs the flow rate and time mean speed. Since a non-linear sigmoid function is used as the activation function, it is expected that the neural network model can accurately describe the non-linear relationships between the state and observation variables. Moreover, since the model necessitates no model parameters, such as the weighting factor in Eqs. (4) and (5), it can describe the relationships just as they are. Although it is a problem in the next step, we can incorporate some specific road conditions, such as topological features which influence traffic flow dynamics, into the model because it can deal with not only quantitative but also qualitative numbers. Similarly, we adjusted the synaptic weights using the back-propagation method and obtained the components of matrix Ψ_k using Eq. (17).

3. NEURAL-KALMAN FILTER

In conventional Kalman filters, both state and the observation equations have to be analytical functions. Now, we propose an alternative algorithm, in which the equations are described by neural network models. First, based on the estimates $\hat{x}(k-1)$ at the previous time k-1, we predict the state variable $\tilde{x}(k)$ at time k using Eqs. (1) and (2). In this process, to estimate the flow rate and time mean speed at the points where traffic data were not observed, we used the neural network model of observation variables but not Eqs. (4) and (5).

Prior to obtaining the actual observed data $y(k)$, we estimate $\tilde{y}(k)$ using the neural network of observation variables. At the same time, using the neural derivative of Eq. (17), we calculate the matrices Φ_k and Ψ_k which are required for determining the Kalman gain K_k. Then we correct the predicted $\tilde{x}(k)$ and obtain the new estimates $\hat{x}(k)$. Before we go to the next time step, we reidentify the neural networks to reflect the effects of the latest observed data into both systems. That is, adding the estimates $\hat{x}(k)$ and the observed data $y(k)$ to the existing training patterns, we adjust the synaptic weights again. It should be noted here that the reidentification would not take much time because the systems are already trained for the other patterns, but not for the new ones. This feature of being able to promptly reidentify systems discriminates the neural method from conventional statistical methods because they require as much computation time as the initial identification. This ability is very convenient for such dynamic systems, in which state variables have to be estimated in real time. Since it is meaningless and time-consuming to reidentify the synaptic weights at every time step, we adjusted them every 1 minute in this analysis. By repeating those procedures, we can estimate traffic density and space mean speed in real time successively.

4. NUMERICAL EXPERIMENTS

4.1. Traffic Data

The observed data used here came from a road section, which was 5130 meters long, on the Yokohane Line of the Metropolitan Expressway in Tokyo. We used the traffic data from Oct. 28 to Nov. 1 in 1993. We divided the road section into 11 segments whose lengths Δl_i were ranged from 400 to 600 meters. Then we assumed that traffic data were observed only at four points of OP1 (entrance), OP2, OP3, and OP4 (exit), although the data were actually observed at all the boundary points. In the original Cremer model which is applicable only to a road section, where traffic data are observed at both/either entrance and/or exit of it, we

defined three subsections, which were divided at every observation point. Each subsection contained 3 or 4 segments and a checking point (CP), whose data were used to examine the effectiveness of each model. Table 1 shows the configuration of each subsection. In the neural network model, we treated the whole road section as a system, although it required a few changes in the equations of Eqs. (1), (2), (4), and (5) and the filtering matrices of Eqs. (10) and (11). That is, we assumed that there were 11 segments, four observation points and three checking points in the section. Considering the balance with the segment length[1], we simulated the traffic flows every 10 seconds.

Table 1. Configuration of subsections.

subsection	length (km)	segments	on-ramp	off-ramp
1	1.92	4	1	1
2	1.23	3	-	1
3	1.98	4	1	1

4.2. Initial Training

4.2.1. Observation Equations

First, we trained the neural network (NN) model for the observation equations. That is, we built a relationship between the state and observation variables at each observation point. Considering the fact that the observation variables at a point were related to the state variables in both the upstream and downstream segments, we supposed three types of neural structures, as shown in Fig. 3. Table 2 presents the number of neurons in each layer for each type of model.

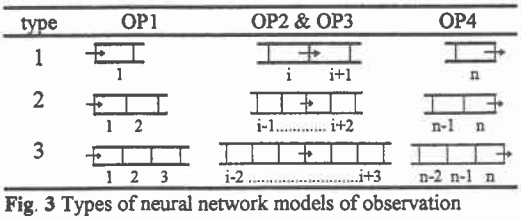

Fig. 3 Types of neural network models of observation equations.

Table 2 Structure of neural network models for observation equations.

type	number of neurons	OP1	OP2	OP3	OP4
1	input layer	2	4	4	2
	intermediate layer	2	3	3	2
	output layer	2	2	2	2
2	input layer	4	8	8	4
	intermediate layer	3	5	5	3
	output layer	2	2	2	2
3	input layer	6	12	12	6
	intermediate layer	4	7	7	4
	output layer	2	2	2	2
training data		240 (13:30-14:30, Oct. 28, 30, 31, Nov. 1)			
checking data		60 (13:30-14:30, Oct. 29)			

What we have to do next is to prepare the training and the checking data from the observed data. Picking out a time period of one hour a day from Oct. 28 to Nov. 1, we produced 240 sets of training data and 60 sets of checking data because the observed data were compiled every 1 minute. Although it may be a bit curious to select the traffic data as the checking data prior to the training, we employed the data on Oct. 29 as the checking data because the time period contained extensive traffic data. Since traffic data were actually observed at all the boundary points of the segments, we estimated the state variables of $c_i(k)$ and $v_i(k)$ using Eqs. (1) and (2), in which $q_{i-1}(k)$, $q_i(k)$, $r_i(k)$, and $s_i(k)$ are the actually observed ones. The state variables estimated here were used as the input signals to the neural networks and the observed data were used as the target signals.

The training procedure was simple. First, we assumed the initial synaptic weights. Then we gave the normalized input signals into the input layer and calculated the output signals, which corresponded to the observation variables. Then, we adjusted the synaptic weights so as to minimize the difference between the actually observed and estimated variables. In the actual computation, the input and output signals were normalized by an appropriate number. We iterated this procedure until the RMS (Root Mean Squared) error became sufficiently small for all of the training patterns. The capability of a NN model can be evaluated by the estimation precision for checking data that are not trained yet. After the completion of the training, we gave the input signals of the checking data and calculated the output signals using Eq. (16). Then we compared them with those that were actually observed. Fig. 4 depicts the average RMS errors of output signals for 60 checking patterns at four observation points for each type of NN model. We can see that the NN model of type 2 gives smaller RMS errors for all the observation points. This means that by incorporating the traffic states of the two adjacent segments in both upstream and downstream into model, we can estimate the observation variables more precisely.

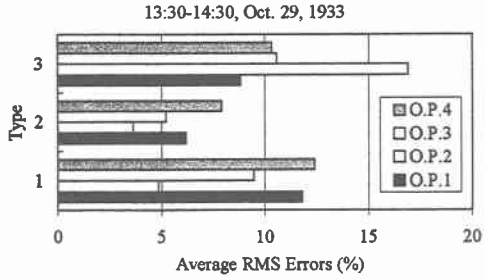

Fig. 4 Average RMS errors of neural observation systems for checking data.

For example, Fig. 5 shows the average RMS errors of flow rate at three checking points. The errors of the NN model were somewhat smaller than those of the analytical equations defined by Eqs. (4) and (5) except for the flow rate at checking points 1. Consequently, the NN model gave better results for 4 of 6 data sets. However, it is a bit early to judge the effectiveness of the neural network method at this stage because the estimate are not corrected yet.

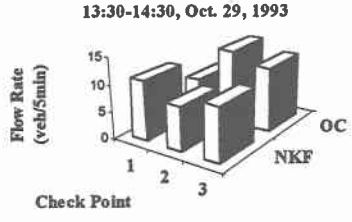

Fig. 5 Average RMS errors of neural observation systems at observation points, compared with those of original Cremer model.

4.2.2. State Equations

Next, we trained the neural network (NN) model for the state equations of Eqs. (1) and (2) to produce the derivative of the matrix Φ_k. That is, we built a relationship between the state variables at time k and those at time k+1 for each segment. As mentioned before, the neural model was designed to emulate the state equations themselves: We input $c_i(k)$, $v_i(k)$, $c_{i+1}(k)$, $v_{i-1}(k)$, $q_{i-1}(k)$, $q_i(k)$, $r_i(k)$, $s_i(k)$, and $V(c_i(k))$ and output $c_i(k+1)$ and $v_i(k+1)$. According to whether segments have an on-ramp or an off-ramp, we classified the segments into three types, as shown in Fig. 6. That is, the number of neurons in an input layer is 7 for segments that have no on- and off-ramps, and 8 for segments that have either on- or off-ramp. In this analysis, we always allocated five neurons to the intermediate layer.

For adjusting the synaptic weights of the neural networks for the state equations, picking out a time period of two hours a day, 13:00 to 15:00, from Oct. 28 to Nov. 1, we produced 480 sets of training data and 120 sets of checking data: For each time period, we simulated the traffic flow using Eqs. (1) and (2) by inputting the actually observed flow rates. Although we obtained 3600 sets of data in total because we simulated every 10 seconds for 10 hours, we thinned out the results every 1 minute. Consequently we got 600 sets of data, 480 as the training data and 120 as the checking data. For the same reason as in the previous section, we used the data on Oct. 29 as the checking data for examining the effectiveness of the NN models. The procedure of the training is exactly the same as in the previous section. By adjusting the synaptic weights using the back-propagation algorithm, we repeated the training operations until the difference between the output signals and the target signals became sufficiently small for all of the training patterns.

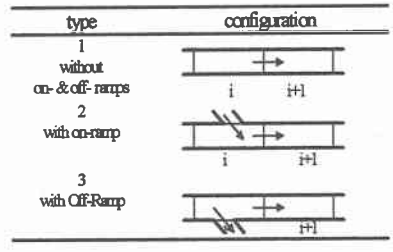

Fig. 6 Types of neural network models of state equations.

After the training was completed, we examined how precisely the NN models could estimate the state variables. That is, giving the input signals of the checking data, we calculated density and space mean speed at each segment using Eq. (15) and compared them with those that were estimated preliminarily. Fig. 7 depicts the average RMS errors of output signals for 120 sets of checking data at all the segments. We can see that the errors are small enough except for a few segments where the errors exceed 10%. The errors in Fig. 7 may be somewhat large. It should be noted that the estimates are not corrected yet by the actually observed data. Moreover, as will be shown later, since we used the neural state equations only for producing the matrix Φ_k but not for estimating the state variables, the errors did not affect the estimation precision. Also, the errors seem to be due partly to the observation errors induced into the traffic detector data themselves.

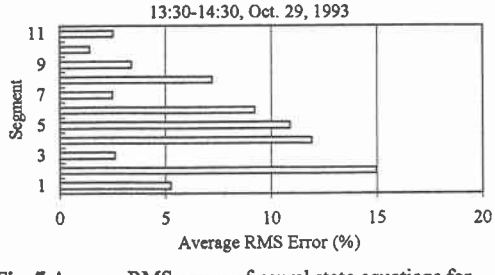

Fig. 7 Average RMS errors of neural state equations for checking data.

4.3. Estimates of Traffic States

To examine how effective the improvements we made were for estimating traffic states, we applied the Neural-Kalman filtering (NKF) models to a road section and compared them with those estimated by the original Cremer (OC) model. As the checking data, we took the time period from 13:30 to 14:30 on Oct. 29, which

included both light and heavy traffic states.

First, we estimated traffic states at the same checking point using the NKF model. That is, as explained in the previous section, we estimated the state variables using Eqs. (1) and (2), in which the neural observation model was used to estimate the observation variables at the points where traffic data were not observed. And then they were corrected by the Neural-Kalman filter. Fig. 8 presents the estimates of flow rate. It can be found that the flow rates estimated by the NKF model got very much closer to the observed ones than the OC model.

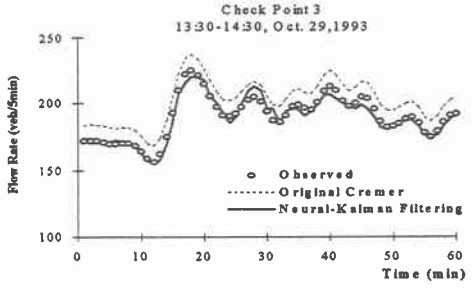

Fig. 8 Estimates of flow rate estimated by Neural-Kalman filtering (NKF) model, compared with those by original Cremer (OC) model and observed ones.

We further investigated the effectiveness of the NKF model. Fig. 9 shows the comparison of the average RMS errors of flow rate at three checking points at the same time period for the OC and NKF models. First of all, we can see that the OC model greatly improved the estimate by the filtering operations, compared with the ones in Fig. 5, which were not corrected yet. This means that the filtering operations were very effective in estimating traffic states on a freeway. Moreover, the NKF model produced much better estimates for all the data sets than the OC model. And the RMS errors are sufficiently small. We can see that the NKF model estimated the flow rate somewhat better than the OC model. That is, the NKF model gave better results for two of three data sets for both flow. Moreover, the deviation of RMS errors of the NKF model was smaller than that of the OC model.

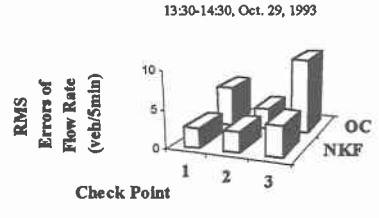

Fig. 9 Comparison of average RMS errors of flow rate evaluated by original Cremer (OC) and Neural-Kalman filtering (NKF) models.

In this way, the Neural-Kalman filtering model was effective in estimating traffic states on a freeway road section in this analysis. However, we have a lot of problems to be solved. The method is a bit time-consuming because the neural operations of Eq. (15) are carried out every time traffic variables are estimated and synaptic weights are reidentified every a few minutes. Application to a road network that contains many observation points is another problem. We confirm that we can refine the Neural- Kalman filtering model through many experiences.

5. CONCLUSIONS

A Kalman filter is a powerful tool for estimating state variables of a dynamic system. And a neural network model has some promising properties that it can excellently represent non-linear and unsteady phenomena and organize their structures flexibly. In this paper we tried to integrate a multilayer neural network model into a Kalman filtering model for estimating traffic states on a freeway road section. Intending to extend the Cremer model, we investigated how a multilayer neural network model could describe both state and observation equations. We applied the method to a road section on the Metropolitan Expressway in Tokyo and compared the results with those produced by the original Cremer model. The major findings are summarized as follows:

1) The neural observation model, which inputs density and space mean speed of two adjacent segments, was somewhat better in estimating flow rate and time mean speed than the analytical equations used in the original Cremer model.
2) The neural network models for state equations and observation equations made it possible to easily produce the derivative matrices that were needed in the Kalman filter.
3) Integrating the neural network models into a Kalman filtering technique, we proposed a procedure to estimate the traffic states on a freeway road section. Based on a feature of the neural networks of being able to promptly reidentify the systems, the method is characterized by re-training while calculating in real time.

REFERENCES

1) M. Cremer: Der Verkehrsfluβ auf Schnellstraβen, Springer-Verlag Berlin Heidelberg New York, 1979.
2) H. J. Payne: Model of Freeway Traffic and Control, Simulation Councils Proceeding Series, Vol. I, No. I, Mathematical Model of Public System, pp.51-61, 1971.
3) C.L.eondes: Advances in Control Systems, Volume 3, pp.219-292, 1966.
4) J. Dayhoff: *Neural Network Architecture*, Van Nostrand Reinhold, 1990.