Control strategies for a highway network

A joint research project of SWOV, the Technical University Delft and the Institute for Perception TNO sponsored by the Dutch Ministry of Transport and Watermanagement

PART III

W.B. Verwey & T. Heijer. *Linking Driver Visual Workload on Near-congested Highways to Inductive Loop Measurement.*

C.M. Gundy & T. Heijer. *A Traffic Model based upon Neural Networks.*

R-94-34 III
Leidschendam, 1994
SWOV Institute for Road Safety Research, The Netherlands

SWOV Institute for Road Safety Research
P.O. Box 170
2260 AD Leidschendam
The Netherlands
Telephone 31703209323
Telefax 31703201261

Contents

Linking Driver Visual Workload on Near-congested Highways to Inductive Loop Measurements

Willem B. Verwey (IZF-TNO) & Tom Heijer (SWOV)

ABSTRACT

In this report a study is presented in which two indicators for visual workload of the driver were measured in an instrumented car while driving on a four- and six-lane freeway. For one-minute periods during which the car passed inductive loops in the road surface, a parameter was computed which was assumed to correlate with traffic safety. This was the weighed localized Time-to-Collision (TTC). The research question in this study was whether visual workload of the driver, which is also assumed to be related to traffic safety, correlates with weighed localized TTC. If so, this would open the possibility to assess traffic safety from inductive loop data. In the experiment, which involved twenty subjects, half of the subjects performed a visual detection task which was secondary to driving. Performance on this task is a good indicator for visual workload. The other half of the subjects drove the route without performing a secondary task and Steering Activity Rate (SAR) was taken as indicator for visual workload. Unfortunately, due to technical problems with the registration of the inductive loop data, the data of only three subjects in the visual task condition and six in the control condition could be processed. This limited set of data was analyzed but the number of occurrences of high weighed localized TTCs was too low to arrive at firm conclusions.

SAMENVATTING

In dit rapport wordt een studie gerapporteerd waarin twee indicatoren voor de visuele werkbelasting van proefpersonen in een geïnstrumenteerde auto werden gemeten tijdens het rijden op een autosnelweg. Voor de één-minuut periodes gedurende welke de auto een inductie-lus in het wegoppervlak passeerde werd een parameter berekend waarvan aangenomen werd dat deze gecorreleerd is met de verkeersveiligheid. Dit was de 'weighed localized Time-to-Collision' (TTC). De onderzoeksvraag was of de visuele werkbelasting van de bestuurder, die ook gerelateerd wordt aan verkeersveiligheid, correleerde met de weighed localized TTC. Indien dit het geval is zou dit inhouden dat een maat voor de verkeersveiligheid uit de inductie-lus gegevens afgeleid kan worden. In het experiment, waaraan twintig proefpersonen deelnamen, voerden de helft van de proefpersonen een visuele detectietaak uit tijdens het rijden. De prestatie op deze taak is een goede indicator voor de visuele werkbelasting. De andere helft van de bestuurders reed de route zonder zo'n tweede taak en de 'Steering Activity Rate' (SAR) werd gebruikt als indicator voor visuele werkbelasting. Helaas bleek dat door technische storingen in de lus registratie, de gegevens van slechts drie proefpersonen in de visuele conditie en van zes proefpersonen in de controle conditie konden worden verwerkt. Deze beperkte set aan gegevens werd geanalyseerd maar het aantal keren dat er een hoge localized weighed TTC optrad was te gering om tot harde conclusies te komen.

INTRODUCTION

Technological advancements have opened up the way for the development of sophisticated traffic flow control systems on freeways which should not include the various disadvantages of current systems. Current systems, for example, are aimed at controlling traffic flow characteristics at an aggregated level, such as average speed, average homogeneity, and average density (Wiersma & Heijer, 1992). The implicit assumption underlying current traffic flow control systems is that traffic safety will automatically increase when traffic flows are well-controlled. However, there is only limited insight in the relation between traffic safety and processes within the traffic stream. For example, it seems plausible that the turbulence developing just before congestion, reduces traffic safety severely. If so, future traffic control systems should include possibilities to detect unsafe situations within a traffic stream. The purpose of the present experiment is to explore possibilities for monitoring traffic safety from the information gathered by inductive loop detectors in the road surface. If this were possible unsafe situations may be prevented by appropriate countermeasures such as changing the posted speed limits, ramp metering, or giving route advices at earlier exits.

One way in which safety is often assessed is by the measurement of driver workload which is known to be related to the error chance (e.g. Wickens, 1984). There are many studies showing that driving performance deteriorates under high workload levels (e.g. Brown, Tickner, & Simmonds, 1969). In itself, this does not necessarily mean that safety is also affected: Human drivers have been found to be able to drop low priority tasks when their attention is really required for driving (Rumar, 1986; Wierwille & Guttman, 1978) or they simply drive more slowly (Harms, 1991; Parkes, 1989; Verwey & Janssen, 1989). Still, the claim that safety is jeopardized in high workload situations is supported by evidence from various sources. In road traffic the most direct evidence comes from studies showing high correlations between average workload in specific situations and the reported number of accidents in those situation (Harms, 1986; Taylor, 1964). Also, accident analyses have shown that insufficient adaptation of attention to current driving demands, which is likely to occur in high workload situations, caused 30 to 50 percent of all traffic accidents (Treat, Tumbas, McDonald et al., 1979; Sussman, Bishop, Madnick, & Walter, 1985). Together, these and other findings corroborate the general claim that traffic safety decreases when workload increases in a specific situation.

Recent research has shown a dissociation of visual and cognitive workload in various road situations (Verwey, 1991, 1993b, 1993c).

For example, visual workload is high and mental load low during curve driving whereas both are high at roundabouts. Both forms of workload are probably related to traffic safety. On the one hand, mental overload increases the chance that attention is not directed at dangerous events in the traffic environment and that anticipatory actions, such as increasing headway when cars upstream are decelerating, are not carried out. On the other hand, visual overload may prevent one from simply detecting an impending danger. However, given the relatively limited mental load associated with freeway driving for experienced drivers (Verwey, 1991,1993b) it is likely that visual workload at freeways is more closely related to traffic safety than mental workload. Earlier research shows that visual load in driving can be assessed by a visual detection task which is secondary to driving (Verwey, 1991, 1993a, 1993b, 1993c).

In addition, half of the subjects served in a condition in which there was no secondary task. This control condition was used to test possibilities to derive an indication for traffic safety from the way drivers control the wheel. Verwey (1991, 1993b) measured workload and SAR in a number of traffic situations and suggested that Steering Activity Rate (SAR: Antin, Dingus, Hulse, & Wierwille, 1990; MacDonald & Hoffman, 1980) is related to visual workload. Since SAR can be unobtrusively measured and has a high temporal resolution as compared to visual workload measurement (about one measurement per s with SAR vs. one per 3 to 4 s with the secondary task), SAR may be an interesting measure for assessing visual workload in driving. However, as SAR is affected also by the load associated with the secondary task (Verwey, 1991) SAR caused by driving can be measured only in a control condition.

So, the present study investigated the level of visual workload associated with various levels of congestion and traffic density on a freeway. From the inductive loop measurements a criterion was determined which might indicate traffic safety. This was done by calculating the average Time-to-Collision (TTC: Van der Horst, 1990) in the one minute period during which the instrumented car passed an inductive loop. This period started about 30 s before the instrumented car passed the loop and ended about 30 s after passing. This criterion was called the weighed localized TTC which indicates the frequency that low and critical TTC values occur. Its derivation is described in the Method section. The aim was to find whether there is a consistent relation between weighed localized TTC and visual workload and SAR onboard cars.

METHOD

The driving task

The experimental route was part of a 17.2 km route, about half of which included driving through the Benelux tunnel when heading north and half of which when driving south through the tunnel, back to the starting location. Subjects in the secondary task condition performed a visual detection task at the entire route. Data gathered at a 5700 m stretch in the north direction, including the 1150 m long tunnel, and data gathered at a 5000 m stretch in the south direction, also including the tunnel, were used for analysis. When driving the route once, 8 inductive loops were passed in the north and 8 in the south direction. Without congestion, driving the route once took about 15 min.

The reason for taking this particular stretch of freeway in the experiment was the frequent occurrence of near-congested traffic and the high density of inductive loops. Outside the tunnel there was a loop each 300 m, inside the tunnel each 100 m.

In an attempt to evaluate performance under varying levels of the TTC criterion described below, each subject drove the route more or less continuously between 15.30h and 17.30h. Time of driving was chosen such that the last runs were expected to coincide with rush hour and, possibly, congestion whereas earlier runs included low density traffic. The experiment took place under normal weather conditions (bright, cloudy, or light rain conditions; not with fog or heavy rain).

Secondary task

Half of the subjects performed a task secondary to driving: The visual detection task required subjects to vocally indicate that they had detected a target stimulus on a dashboard-mounted display by saying the Dutch equivalent for 'yes' (i.e. 'ja'). To prevent peripheral detection of stimuli a neutral stimulus ('GG') was presented between stimuli. Targets were all two-digit numbers ranging between 19 and 100. The intervals between onset of succeeding stimuli varied randomly between 2 to 4 s. Presentation time was 750 ms. In the control condition no secondary task was performed.

Procedure

Subjects were welcomed at the Institute at about 14.00h. where they obtained an oral instruction on the aim of the study. They then drove to the experimental route which took about 60 min. During this drive, subjects in the visual detection task condition practised the secondary task for about 5 min. When they arrived they drove the experimental route once without the secondary task. At about

15.30h the experimental drives began. Driving the experimental route, which took about 15 min, was always followed by a short break of about 5 min. Then the next drive started. For safety reasons, a licensed driver-training instructor acted as experimenter who could take over with dual controls. It was explicitly stated, before and during the experiment, that the secondary task was only of minor importance and that the subjects were supposed to drive as they would normally do. They were, for example, allowed to change lanes when they felt like doing so. Hence, only when the driving task would allow them to pay attention to the secondary task they were supposed to do so.

Apparatus and data registration
The experiment was carried out in one of the institute's instrumented cars (Van der Horst & Godthelp, 1989). This car, a Volvo 240 station-wagon with dual controls, contains an IBM 486 personal computer and various possibilities for measuring driving behaviour and generating stimuli.

Secondary task stimuli were presented on a plasma display mounted high on the centre panel of the instrumented car. A black cardboard shield was placed over the display to reduce effects of glare. The eye-screen distance was about 65 cm for a normal subject of about 1.80 m length. The height of the stimuli on the screen was 1 cm (0.9° for a normal subject), the width 0.75 cm per digit (0.7°). The visual angle between the normal fixation point on the road ahead and the screen was about 27° horizontally to the right and about 20° vertically down.

During the experiment, the subjects' responses were typed into the computer by the technician in the back of the car. To prevent mishearing of the subjects' responses subjects wore a light-weight head-mounted microphone which was connected to a headphone worn by the technician. Driving performance was measured in terms of speed and position of the wheel. These were registered at a 10 Hz sample rate and were used for later derivation of Steering Action Rate (SAR). In order to synchronize workload measures with registrations at the individual inductive loops at the experimental route the on-board computer clock was synchronized with the clock of the road-side computer that registered the loop data. Together with the onboard distance registration this allowed analyses of the traffic conditions at the moment the instrumented vehicle passed the loops. Video registration of the forward scene from the car were made with the aim to have experts judge the safety of various situations. These data will be reported else where.

Data analysis

Secondary task performance was quantified for each subject in the period of passing an inductive loop. In total there were eight such loops when driving north and eight when driving south. Steering wheel position was recalculated later on into Steering Action Rate (SAR). SAR included the number of times per second that the absolute rotational velocity of the wheel crossed the five degrees per second threshold in either direction (from $\leq 5°/s$ to $> 5°/s$, or v.v.). To reduce effects of noise a transition was counted only when two succeeding samples in the data file showed a value of either below or above this threshold (i.e. when that rotational velocity occurred for more than 100 ms).

In the course of the research programme of which the present study is part (see Wiersma & Heijer, 1992), several parameters were defined that can easily be derived from raw loop data and were assumed relevant, either as a qualifier of individual behaviour or as a qualifier of the general context. The parameters measured by the inductive loops were: speed, headway and speed difference. From these parameters Time To Collision (TTC) of succeeding vehicles can be calculated when, by definition, the second vehicle is the faster one. This criterion was chosen as a measure for traffic safety because TTC is a well known indicator for safety in car-following situations (Van der Horst, 1990). TTC was calculated as follows:

$$TTC = \frac{D}{\delta V}$$

Here, D represents the distance between two succeeding vehicles in a lane and δV is the difference in speed between these vehicles. Next, the TTC value was divided into categories of increasing criticality. This as particularly useful for the present purpose because TTC is often very large indicating relatively safe and, in the present situation, not very interesting values. Categorisation largely eliminates this problem. In the present experiment individual TTCs, obtained in the 1 min period that the instrumented vehicle passed the inductive loops (see below), were averaged over lanes to obtain a single TTC value. Categorization of these TTCs was chosen on the basis of an exponential curve which emphasises criticality in the low TTC categories (Table I).

Table I An overview of the categorization of average TTCs in 1 min. periods.

category	TTC (s)
1	>60
2	10 - 60
3	3 - 10
4	1.5-3.0
5	0.6-1.5
6	0.0-0.6

One important characteristic of low TTCs in low density traffic is that these may indicate the occurrence of a lane change manoeuvre which is usually not dangerous. In high density situations this is usually not possible and low TTCs suggest critical situations. Therefore, the average occupancy for the whole carriageway was used for weighing the indicator as follows:

$$\text{indicator} = 2.8 * O_{ave} * T_{ave}$$

with O_{ave} = average occupancy over all lanes
T_{ave} = average TTC category in 1-min period (see Table I)

The factor 2.8 is choosen so, that an average occupancy of .35 produces a weighing factor of 1 : only in heavy traffic this value is sometimes exceeded.
As a last step, this weighed result was divided into 7 classes of ascending severity by simple truncation. Of course, the latter two steps are rather arbitrary, but the main point remains: it is a criterion that emphasises critical behaviour (low TTC's) and moreover increases the severity if critical behaviour takes place in denser traffic.
This criterion is referred to as weighed localized TTC.

Passing an inductive loop takes only a part of a second. However, secondary task performance, SAR and weighed localized TTC can be only computed over a relatively long period of time. So, these values were computed over a period which started before the vehicle passed the loop and which ended some time after passing. It was not clear how long this time should be for optimal correlations. When the period is too short the indicator will not be stable. When it is too long, local flow characteristics round the instrumented car have probably changed and the measurements do not indicate the relevant conditions properly.

Hence, task performance and SAR were computed for three different periods of time, 15 s before until 15 s after passing a loop, 30 s before and after, and 60 s before and after loop passing.

Subjects
Twenty subjects participated, 13 males and 7 females. They had been selected from the institute's subject pool. Subjects were not older than 45 years of age and experienced drivers in that they had driven at least about 10,000 km per year in the last five years.

RESULTS

Due to technical failure a large number of inductive loop measurements had not been logged properly. In total, only the data of three subjects in the secondary task condition and the data of six subjects in the control condition could be analyzed. In addition, one subject was excluded from the control condition because this subject had served only at low weighed localized TTC levels. The analyses involved computation of weighed localized TTCs while passing each of the 16 inductive loops on a single drive. Average secondary task performance and speed were computed for each loop passage in the secondary task condition. Likewise, average SAR and speed were computed for loop passages in the control condition. The resulting number of observations are presented in Table II. These analyses were carried out for periods of three different durations, 15 s before and after passing an inductive loop, 30 s before and after, and 60 s before and after loop passage.

Table II Number of observations as a function of weighed localized TTC and condition.

		weighed localized TTC			
condition	subjects	1	2	3	456
control	5	211	109	27	805
sec. task	3	113	52	10	200

Control condition
Average SAR and speed were computed as a function of weighed localized TTC. The results are shown in Table III. The results show a clear SAR increase and speed decrease with increasing weighed

localized TTCs. It should be noted, however, that the five observations associated with level six on the weighed localized TTC come from one subject only. A weighed localized TTC x subject ANOVA with the first four levels, including five subjects, showed no significance at the four weighed localized TTC levels [±15 s: $F(3,12)=0.8$; ±30 s: $F(3,12)=1.4$; ±60 s: $F(3,12)=1.8$, all $ps>.19$]. Table IV presents pairwise correlations between weighed localized TTC, SAR and speed on all levels.

Table III SAR (1/s) and speed (km/h) as a function of weighed localized TTC for five subjects.

		weighed localized TTC				
	period	1	2	3	4	56
SAR	±15	0.86	0.86	0.91	0.98	-
	1.11					
	±30	0.85	0.86	0.86	0.96	-
	1.01					
	±60	0.84	0.86	0.84	0.86	-
	1.06					
speed	±15	87.5	85.9	73.7	64.7	-
	37.5					
	±30	87.2	85.6	72.5	65.8	-
	36.3					
	±60	87.2	85.2	71.0	64.7	-
	37.6					

Table IV Correlations between weighed localized TTC (WLTTC), SAR and speed on data from five subjects on WLTTC level 1 to 6.

	WLTTC-SAR	WLTTC-speed	SAR-speed
±15	.12*	-.44***	.16**
±30	.10	-.46***	.15**
±60	.11*	-.50***	.15**

note: * $p<.05$, ** $p<.01$, *** $p<.001$

Secondary task condition

Despite the low number of observations in this condition with only three subjects, an attempt was made to analyze performance on the secondary task and speed as a function of weighed localized TTC. Table V shows secondary task performance and speed as function of weighed localized TTC. Weighed localized TTC x subject ANOVAs showed no significant effect of weighed localized TTC on secondary task performance [±15 s: $F(3,6)=1.7$, $p>.20$; ±30 s: $F(3,6)=13.6$, $p=.05$; ±60 s: $F(3,6)=1.5$, $p>.20$]. So, only with the ±30 s interval a marginally significant effect of weighed localized TTC on secondary task performance was found which indicated that secondary task performance increased with increasing weighed localized TTC and, hence, visual workload of driving decreased with increasing weighed localized TTC. This was unexpected. Table VI presents correlations between weighed localized TTC, secondary task performance and speed. It confirms the unexpected secondary task performance increase as WLTTC increased. Table VI also shows that speed did not correlate with weighed localized TTC and that performance increased as speed decreased, which was expected.

Table V Secondary task performance (% correct) and speed (km/h) as a function of weighed localized TTC for three subjects.

		weighed localized TTC				
	period	1	2	3	4	56
perf.	±15	69.6	81.0	71.7	83.3	--
	±30	70.0	79.3	81.6	85.7	--
	±60	70.5	78.2	71.5	80.5	--
speed	±15	83.6	83.8	78.5	88.6	--
	±30	83.3	83.6	79.3	90.6	--
	±60	82.4	83.7	79.1	88.7	--

Table VI Correlations between weighed localized TTC (WLTTC), secondary task performance and speed on data from three subjects on WLTCC level 1 to 4.

	WLTTC-perf.	WLTTC-speed	perf.-speed
±15	.16*	-.03	-.25***
±30	.21**	-.01	-.23**
±60	.12	.02	-.13

note: * $p<.05$, ** $p<.01$, *** $p<.001$

DISCUSSION

The aim of this study was to find whether visual workload increases as a function of a criterion that might indicate unsafety, the weighed localized TTC. Unfortunately, more than half of the inductive loop data was lost because of damaged registration tapes. An attempt to analyze the remaining data was further troubled by the fact that there were virtually no observations at the higher levels of weighed localized TTC. On the six point scale of the weighed localized TTC, there were only 15 observations at levels 4 to 6.

In the control condition, where data from six out of ten subjects was available, Steering Activity Rate (SAR) served as indicator for visual workload. SAR increased with weighed localized TTC but in an ANOVA this did not reach significance. The correlation was highest with a period of 15 s before and after passing an inductive loop but remained quite modest. As expected, average speed decreased with increasing weighed localized TTC. This suggests that the currently used categorization by weighed localized TTC may be a good indicator for task load. However, these correlations are to some extent caused by the data from only one subject who drove in category 6. Therefore the data at level 6 were not included in the ANOVA which showed no significant effect of weighed localized TTC on SAR.

Unfortunately, in the visual secondary task condition the data from only three subjects were available. These data showed no significant relation between speed and weighed localized TTC. This was unexpected. Bearing in mind that speed did decrease in the control condition this suggests that the speed increase in the secondary task condition was due to the limited number of observations. No correlation was found in the expected direction which could be attributed to increasing visual load with increasing weighed localized TTC. So, the very limited number of observations in this condition does not justify any conclusion on the relation between weighed localized TTC and visual workload.

Together, the limited number of observations precludes any firm conclusions. The data do not show whether visual workload increases with weighed localized TTC and even suggest a reversed relation which can be attributed to the limited number of observations. As expected, SAR appeared to increase with increasing weighed localized TTC but the number of observations in category 5 and 6 was too small to allow proper testing. The data do not allow conclusions for the optimal period over which SAR and performance on the secondary task should be assessed.

Given that with six out of the original ten subjects in the control condition only 13 observations in categories 4-6 were found, this suggests that, even without data loss, the number of observations in these categories would have been limited. That this was caused by the rare occurrence of near-congested traffic and not because weighed localized TTC was invalid, is indicated by the infrequent observations of low driving speeds and the subjective estimates of congestion by the experimenter. This suggests that either the time of experimenting, which took largely place in June, or the route, were not appropriate for this experiment. Possibly, derivation of the instrumented car's own TTC from the video tapes that were made during the present experiment may be used for finding a relation between SAR and task performance with a TTC related measure. Yet, the rare occurrence of congestion makes the usefulness of such an enterprise doubtful.

For future research the rare occurrence of critical situations as indicated by weighed localized TTC implies that one should better investigate the relation between visual workload and conditions in the traffic stream in a driving simulator first. This gives proper experimental control over the conditions in the traffic stream by making traffic stream conditions an independent rather than a dependent variable. This also reduces the problem that local TTC values in the area round the subject's car are hard to relate to the subject's workload because these TTCs are registered by a loop at a fixed location whereas workload is registered in the moving car. Instead, in a simulator one can precisely and continuously measure local TTCs around the subjects' car. The only precondition is that proper modelling will yield processes in the traffic stream that are comparable to those in the real world. Data required for such modelling can be easily obtained from the loop data assessed in the present study. If and when a valid criterion has been found for driver workload that can be extracted from loop data, such as the presently used weighed localized TTC, the present study may be repeated in actual freeway driving.

REFERENCES

Antin, J.F., Dingus, T.A., Hulse, M.C., & Wierwille, W.W. (1990). An evaluation of the effectiveness and efficiency of an automobile moving-map navigational display. Int. Journal of Man-Machine studies, 33, 581-594.

Brown, I.D., Tickner, A.H., & Simmonds, D.C.V. (1969). Interference between concurrent tasks of driving and telephoning. Journal of Applied Psychology, 53, 419-424.

Harms, L. (1986). Drivers' attentional responses to environmental variations: A dual-task real traffic study. In A.G. Gale, M.H. Freeman, C.M. Haslegrave, P. Smith, S.P. Taylor (eds), Vision in Vehicles (pp. 131-138). Amsterdam: North-Holland.

Harms, L. (1991). Experimental studies of variations in cognitive load and driving speed in traffic and in driving simulation. In A.G. Gale, I.D. Brown, C.M. Haslegrave, I. Moorhead, & S. Taylor (eds), Vision in Vehicles III (pp. 71-78). Amsterdam: North-Holland.

Horst, A.R.A. van der (1990). A time-based analysis of road user behaviour in normal and critical encounters. Unpublished doctoral dissertation. Soesterberg, the Netherlands: TNO Institute for Perception/Human Factors.

Horst, R. Van der, & Godthelp, H. (1989). Measuring road user behavior with an instrumented car and outside-the-vehicle video observation technique. Transportation Research Record, 1213, 72-81.

MacDonald, W.A., & Hoffman, E.R. (1980). Review of relationships between steering wheel reversal rate and driving task demand. Human Factors, 22, 733-739.

Parkes, A. (1989). Changes in driver behavior due to two modes of route guidance information presentation: A multi-level approach. Loughborough: HUSAT Research Centre.

Rumar, K. (1986). In-vehicle information systems. Int. J. of Vehicle Design, 9, 557-564.

Sussman, E.D., Bishop, H., Madnick, B., & Walter, R. (1985). Driver inattention and highway safety. Transportation Research Record, 1047, 40-48.

Taylor, D.H. (1964). Drivers' galvanic skin response and the risk of accident. Ergonomics, 7, 439-451.

Treat, J.R., Tumbas, N.S., McDonald S.T. et al. (1979). Tri-level study of the causes of traffic accidents: Final report. U.S. DOT HS-805-086, (NTIS PB 80-121064).

Verwey, W.B. (1991). Adaptive interfaces based on driver resource demands. In F. Queinnec and F Daniellou (Eds.), Designing for Everyone and Everybody. Proceedings of the 11th Congress of the International Ergonomics Association (Vol. II, pp. 1541-1544). London: Taylor and Francis.

Verwey, W.B. (1993a). How can we prevent overload of the driver? In A.M. Parkes and S. Franzen (eds), Driving future vehicles (pp. 235-244). London: Taylor & Francis.

Verwey, W.B. (1993b). Driver workload as a function of road situation, age, traffic density, and route familiarity. (Report IZF 1993 C-11). Soesterberg, the Netherlands: TNO Institute for Human Factors.

Verwey, W.B. (1993c). First test of man-machine interface adaptation to driving situation. (Report IZF 1993 C-44). Soesterberg, the Netherlands: TNO Institute for Human Factors.

Verwey, W.B., & Janssen, W.H. (1989). Route following and driver performance with in-car route guidance systems. Proceedings of the INRETS/VTI congress "Road safety in Europe" (pp. 81-97). Gothenburg, Sweden: VTI.

Wickens, C.D. (1984). Engineering Psychology and Human Performance. Columbus: Merrill.

Wiersma, J.W.F., & Heijer, T. (1993). Dynamic aspects of motorway traffic safety. In J.L. de Kroes & J.A. Stoop, Proceedings of the First World Congress on Safety of Transportation (pp. 76-86). Delft, The Netherlands: Delft University Press.

Wierwille, W.W., & Guttman, J.C. (1978). Comparison of primary and secondary task measures as a function of simulated vehicle dynamics and driving conditions. Human Factors, 20, 233-244.

A Traffic Model based upon Neural Networks

C.M. Gundy & T. Heijer
Leidschendam, 1994
SWOV Institute for Road Safety Research, The Netherlands

SWOV Institute for Road Safety Research
P.O. Box 170
2260 AD Leidschendam
The Netherlands
Telephone 31703209323
Telefax 31703201261

1.0 INTRODUCTION

The research reported here is a part of a much larger project aimed at the development of a comprehensive control strategy for traffic on a freeway network. This larger project roughly consists of three parts:

a) development of an "assignment model" which predicts traffic flow intensity in all parts of the network as a function of time and place and 30-60 minutes ahead in time : this model will be used as a basis for redirection, general traffic information and adaptation of lower level traffic controllers.

b) development of an adaptive control strategy for speed and density in the network: to this end the network is divided into small "subsystems" (limited stretches of road with 1-6 on- and off-ramps) in which separate controllers operate; these controllers are coordinated by intermediate level coordinators to ensure consistency within the network. Furthermore, these controllers adapt their set-points according to predictions generated by the assignment model of part a).

c) the development of a "safety module". This module will be present in each subsystem and have several functions:

- continuous evaluation of the safety of the local traffic stream; the results will also be used to modify the settings of the control-system as developed in b),
- implementation of special measures: often the more generalized control of average speed and density do not suffice to stabilize certain behaviour and more dedicated feedback will be called for: these modules will also contain a repertoire of these messages/measures,

and

- incident detection and handling.

Artificial neural networks seem to be a promising, new tool for application in diverse aspects of these three of these parts. For this reason, the following section of this report concerns a general orientation on artificial neural networks, intended for those readers not yet familiar with this technology.

The second main section concerns a specific application of neural networks, which addresses a portion of the research involved with part b) mentioned above. Namely, having developed a model for an adaptive control strategy, one needs to test it before actual implementation. Due to the associated costs, one would like to test it in the laboratory before moving to real-world tests. One form of laboratory testing would involve the use of simulations of traffic flow patterns. We therefore need a computer model to simulate traffic flow characteristics. There are many ways to achieve this, some being simple, others being more complex and time consuming.

Neural network models were chosen to implement such a simulation, and for a number of reasons:

In the first place, while rather time-consuming in implementation, their actual execution (one the model is fit) is relatively fast. Secondly, it was

decided not to use a linear model for the simulated traffic model, because it was deemed likely that a linear model would be involved in the adaptive control strategy. This would result in a linear model controlling a linear model, which would not seem to be a sufficiently robust test of the adaptive control model. Third of all, it is well known that neural networks are capable of modelling non-linear relations (which do not have to be specified before-hand). Thus it seemed possible that a neural network could actually be more robust model of the traffic process itself.

Therefore, the second main section of this report considers the results of a neural network modelling of traffic flow characteristics.

2.0 GENERAL ORIENTATION ON ARTIFICIAL NEURAL NETWORKS[1]

Artificial Neural Networks, to distinguish them from the real thing, are interesting mathematical models which have created a great deal of enthusiasm (some say even a "paradigm shift") in recent years.

Their abilities are extensive. They are able, for example, to learn arbitrarily complex input-output relations, to adapt to changing environments, to abstract classes of stimuli, to make 'best guesses' about missing data, and to find 'reasonable' solutions to complex, non-linear problems with a large number of constraints. They are able to generalize to new situations on the basis of known ones, are tolerant of noise and ambiguity, and demonstrate 'graceful' degradation, instead of catastrophic failure, when damaged.

Artificial neural networks (hereafter ANNs) can be viewed from a number of perspectives.
They can, of course, be seen as interesting objects in themselves, lying somewhere between (applied) mathematics and (theoretical) engineering. One can study the properties of certain networks, or build a network to demonstrate that it can do something interesting.
ANNs can also be viewed as models of something else, primarily useful in understanding how a certain phenomenon works. Academic psychology is, for example, quite busy building ANN models of memory, learning, perception, and attention. While there is still a great deal of controversy (and confusion), one can safely say that ANN modelling has had a major impact on contemporary thinking about psychological functions.
A third, and more mundane, viewpoint is that ANNs are also simply mathematical tools that one can use to get a job done. The only question in this case is whether the tool is appropriate for the task. For the rest, one could view the tool as a black box. (One does not have to understand physics in order to use a television set.)

In the present case, we are primarily interested in this third, practical, viewpoint. We are interested here in predicting characteristics of traffic flow on highways. We do not want to imply that somehow traffic flows

---

[1]This chapter is intended to be an introduction, limited to the knowledge specifically needed for understanding the present application. As such, it is neither complete nor are the ideas presented here original for the present authors. The interested reader is referred to the sources mentioned in the bibliography.

could be better understood by studying the workings of brains, artificial or otherwise.

In the following sections, we will first describe the components and characteristics of ANNs in general. Secondly, we will briefly consider the historical development of a specific class of ANNs, namely 'feed forward' models. In a third, and final section, we will consider a specific algorithm, back propagation', and enumerate its' potential strengths and weaknesses for the present application.

2.1 What is a Artificial Neural Network?

There are probably (tens of) dozens of classes of ANNs, each class possessing one or more variants. More are being invented each day. Consider the following (incomplete) list:

-(Multiple) ADaptive LInear NEuron (MADALINE);
-Adaptive Resonance Theory (ART);
-Back-Propagation (Backprop);
-Bi-directional Associative Memory (BAM);
-Boltzmann Machine;
-Brain-State-in-a-Box (BSB);
-Cascade Correlation;
-Categorization And Learning Module (CALM);
-Counter-Propagation;
-Digital Neural Network Architecture (DNNA);
-Directed Random Search (DRS);
-Functional Link Networks (FLN);
-Hamming Networks;
-Hopfield Networks;
-Learning Vector Quantization (LVQ);
-Perceptrons;
-Probabilistic Neural Networks (PNN);
-Recirculation Networks;
-Self Organizing Maps (SOM);
-Spatio-Temporal pattern Recognition.

Each class probably has hundreds or thousands of actual implementations, each of which is more or less unique. In fact, each implementation can have multiple solutions.

What do all of these things have in common? What are the basic ANN building blocks?

First of all, every ANN that ever existed is built up out of nodes and connections. We can roughly view each node as representing a variable. Nodes associated with input are called 'input' nodes, those associated with output are called 'output' nodes, and those associated with nothing in particular are called 'hidden' nodes. We could perhaps view a hidden node as representing the analogue of a latent variable or a principal component. Each node has, at a particular moment in time, an activation, which we will return to in a moment.
Nodes are attached to each other by means of connections. (Hence the

term connectionism.) Connections between two nodes may be uni-, bi- or non-directional, or even be non-existent. Each existing connection has a weight, or value, which may be fixed, variable, or somehow limited in the values that it can take. For example, bi-directional connections may have equal weights for each direction, may be restricted to be greater than or equal to zero, or may have a freely varying weight for each direction.

Each node acquires an activation, which is either directly given by the user (if the node is an input node), or is some function of the activations of the nodes connected to it, and the weights associated with those connections. This function is called the transfer function.
A very simple example is the weighted sum, or:

$$a_i = \Sigma_j a_j w_{ij}.$$

In words, the activation of node i is equal to the sum of the activations of nodes j, weighted by their connection to node i.

More commonly, however, a non-linear transformation, such as the logistic or hyperbolic tangent, is first applied to this weighted sum. Of course, many different transfer functions can be, and are, applied.

Once the activation of node i is calculated, then this value can then be passed on to the other nodes that node i is connected to.[2]

Activations can change very rapidly, either on demand by the user, or as a function of other rapidly changing activations. Weights, on the other hand, either do not change at all or change more slowly. Fixed, or 'hard-wired', weights do not learn. This is useful in some applications, such as constraint satisfaction, yet the actual choice of these fixed weights determines whether the application will be useful.
(Slowly) changing weights, however, can represent learning. (In this way, one can say that the knowledge is in the weights.) The method used for changing weights, or the learning rule, is clearly crucial for determining what a network can or cannot do. An example of a simple learning rule is that when two nodes have the same activation the corresponding weight should be increased; when they differ the corresponding weight should be decreased. Other learning rules, error-based learning and competitive learning, are more complex.

Finally, an ANN has an overall structure, or architecture. Once we specify how many nodes there are, what each node's transfer function is (they don't all have to be the same), what connections exist between the nodes (and the world), and which restrictions and which learning rule (if any) apply to each connection, then we have an instantiation of one of the above mentioned classes of neural networks. If we furthermore specify a

---

[2]The astute reader may note that this can lead to certain problems. Namely, if nodes i and j and connected, and we can only update the activation of one node at a time, because our computer is a serial machine, then which node is updated first? There are a number of ways of solving or avoiding this problem, yet they will not concern us further here. This problem, however, doesn't arise in feed-forward networks, where activation is propagated from input nodes forward towards hidden nodes and then onto the output nodes.

number of parameters, such as how fast the network should learn, then we only have to connect it to the world (by making input available).

Summarizing, an ANN can be viewed as a complex (non-linear) system of nodes (or variables) and (numerical) relations between those nodes. By means of a set of updating rules, the activations of the nodes and/or weights are altered so as to minimize some function. This function may be explicit: for example, it could be the sum of squares of differences between actual and desired output. It may also be quite implicit, reflecting only some researcher's notion as to how he wants his system to behave.
All neural networks are basically variations on this theme.

2.2 A Short History

The origin of neural network research has two main roots, dating from the 1940's. One root originated from the work of McCulloch and Pitts, two mathematicians who showed that neural nets could be programmed to perform any arbitrary logical function. (This work later founded the basis for the modern computer.) Unfortunately, these networks did not learn: a human had to program them. In other words, they showed that networks useful could be built, they only failed to provide an algorithm for doing so.
The other root originated from the work of D. Hebb, a psychologist who was interested in models of how real neurons interacted and could learn. Many of Hebb's ideas, such as correlational learning (see previous section), can still be found today in current research.

Between the mid 1950's and 1970's, neural net research can be generally divided into three major classes reflecting the type of learning rules used, and the purposes for which they are well adapted. One class emphasizes competitive learning which attempt to divide the input space into sub-regions. This class is well-suited for e.g. classification tasks where there is no a priori sub-divisions. Another general class emphasizes correlation learning which attempts to abstract statistical generalities from input space. This class is well-suited for cleaning up noisy data or making guesses about missing data. A third class emphasizes learning by comparing its output signal with an externally defined criterion, and computing the error. This class is particulary useful when we have a set of predictors and a set of to-be-predicted variables, but we don't know to relate the one to the other.

This last problem is essentially the one that we have at hand here: we want to predict traffic flow characteristics at a certain time and place, based on data from another time and place.
For this reason, we will henceforth neglect developments in the first two classes just mentioned[3].

A major breakthrough in learning neural networks was made by Rosenblatt in the late 1950's with the development of the Perceptron[4]. This model was originally developed with optical pattern recognition in mind. This network was noise tolerant, could make limited generalizations, and was capable of re-learning even after part of the network was ablated.

The idea was that an object (in the visual field) activated a pattern in an array of sensors. Each sensor was randomly connected to a number of associator units, or 'feature demons'. If the weighted sum of input to a specific demon exceeded a certain threshold, then the demon fired and passed its activation to a final layer of perceptrons. The perceptron layer

[3]Actually, there is a fourth class of models, derived from statistical mechanics. This models are, however, of even later origin, and not directly relevant for our present problem.

[4]Widrow and Hoff developed similar models in the early 1960's.

compared the weighted sum of its' inputs to an (externally supplied) vector of desired output. The weights connecting the demon with the perceptron layers were modifiable, i.e., they could learn, and then in the following manner.
If the output of a perceptron and the desired output were identical, then the weights were left unaltered. If the perceptron output was larger than the desired output then the weights on all active input lines should be decreased. Otherwise, they should be decreased. The amount of decrease or increase is based on the size of the difference between actual and desired output: roughly, small errors should lead to small changes and vice versa.
Rosenblatt demonstrated that for a given set of inputs and desired outputs, then the perceptron would learn the correct set of weights for accomplishing this, if that sets of weights existed.

However, that is the problem! Namely, Minsky and Papert demonstrated in the late 1960's that no such set of weights existed for a single layer of perceptrons for a important group of problems. This set of problems, which we will return to later, did admit to a solution in a system with several layers of perceptrons, but no one knew how to train the intermediate layers. These intermediate layers had no desired output to be compared to, and thus, no error.

Minsky and Papert's work convinced a lot of people that perceptrons were only useful for simple, uninteresting problems, and the whole field of research languished. One could say that the (somewhat premature) death knell of the perceptron diverted interest towards the symbolic information processing paradigm, which still dominates artificial intelligence and cognitive psychology. However, a number of researchers, e.g., Grossberg, Anderson, and Kohonen, did continue their work through the 1970's, and provided a basis for present day developments.

2.3 Linear Separability

As mentioned in the previous section, there is a class of problem that a one-layer perceptron cannot solve; namely, problems whose solution regions are not linearly separable.

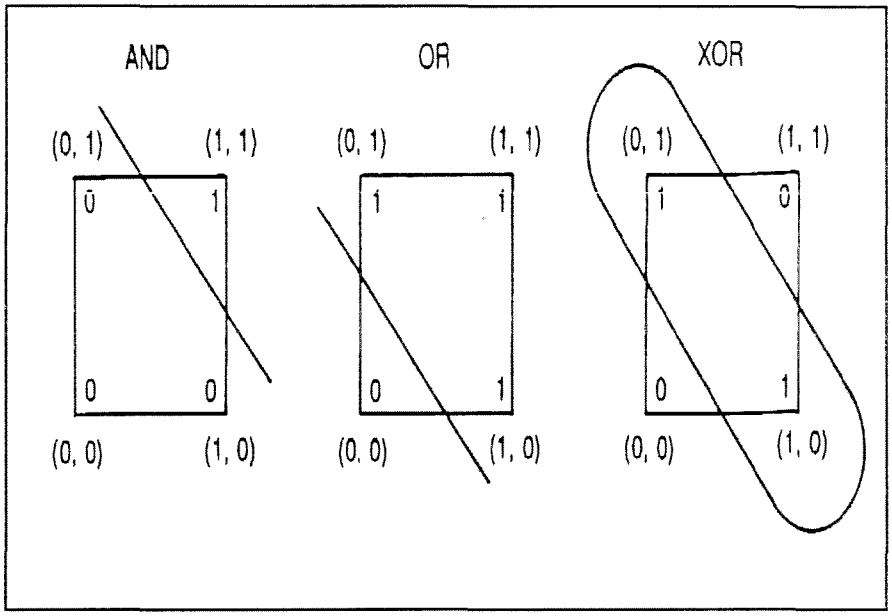
To understand this, consider the following example.

Skipping the input buffer level, assume that we have two feature 'demons', both of which can assume one of two values, either 'on' or 'off'. These two demons are linked to a single, binary-valued perceptron. These links can take on any value. The question is then: which input-output relations is the perceptron capable of representing?

The answer is simple. Since the input to the perceptron is the weighted sum of two input variables, which is equivalent to drawing a line in the two-dimensional input space, the perceptron can only learn distinctions which can be made by drawing a line through that input space.
This result is generalizable to situations with more input dimensions, in which we can speak of planes or even hyper-planes, and situations with more output dimensions, in which we can speak of multiple planes or multiple hyper-planes. And, of course, this generalization is not limited to

binary input/output variables.

Are there problems whose solution space is not linearly separable? Consider Figure 1, which is a geometric representation of the four possible combinations of values that two binary variable can take.



Afbeelding 1

If we consider the logical AND function, for which the output is equal to one if and only if both input variables are equal to one, then it is easy to draw a line separating the point '11' from the other three points ('00', '01', and '10'). This is also possible for other functions, such as the logical OR. If however, we consider the logical function EXCLUSIVE OR, for which the output is equal to one if and only if one and only one of the input variables is on, then we cannot draw a straight line separating '10' and '01' from '11' and '00'. This problem is therefore not linearly separable.

There are generally two ways of solving this (and other similar) problems.

The first solution is to pre-process the input data so that possible combinations of input data values are already included in the input layer. Some researchers advocate this approach, primarily due to its' learning speed. However, the number of combinations of variables grows much faster than the number of variables, and this approach can easily result in an immense amount of computation. Secondly it transfer the burden of learning the appropriate combinations from the computer algorithm to the insight (and good luck) of the researcher.

The second approach is to create multi-level networks, with hidden or auxiliary variables, and let the network find the appropriate transformations. However, as we mentioned above, no one knew to train multi-layer networks. This problem was independently solved a number of times, yet McClelland and Rumelhart, two psychologists, popularized the technique during the mid 1980's. Their algorithm was called Back Propagation or Back Prop.

2.4 Back Propagation

Back Propagation is a multi-purpose algorithm for training multi-layer networks. It has been shown that Back Prop is capable of fitting any arbitrary input-output relation, if the network has enough hidden nodes and enough hidden layers. This statement reveals three types of problems.

Concerning the number of layers, it has been shown that one layer (of trainable weights) is enough for any linear separable problem, two layers are enough for any problem which requires dividing the into space input convex regions, and three layers are enough for any other problem.

Concerning capabilities, back propagation is capable of learning an adequate solution, but this is no guarantee that it will learn an adequate solution. Back propagation uses an approximation of gradient descent, and may become trapped in a local minimum.
The algorithm attempts to find a combination of weights such that the sum of the squares of error (the difference between desired and actual output) is as small as possible. Each combination of weight values has, therefore, an associated sum of squares of error. This can be visualized by assuming that each weight corresponds to a combination of longitude and latitude, and the sum of squares of error correspond to elevation. (With more weights, the problem becomes multi-dimensional, and a simple visualization is not possible.) The algorithm is randomly initialized, or blindly parachuted onto the error landscape, so to speak. Its' task is to find the deepest valley (or smallest sum of squares of error) without the aid of a map and in a dense fog, a rather daunting task. It proceeds by determining the direction with the steepest decline, and takes a step in that direction. It is clear that this approach does not guarantee finding the deepest valley, only a valley. Another 'parachute drop' (c.q. random initialization) could lead to better results.
McClelland and Rumelhart, aware of the potential problem, remark that, with enough 'hidden' nodes, this is rarely a severe practical problem.

Concerning the number of hidden nodes needed, the choice is more of an art than a science. One could conceivably have more hidden nodes than input/output pairs, in which finding a 'good' solution is rather trivial. (Namely, there are more free parameters than degrees of freedom.) Too few nodes and the errors will be unacceptably large.
This issue is especially thorny, and simultaneously reflects one of the strengths and weaknesses of the procedure. Namely, the representation of the problem space is distributed over the hidden nodes, i.e., a hidden node simultaneously reflects multiple aspects of the same problem. In addition, there is no a priori reason which prevents two nodes from doing exactly the same thing. Fully connected back prop networks are therefore neither easily interpretable nor necessarily parsimonious. On the other hand, this same characteristic enables them to be noise tolerant and to suffer ablation with 'graceful degradation'.
We can therefore only offer the advice to use as few hidden nodes as one can get away with for a specific problem.

2.4.1 The Algorithm[5]

We have already mentioned that back propagation uses a form of gradient descent for training multi-level networks, previously a problem with no known solution. A crucial insight into solving this problem was that each node has to perform a non-linear transform on its input (the weighted sum of outputs of nodes in lower layers). This is because a linear transform of a linear transform remains linear, and the linear separability barrier cannot be breached. Secondly, this non-linear transform had to be (easily) differentiable. A suitable candidate for the position seemed to be the logistic (or s-curved) function.

In the following we will first derive the learning rule, and then step through the algorithm.

First, define the following:

$$net_{pj} \doteq \sum_i w_{ji} o_{pi} \qquad (1a;1b)$$

$$o_{pj} \doteq f(net_{pj})$$

where o_{pi} is the output of node i for pattern p;
w_{ji} is the weight from node i to node j;
net_{pj} is the net activation of node j for pattern p;
and f refers to some function.

If f refers to the logistic function, then its' derivative f' is:

$$f' = f(1-f) \qquad (2)$$

Furthermore, define the following:

$$\delta_{pj} \doteq -\frac{\partial E_p}{\partial net_{pj}} \qquad (3)$$

Now, suppose that we want to minimize the (sum of squares of) error for pattern p, or:

$$E_p = 1/2 \sum_j (t_{pj} - o_{pj})^2 \qquad (4)$$

where E_p refers to the sum of squares of error for pattern p;
t_{pj} refers to the desired, or target, output, with the subscripts as above.

(Note: We have not included any weights in this last equation, which is equivalent to assuming that all output variables are equally important. This does not always have to be the case, and weights may be included if

[5]The reader may skip this section without loss of continuity.

necessary.)

In order to alter weights so as to minimize the error, we have to differentiate E with respect to those weights. Using the chain rule:

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\partial net_{pj}} \frac{\partial net_{pj}}{\partial w_{ji}} \tag{5}$$

The first factor on the right hand side has already been defined above (3). The second factor is:

$$\frac{\partial net_{pj}}{\partial w_{ji}} = \frac{\partial \sum_k w_{jk} o_{pk}}{\partial w_{ji}} = o_{pi} \tag{6}$$

Therefore,

$$-\frac{\partial E_p}{\partial w_{ji}} = \delta_{pj} o_{pi} \tag{7}$$

and our weight changes should be proportional to this value, or:

$$\Delta w_{ji} = \alpha \delta_{pj} o_{pi} \tag{8}$$

which is called the delta rule. It is the value of δ that we have to derive.

There are two cases: for output units, and for hidden units.

In the first case we will apply the chain rule to the definition (3) above:

$$-\frac{\partial E_p}{\partial net_{pj}} = -\frac{\partial E_p}{\partial o_{pj}} \frac{\partial o_{pj}}{\partial net_{pj}} \tag{9}$$

From the definition above (1), we also see that:

$$\frac{\partial o_{pj}}{\partial net_{pj}} = f' (net_{pj}) \tag{10}$$

Differentiating our error function (4), we also see that:

$$\frac{\partial E_p}{\partial o_{pj}} = (t_{pj} - o_{pj}) \tag{11}$$

Substituting into (9), we find:

$$\delta_{pj} = (t_{pj} - o_{pj})\, f'(net_{pj}) \tag{12}$$

for output unit j.

In the second case, we are concerned with the hidden unit, again with a subscript j. Here we have to apply the chain rule again:

$$\begin{aligned} \frac{\partial E_p}{\partial o_{pj}} &= \sum_k \frac{\partial E_p}{\partial net_{pk}} \frac{\partial net_{pk}}{\partial o_{pj}} \\ &= \sum_k \frac{\partial E_p}{\partial net_{pk}} \frac{\partial \sum_i w_{ki}\, o_{pi}}{\partial o_{pj}} \\ &= \sum_k \frac{\partial E_p}{\partial net_{pk}} w_{kj} \\ &= -\sum_k \delta_{pk}\, w_{kj} \end{aligned} \tag{13}$$

Substituting back in again,

$$\delta_{pj} = f_j'(net_{pj}) \sum_k \delta_{pk} w_{kj} \tag{14}$$

when the subscript j refers to a hidden unit.

The algorithm works as follows:

0) Select learning parameters and initialize all weights.
1) Present a (pattern) vector to the input layer.
2) For each hidden layer node, weight each incoming input value and calculate the sum.
3) For each hidden layer node, transform this weighted sum by a logistic function, and output this value to the next layer.
4) Repeat 2) and 3) for each hidden layer.
5) For each output layer node, calculate the weighted sum of the incoming input values, and apply a logistic transform. This transformed valued is the actual output.
6) Compute the difference between the actual output and the desired output for each node, and multiply this by the derivative.
7) Update the last hidden layer to output layer weights, on the basis of the values calculated in 6).
8) Take the values calculated in 6), multiply them by the weights updated in 7), and take the sum. Multiply this sum by the derivative for each hidden layer node.
9) Take the value calculated in 8), and update the weights to this hidden layer on the basis of this.
10) Repeat 8) and 9) for each additional hidden layer.

11) Go to 1, and repeat until all patterns have been presented.
12) Calculate the total sum of squares of error (between desired and actual output). If this value is 'sufficiently' small, then stop. Otherwise, go to 1.

It should be noted that there are many variations on this basic algorithm (and the accompanying mathematics). For example:

-Patterns may be randomly sampled (without replacement) for each pass through the data set or they may presented in the same sequential order.
-Weight updates may be done after each pattern (as above), or they may be done after the entire data set has been presented. This last 'batch' method usually gives smoother convergence.
-Functions (with derivatives) other than the logistic one may be used.
-Different learning parameters may be used for each layer. In addition, learning schedules may be selected, so that the rate of learning changes as a function of the number of examples presented.
-The basic algorithm may be augmented with additional features intended to speed up learning. These additional features, for example, may take the rate of weight changes into consideration in order to adjust learning parameters for individual nodes.
etc.

2.4.2 Limitations of Back Propagation

As we have seen in previous section, back propagation is a powerful technique for mapping complex input-output relations. A major theoretical limitation is that it can become trapped in local minima. Another difficulty is that it may require a great deal of work to understand what a network solution is actually doing, something which may require the use of multivariate techniques.

In this section we will discuss a number of other practical problems.

Learning is slow and uncertain. For example, the solution of the exclusive OR problem mentioned above is well understood. However, the standard algorithm may require several hundred passes through the data set before any real learning becomes apparent. If one quits too soon, then the incorrect conclusions may be drawn. This problem is exacerbated with large data sets, with many thousand of observations and (tens of) dozens of variables, especially when we don't know that a better solution is possible. One could conceivably spend months of fruitless computing.
Special computational speed-up features alleviate this problem somewhat, but do not solve it[6].
In addition, many researchers have found that alternative architectures and/or learning schedules (or whatever) sometimes achieve better results. (One rule of thumb is that if learning has reached a plateau, then decrease the learning parameters.) Systematically searching the entire parameter space is computationally prohibitive (even though some researchers utilize genetic algorithms towards this end). We could conclude that at least some aspects of fitting neural networks is more of an art than a science.

---

[6]It should be noted, however, that once the network has learned satisfactorily, and only prediction is required, computation is very fast.

In addition to the gnawing uncertainty just mentioned, back propagation exhibits a somewhat more fundamental difficulty: it can not only learn, but it can also forget, and even 'catastrophically' so. For example, if we first train a network to perfection on data set A, then train it on data set B, and then test what it knows about data set A, then we find that there is not only evidence of retroactive interference, but that it seems to have forgotten almost everything about the first data set. Adding additional nodes, or implementing exotic learning parameter schedules do not alleviate the problem. A number of 'tricks' have been discovered, yet the sequencing of training examples is crucial, especially if the world is non-stationary.

This last point cannot be emphasized enough. Everyone realizes that it would be foolhardy to assume that a network trained on a congested, urban highway could be directly applied to a sparsely populated, rural section of the same road. (We would, however, be quite pleased if this turned out to be the case.) It is not as obvious, but nevertheless equally dangerous to assume that a back propagation network trained on a given situation will always retain its' knowledge of that situation after further training on other situations.

These last two points, slow learning and catastrophic forgetting, make it imperative that training be carefully supervised and continuously evaluated. Although this is not the point of the present study, carefully formulated guidelines have to be set up.

3.0 THE APPLICATION OF A NEURAL NETWORK TO TRAFFIC FLOW DATA

In this chapter we will first briefly discuss the problem to be solved and the data set(s) used. We will then discuss the network architecture chosen and it's variants. Finally, we will describe the results.

3.1 The problem

Fundamentally, the local control system will be designed to keep two major traffic parameters, average speed and average density, within certain stable limits. The design is based upon modern control theory and employs a state-space description of the traffic process as a reference model. During the design, this control strategy will frequently need testing and calibration against circumstances that are as realistic as possible. Since full scale experimentation is hazardous, we must rely on computer simulation techniques and to that end we need a computer program to emulate traffic behaviour sufficiently reliably. Since we gathered a large amount of relatively detailed traffic data in a previous stage of the project, it was decided to try and use these data as a basis to develop this computer program. The choice of neural networks as a modelling tool to that end was based on the following considerations:
- the method allows a large variety of data and rather different forms of representation (metric as well as nominal) to be used simultaneously in a single model which is decidedly advantageous in the representation of traffic phenomena.
- the method requires only a minimum of structural preconceptions for such a model: in this sense it may rapidly produce a workable" black box" model. Unfortunately, the nature of the fitted model is often opaque to the modeller, and requires extra analyses to understand exactly how the model is representing the data.
- once a model has been "fit", the resulting computer code is relatively simple and executes fast, which greatly enhances the speed of the emulation of the complete control-system; this contrary to other, more "causal", models that require considerably more computing time due to their iterative nature.

The nature of the data used by this model derives from the requirements of the control system itself : all systems that are proposed in literature employ averaged data about speed and density in which the period over which the averaging takes place and the location (per lane or all lanes together) varies between authors and applications. Anyway, the averaging period should not be so long that major changes in the parameters are suppressed, therefore in practice these periods vary between 30 seconds and 15 minutes. In our case, the averaging period and the location is chosen in such a way, that all kinds of wishes in this respect can be accommodated: we use an averaging period of 30 seconds, which is the shortest period found in practice and averaging per lane, which always allows aggregation if necessary.

3.2 The Data

For our modelling attempts, we used data obtained in a 10 hour observation of the dutch highway A13. The data are taken from 64 induction loops imbedded in the road over a stretch of ca. 8 kilometres and a dis-

tance between measuring sites of 300-500 metres. An induction loop is present in each lane and per loop the following parameters are measured:
-time of passage since the start of the experiment (in milliseconds),
-speed of the passing vehicle in km/h,
-approximate length of the vehicle.
Since the scanning of all 64 channels takes place within a few microseconds, the scanning period is irrelevant and the registration of passing vehicles can be considered event driven. So, all passing cars are measured during the 10 hour observation period (apart from those that pass "in between" loops during overtaking; this sometimes results in a faulty measurement, but most often in a complete "miss").

Although the control system itself traditionally mainly operates on two parameters, averaged speed and density (vehicles/m), we want our model to be as accurate as possible. Since induction loops also provide data about vehicle length, we decided to use this parameter to adapt the descriptive variables. Based on the assumption that the predictions of the neural network might be best if we provide as much information about the traffic process as available we decided to increase the number of parameters beyond the two mentioned above. (In addition, we also felt that later stage, we might need a greater level of detail).

Therefore, the 'raw' data mentioned above was coded and selected in the following way:
- average speed
- average lane occupancy
- average 'production' , a new parameter indicating passing vehicle length per time unit (m veh /sec)
- average vehicle length

We calculated these averages as moving averages over an averaging period of 60 seconds, using a discounted least squares algorithm.

We did not use the complete data-set, but attempted to fit the neural network model on data derived from part of the second half of the observation period (i.e., between 14:00h and 16:00h). The reason for this is twofold: this part of the data contains the most variation in traffic circumstances, and we also wanted to validate the model against a part of the data that was not included in the actual model fitting.
In addition, we only considered a small number (of the sixteen) locations. This was done for two reasons: simplicity, and reduction in computing time.

The modelled data-set was produced by a computer program specially developed for this purpose.

After many false starts[7], a final data-set was created. It consisted of four

[7]Namely, unforeseen progress in the development of a reference model for the adaptive control strategy lead to a continuous flow of new insights into the nature of traffic flow processes (see Polak & Heijer, 1993). These new insights lead to a continuous redefinition of relevant traffic flow variables. This resulted in the obsolescence of to-be-modelled data-sets and of the models which were derived for those data-sets. It is not unlikely that the data-set described here, and the resulting neural network model, will

highway locations, each separated by a distance of 500 m. from the previous location. Each location included three lanes, and the second location had an on-ramp. The first, third, and fourth location had (3 lanes times 4 variables per lane =) twelve variables per measurement period; the second location (with an on-ramp) had sixteen variables. 598 measurement periods (of 12 seconds each) were studied in the modelling effort.

The input and output patterns for each clock tick (of 15 seconds) were constructed in the following way. First of all, the values for all variables for the first three locations (A,B, and C) measured at time t_i were assembled into an input vector. Secondly the values for all variables for the last three locations (B,C, and D) measured at time at time t_{i+1} were assembled into an output vector.
Locations A and D were therefore considered to be the spatial boundaries of the modelled system. The parameters of location B at time t_{t+1} could be predicted by the parameters of the upstream, the identical, and the downstream location, at a previous moment in time. The same is true for location C. Parameters for locations A were never predicted, and parameters for location D were only predicted by parameters at the identical and the up-stream location, at a previous moment in time.
Only one previous time step was used for each to-be-predicted output pattern.

3.3 Model Characteristics and Architecture

Our primary goal was to achieve a reasonable model fit with a minimum of computation: that is, we wanted an accurate as possible description of the traffic flow data. Analysis of what the network was actually learning was only considered to be a means towards that goal, and not a goal in itself. In addition, exploration of alternative network architectures was also of secondary concern.

In the initial stages of this study (where we considered data-sets which were subsequently discarded), the work cycle was: choose a network configuration, fit it to the data, look at the results, and make a guess as to how it could be improved. Since, as we previously mentioned, it was virtually impossible to systematically vary all possible combinations of choices, it would appear that the present authors were doing their own form of hill-climbing!

Some of the variations considered were:
- using the hyperbolic tangent function, instead of the logistic function, and doing the appropriate transformations;
- using different combinations of learning parameters, and learning parameter schedules for each network layer;
- retrying the same network but with new random initializations;
- adding random noise to node activation, and occasionally jogging connection weights (by adding noise);
- adding new nodes (and weights) to a layer after the old nodes (and weights) had already been fit;
- using speed-up techniques (the extended bar delta bar technique

---

eventually be subject to the same fate.

seemed to be quite useful in this context);
- adding bias terms to nodes within one or more layers (which allowed the logistic function to be translated along its' summed input axis);

etc.

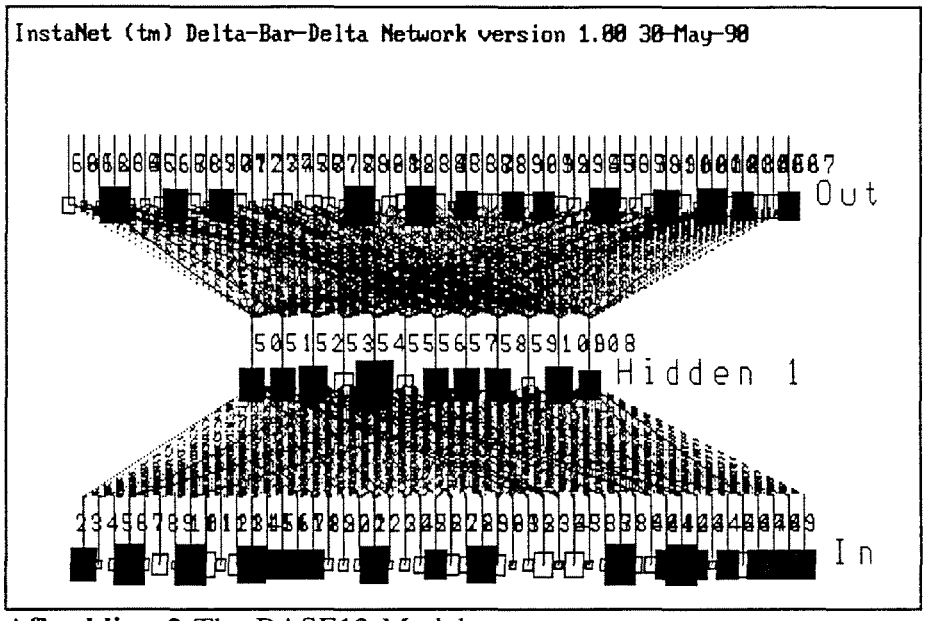
Eventually, however, after a great deal of trial and error, we were able to choose (computer) program and model parameters which seemed to work well. We will skip a discussion of the details of this learning process (and the discarded models and data-sets), and only consider a description of the general characteristics of the basic model, followed by descriptions of each of a number of variations on the basic model.

All of the variations on the basic model considered consisted of a number of common characteristics. They all used the logistic transform, with injection of uniformly distributed random noise, with no data clipping, and no derivative offsets. All models had one hidden layer. There were no direct connections from the input to the output layer. All hidden and all output nodes had a bias term. The learning rates for the hidden and output layers were rather large for only the first hundred passes though the data set, and then drastically reduced. Batch learning was used, with random pattern sampling.

It should be noted that this general type of network (and its learning procedure) is a fairly simple, standard, and direct one, which could be recomputed without exotic software by anyone with enough patience. The only real deviation from the 'standard form' is the use of the extended bar delta bar learning procedure, which was used to speed up the learning process.

This general form is rather pleasing in that it minimizes the number of a priori assumptions about underlying structure. On the other hand, a priori assumptions can greatly enhance clarity of understanding (as well as learning speed and vulnerability to catastrophic forgetting).

Starting from this general architecture, we investigated a number of variations on this basic theme. This was done in order to ascertain the possibility of obtaining 'cheaper' (in the sense of less inter-node connections), yet adequate, models, with a more transparent internal structure.



Afbeelding 2 The BASE12 Model

First of all, we developed the so-called "standard" model. The hidden layer, consisting of twelve nodes, was fully connected to both the input and output layers (see Fig. 2). We refer to this as the BASE12 model.

Secondly, we considered a similar model, the so-called BASE40 model. The major difference with the previous model is that the hidden layer consists of 40 nodes instead of 12.

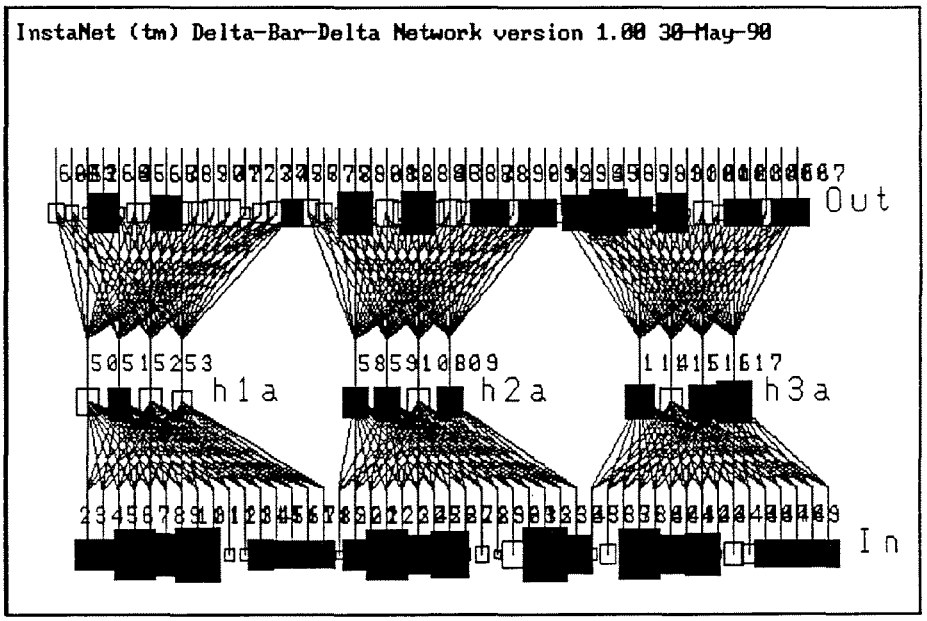


Figure 3 The STRUC3x4 Model

Third of all, we considered a model which also consisted of a hidden layer of 12 nodes, as in the BASE12 model. However, the 12 nodes were divided into three groups of four nodes, each group corresponding to one pair of (input-output) measurement locations. Namely, the connections for the hidden nodes (of each group) were fully connected to all of the variables for only one input location at time t_i, and to all of the variables for the following location at time t_{i+1}. (See Fig. 3.) Let us designate this as the STRUC3x4 model.

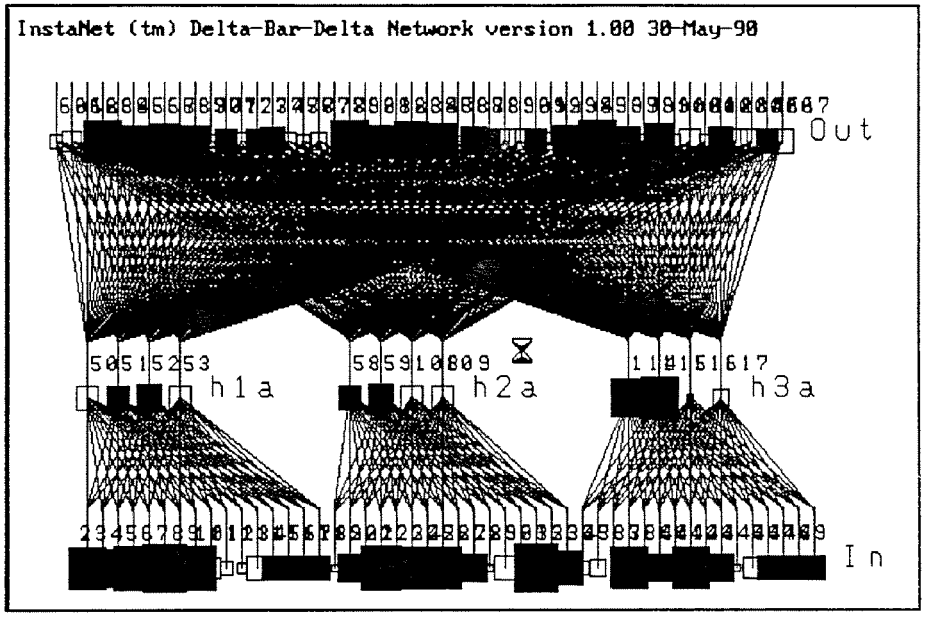


Figure 4 The INPUT3x4 Model

A fourth model, the so-called INPUT3x4 model, was somewhat similar to the STRUC3x4 model. Namely, the input connections were structured similarly to that model. However, the connections from each hidden node was fully connected to all of the output nodes for their respective locations. (See Fig. 4.)

Two other models were considered, but not implemented due to time constraints and computer problems. One model was an inverted INPUT3x4 model: the hidden nodes were fully connected to the input nodes, and connected to the variables of only one output measurement location. The other model was similar to the STRUC3x4 model, yet the connections to and from the hidden nodes were forced to have identical values for all pairs of locations. We will not further discuss these two models in this present report.

The purpose of fitting these four models was as follows. The BASE12 was intended to obtain a baseline model with a good fit, and with a limited number of hidden (or latent) variables. Since each input (and output) pattern has forty variables, and the hidden layer only twelve, we are clearly implementing a data reduction. This model can viewed as a global processing model, since all of the input locations are used to predict all of the output locations. The BASE40 model was intended to see to what extent the fit could be improved, albeit with a loss of parsimony (40 hidden variables instead of only 12). The STRUC3x4 model considers the consequences of only assuming local influences of traffic at one location at one point in time on the following location at the next moment in time, with the assumption that each pair (of following) locations is unique. The assumption of local influences can lead to local processing, which would have decided advantages in a real world implementation. The INPUT3x4 model was intended as a first step in investigating how the local processing assumptions could be systematically relaxed into an adequate semi-local model. Unfortunately, practical constraints precluded a systematic investigation of this aspect.

3.4 Model Fits

These neural network models minimize the (root mean) sums of squares of error (i.e., the differences between the actual 'output' variables and the predictions of the neural network.) In the following we will only consider comparisons based on those root mean sums of squares. (Actually, one should also take the number of connections in a model, or degrees of freedom, also into consideration. However, this will not be the main thrust of our comparisons.)

The BASE12 model reliably achieved a fit of approximately 0.0750. (A fit of 0.000 refers to perfect prediction.) The BASE40 model, with an additional 28 hidden nodes, was able to improve this to only about 0.0675. This would seem to be a minuscule improvement, made at the cost of a large loss of parsimony.
The STRUC3x4 model was only able to achieve a fit of about 0.1525, even though it used only 33% of the number of connections as the BASE12 model. Whether this trade-off between increased simplicity versus decreased predictive accuracy is desirable, is another question.
The INPUT3x4 model achieved a fit of approximately 0.1100, which intermediate to the BASE12 and the STRUC3x4 models, both in terms of fit and in terms of number of connections. This result is somewhat disappointing. We had hoped that we could have purchased more predictive accuracy for the increased cost (of adding connections). Further systematic investigation should reveal whether a principled selection of additional connections (relative to the STRUC3x4 model) could achieve a better cost/benefit ratio.

3.5 Comparison to a Linear Model

As we mentioned above, neural networks are capable of describing non-linear relations between dependent and independent variables, something which is, per definition, not possible with linear models. By comparing the network solution to a linear model, we can obtain some idea of the amount of no-linearity in the data-set, and some indication of the value added by the above effort.

Of course, we would not want to compare any linear model with the neural network, nor would we want to compare the network with a specially tailored time-series model. (Namely, the data-set was heavily filtered, using a discounted weighted least square model (see above). This created rather large auto- and cross-correlations. Time series modelling could then be profitably used for fitting the data.) Rather, we would want to compare a model with a similar architecture. That is, we would want to use the same information (c.q., structure of input and output variables), and a similar number of latent (or hidden) variables.

We felt that a linear canonical correlation analysis (of the data-set used by the network) would enable a fair comparison between the BASE12 and BASE40 network solutions and linear models. (Comparisons between the structured network solutions and a linear model would require setting some linear model coefficients equal to zero. This would have required a special modelling effort, and was therefore not attempted.)

The BASE12 network solution resulted in an average 'explained variance' (or average squared correlation coefficient) equal to 79.3%. This is, prima facie, a rather good fit, yet is not entirely surprising having noted that the data is heavily filtered. The average explained variance of a twelve dimensional linear solution (as indicated by canonical redundancy analysis) was somewhat less, 76.6%. (Please note that the number of latent variables and the number of connections/coefficients are equal in both cases.)
The linear model required 15 latent dimensions in order to achieve the accuracy of the BASE12 neural network model; the network model was able to achieve the accuracy of the twelve dimensional linear solution in about 10-11 dimensions[8].

It would therefore appear that the neural network achieves a slightly better fit than a comparable linear model. Whether this difference, of about 2.5 percentage points of explained variance, would have any far reaching real-world consequences is a point of discussion beyond the scope of this report.

It should be pointed out that analysis of the linear model indicated that more that twelve (or even fifteen) latent dimensions were statistically significant predictors (although 12 dimensions seems to be adequate). We therefore also compared the BASE40 solution with the complete 40 dimensional linear canonical correlation results.
Both (saturated) models were able to achieve about 85.5% explained variance of the output variables.
Furthermore, the first set of variables (the input or independent variables) in the linear canonical correlation analysis was able to explain maximally 98.7% of the variance of the network predictions. (The network predictions explained less, 92.2%, of those independent variables.)

Our conclusions are therefore twofold:
-the use of non-linear transforms (i.e., the use of a neural network) results in a somewhat more parsimonious description of traffic flow characteristics in the present case (as compared to a similar model without those transforms). This increase in parsimony, however, can hardly be considered dramatic.
-in the limit (i.e., the saturated models), the linear and the neural network models are more or less identical. The neural network would therefore seem to achieve its' superior parsimony by filtering error. Since we already made extensive attempts to filter out error within variables (see above), it would seem that the network is filtering out correlated errors over variables. (It may also be possible that the filtering procedure introduced systematic correlated errors. See Footnote 8.)
Whether this speculation is correct, and whether these hypothetical correlated errors are within or over measurement locations is unknown at this moment.

---

[8]Interestingly enough, this 2.5% superiority of the neural network was also found in preliminary studies done on data from other locations, with different time windows, smoothing assumptions, and variables. This superiority was <u>not</u> found when no smoothing was done.

4.0 DISCUSSION AND CONCLUSIONS

The purpose of this study was to use a neural network to predict highway traffic flow characteristics on the basis of measurements made at a previous moment in time. Once fit, such a model could then be used to test and calibrate an adaptive control strategy. As such, this study should be viewed as part of a larger effort to develop such a system-wide strategy (Wu & Heijer, 1991).

Neural network models were chosen because, once fitted, they are very fast, and because they are capable of detecting and utilizing complex, non-linear relationships in the data. In addition, it was felt that the use of a variety of modelling methodologies would contribute to the robustness of the overall effort.

Raw data was obtained from induction loops located on the Dutch national highway A13. This data was coded and smoothed in a manner commensurate with theoretical insights developed by Polak & Heijer (1993), and fit to a neural network.

The overall fit of the neural network model to a description of traffic flow characteristics was rather good: on the average, 79.3% of the variance of the smoothed ('future') variables was 'explained' by the model. This fit was, however, only somewhat better than a comparable multivariate linear model, which explained 76.6%. It remains to be seen whether this improvement in predictive power is of practical significance, or whether, for all intents and purposes, the traffic flow characteristics (as measured and defined in this study) can be viewed as a linear process with noise. (The nature of this 'noise' is yet to be determined.)

It remains uncertain, however, whether this generalization also applies to lower levels of aggregation, to other traffic situations, and to measurements over larger sections on a highway network. (For example, congestion can lead to oscillations in flow characteristics, and downstream conditions can also effect upstream conditions.)

A number of limitations of the present study should be pointed out. In the first place, a model was fitted, yet not validated on another, similar road section, or on the same road section, on another day. Such a validation is necessary, and can be implemented without much difficulty. A second limitation is that we have no idea of how well the fitted model parameters (or the general model itself) would generalize to other road sections and/or traffic conditions. It would be foolish (and risky) to assume that such a generalization would be automatic: it has to be explicitly tested.

A additional point should be made concerning a central aspect of the neural network model used: it can continue to learn, even after initial training, yet it can also forget. This is not a problem if we fit the model to a given learning data set, and then 'freeze' the parameters. If, however, we allow the model to continue learning, in the hope that new, unforeseen regularities can also be assimilated, then we run a risk that acquired knowledge about old, known situations may also be (partially) lost. Due to the possibly severe consequences in this case, it is imperative that guidelines be formulated for the supervision and evaluation of ongoing

neural network adaptation.

It should be emphasized that these precautions are not unique for the application of neural networks: they make methodological good sense for any model.

In conclusion, we can state that neural networks can be reliably applied to the prediction of traffic flow characteristics. It has, however, not been demonstrated that such network models are unquestionably superior to other, simpler models. Nevertheless, this seems to be more of a question of the investigated data-set than the potential power of the models. We fully expect that, in the near future, neural network models will be fruitfully applied to many problems of traffic flow measurement, prediction, and control.

LITERATURE

Anderson, J.,A., & Rosenfeld, E. (eds.), 1988, NEUROCOMPUTING: Foundations of Research, The MIT Press, Cambridge, Massachusetts.

Wu, Bai-fan, Heijer, T., 1991,A Hierarchical Adaptive Control Scheme for a Freeway Network based on a Twofold Modelling and Control Strategy, SWOV, Leidschendam.

McClelland, J.L., Rumelhart, D.E., & The PDP Research Group, 1986, PARALLEL DISTRIBUTED PROCESSING: Explorations in the Microstructure of Cognition, vol. 2., The MIT Press, Cambridge, Massachusetts.

NeuralWare, Inc., 1991, Neural Computing, NeuralWare, Inc., Pittsburgh, Pennsylvania.

Polak, P., & Heijer, T., 1993, The development and implementation of a theoretical model for traffic control, forthcoming.

Rumelhart, D.E., McClelland, J.L., & The PDP Research Group, 1986, PARALLEL DISTRIBUTED PROCESSING: Explorations in the Microstructure of Cognition, vol. 1, The MIT Press, Cambridge, Massachusetts.

Appendix A

The computer code for the neural network, written in the programming language C, is listed below . The code represents the last model that was fitted which contains 3 instead of 4 cross-sections namely: section A with 3 lanes, section B with 3 lanes and an on-ramp and section C again with 3 lanes. For each section there were 16 parameters: 4 for each lane plus 4 for an on- or off ramp: if none present, the parameters are 0.
This program transforms input vector Yin into prediction Yout. Both in - and outputvector contain data for all 3 subsequent cross-sections in the following manner:
first cross-section lane 1-4: vehicle length, production, lane occupancy, speed
second cross-section : etc.
Thus, per cross-section there are 4 variables per lane, which, with 3 cross-sections and an on-ramp, leads to a total of 48 for the complete vector.
Lane 1 in those cross-sections is always the leftmost lane, lane 4 represents an on-ramp, if present (if not, the parameters for lane for are all 0).
The predictions must be primed once with a measured state in all 3 cross-sections. After this initiation, the model can be run by providing new parameter values only for the first cross-section (the first 12 parameters) and the on-ramp (lane 4 of cross-section 2).
There are several possibilities to obtain new parameters for this first section:
- random draw from existing data in a limited time window: the window may be shifted over these data to simulate longer periods
- sythesize data on the basis (simultaneous) distribution properties of the variables.
The first option is simple and fast but limited to the range of parameters measured. The second option permits larger variations than present in the original data, which can be useful to investigate extremes, but requires more preparation.

The remaining varables of the input vector can be obtained by copying the relevant parts of the output vector Yout to the input vector:
the first 12 positions of Yout (prediction for the state of the second cross-section) move to the positions 17 to 32 of Yin, 17-32 of Yout is copied to 33- 48 of Yin. Positions 29-32 of Yin correspond to the on-ramp of the system nad these parameters may be generated by a control algorithm.

In this way, we obtain a prediction of the state of cross-section 3 in Yout 33-48, incorporating the effect of the initial state and the (possibly metered) input of the on-ramp.

Appendix A

```
/*    Control Strategy is: <dbd> */

#if __STDC__
#define  ARGS(x) x
#else
#define  ARGS(x) ()
#endif /* __STDC__ */

/* --- External Routines --- */
extern double exp   ARGS((double));
/* *** MAKE SURE TO LINK IN YOUR COMPILER's MATH LIBRARIES *** */

#if __STDC__
int NN_Recall( void *NetPtr, float Yin[48], float Yout[48] )
#else
int NN_Recall( NetPtr, Yin, Yout )
void  *NetPtr; /* Network Pointer (not used) */
float Yin[48], Yout[48];  /* Data */
#endif /* __STDC__ */
{
   static float  Xout[110]=0;  /* work arrays */
   long   ICmpT;  /* temp for comparisons */

   /* *** WARNING: Code generated assuming Recall = 0 *** */


   /* Read and scale input into network */
   Xout[2] = Yin[0] * (2.6315815) + (-11.263169);
   Xout[3] = Yin[1] * (0.39999999) + (-0.37999999);
   Xout[4] = Yin[2] * (11.111111) + (-0.22222222);
   Xout[5] = Yin[3] * (0.13774104) + (-3.6611569);
   Xout[6] = Yin[4] * (0.89285723) + (-3.8214291);
   Xout[7] = Yin[5] * (0.42553192) + (-0.40000001);
   Xout[8] = Yin[6] * (16.666667) + (-0.83333336);
   Xout[9] = Yin[7] * (0.17331027) + (-4.351821);
   Xout[10] = Yin[8] * (0.29154516) + (-1.5072885);
   Xout[11] = Yin[9] * (0.29761906) + (-0.36309526);
   Xout[12] = Yin[10] * (20) + (-1);
   Xout[13] = Yin[11] * (0.21598276) + (-4.8056164);
   Xout[14] = Yin[12];
   Xout[15] = Yin[13];
   Xout[16] = Yin[14];
   Xout[17] = Yin[15];
   Xout[18] = Yin[16] * (2) + (-8.5799999);
   Xout[19] = Yin[17] * (0.47619048) + (-0.45714285);
   Xout[20] = Yin[18] * (11.111111) + (-0.22222222);
   Xout[21] = Yin[19] * (0.17605639) + (-5.0228889);
   Xout[22] = Yin[20] * (0.94339628) + (-4.0660379);
   Xout[23] = Yin[21] * (0.49261084) + (-0.50246305);
   Xout[24] = Yin[22] * (16.666667) + (-0.83333336);
   Xout[25] = Yin[23] * (0.19762848) + (-5.1027673);
   Xout[26] = Yin[24] * (0.39370079) + (-1.9448819);
   Xout[27] = Yin[25] * (0.5181347) + (-0.67357508);
   Xout[28] = Yin[26] * (16.666667) + (-1);
   Xout[29] = Yin[27] * (0.28901726) + (-6.5953737);
   Xout[30] = Yin[28] * (0.21367524) + (-0.91880359);
   Xout[31] = Yin[29] * (0.33557044) + (-0.771812);
   Xout[32] = Yin[30] * (50.000001) + (-0.5);
   Xout[33] = Yin[31] * (0.24570026) + (-5.7886981);
   Xout[34] = Yin[32] * (3.3333312) + (-14.69999);
   Xout[35] = Yin[33] * (0.42016808) + (-0.38655464);
   Xout[36] = Yin[34] * (11.111111) + (-0.33333333);
   Xout[37] = Yin[35] * (0.13192612) + (-3.5580474);
   Xout[38] = Yin[36] * (1.0416666) + (-4.5624999);
   Xout[39] = Yin[37] * (0.62499994) + (-0.64999992);
   Xout[40] = Yin[38] * (16.666667) + (-0.83333336);
   Xout[41] = Yin[39] * (0.17241382) + (-4.2879317);
   Xout[42] = Yin[40] * (0.30769231) + (-1.6553847);
   Xout[43] = Yin[41] * (0.49751241) + (-0.5820895);
   Xout[44] = Yin[42] * (16.666667) + (-1);
   Xout[45] = Yin[43] * (0.25062658) + (-5.5814541);
   Xout[46] = Yin[44];
   Xout[47] = Yin[45];
   Xout[48] = Yin[46];
   Xout[49] = Yin[47];
LAB110:
   /* Generating code for PE 0 in layer 3 */
   Xout[50]  = (float)(-0.63495332) + (float)(-0.011373612) * Xout[2] +
      (float)(0.21471745) * Xout[3] + (float)(-0.24948266) * Xout[4] +
      (float)(0.91676611) * Xout[5] + (float)(0.5359) * Xout[6] +
      (float)(-0.58358109) * Xout[7] + (float)(-0.22994356) * Xout[8] +
      (float)(-0.61512858) * Xout[9] + (float)(-0.68383133) * Xout[10] +
      (float)(-0.431142) * Xout[11] + (float)(0.76919115) * Xout[12] +
```

```
    (float)(-0.46741852) * Xout[13] + (float)(-0.31274533) * Xout[14] +
    (float)(-0.25629923) * Xout[15] + (float)(-0.19553611) * Xout[16] +
    (float)(-0.33724785) * Xout[17] + (float)(-0.66709805) * Xout[18] +
    (float)(-0.79743904) * Xout[19] + (float)(-0.011748856) * Xout[20] +
    (float)(0.9651677) * Xout[21] + (float)(1.197683) * Xout[22] +
    (float)(-0.070656903) * Xout[23] + (float)(0.032656174) * Xout[24] +
    (float)(-0.66685259) * Xout[25] + (float)(-0.11587924) * Xout[26] +
    (float)(-1.7267398) * Xout[27] + (float)(0.32751361) * Xout[28] +
    (float)(-0.24657559) * Xout[29] + (float)(-0.058521315) * Xout[30] +
    (float)(1.483824) * Xout[31] + (float)(1.1801918) * Xout[32] +
    (float)(1.7115626) * Xout[33] + (float)(0.69951856) * Xout[34] +
    (float)(-0.06650579) * Xout[35] + (float)(0.15894367) * Xout[36] +
    (float)(0.35241473) * Xout[37] + (float)(-1.3754277) * Xout[38] +
    (float)(0.46970424) * Xout[39] + (float)(-0.0043533775) * Xout[40] +
    (float)(-1.3404995) * Xout[41] + (float)(1.7123841) * Xout[42] +
    (float)(-1.5074114) * Xout[43] + (float)(0.93532354) * Xout[44] +
    (float)(1.6701519) * Xout[45] + (float)(-0.33005369) * Xout[46] +
    (float)(-0.24610591) * Xout[47] + (float)(-0.21275181) * Xout[48] +
    (float)(-0.26227057) * Xout[49];
Xout[50] = 1.0 / (1.0 + exp( -Xout[50] ));

/* Generating code for PE 1 in layer 3 */
Xout[51]  = (float)(0.54086202) + (float)(-0.3700186) * Xout[2] +
    (float)(-0.18168452) * Xout[3] + (float)(-0.055866528) * Xout[4] +
    (float)(0.12926193) * Xout[5] + (float)(-0.06596759) * Xout[6] +
    (float)(-0.21907549) * Xout[7] + (float)(-0.042917494) * Xout[8] +
    (float)(0.70855105) * Xout[9] + (float)(-0.099762164) * Xout[10] +
    (float)(0.27268153) * Xout[11] + (float)(0.014543353) * Xout[12] +
    (float)(0.080683254) * Xout[13] + (float)(0.31348673) * Xout[14] +
    (float)(0.17603369) * Xout[15] + (float)(0.17072241) * Xout[16] +
    (float)(0.31045559) * Xout[17] + (float)(-0.66689813) * Xout[18] +
    (float)(0.1236593) * Xout[19] + (float)(-0.45657286) * Xout[20] +
    (float)(0.38499913) * Xout[21] + (float)(-0.090728104) * Xout[22] +
    (float)(0.42508364) * Xout[23] + (float)(0.099161647) * Xout[24] +
    (float)(0.088184446) * Xout[25] + (float)(-0.87408054) * Xout[26] +
    (float)(-0.10667372) * Xout[27] + (float)(-0.53787899) * Xout[28] +
    (float)(0.47143811) * Xout[29] + (float)(-0.11706431) * Xout[30] +
    (float)(0.53921086) * Xout[31] + (float)(0.50191516) * Xout[32] +
    (float)(-0.65274042) * Xout[33] + (float)(-1.8541684) * Xout[34] +
    (float)(0.29308212) * Xout[35] + (float)(0.033048846) * Xout[36] +
    (float)(-0.2543346) * Xout[37] + (float)(-0.11308935) * Xout[38] +
    (float)(0.031081337) * Xout[39] + (float)(-0.081991494) * Xout[40] +
    (float)(-0.35203421) * Xout[41] + (float)(-0.18631627) * Xout[42] +
    (float)(-0.57218748) * Xout[43] + (float)(-0.17646015) * Xout[44] +
    (float)(0.3991428) * Xout[45] + (float)(0.29120764) * Xout[46] +
    (float)(0.17695381) * Xout[47] + (float)(0.29773137) * Xout[48] +
    (float)(0.27420762) * Xout[49];
Xout[51] = 1.0 / (1.0 + exp( -Xout[51] ));

/* Generating code for PE 2 in layer 3 */
Xout[52]  = (float)(-0.17357253) + (float)(0.51026928) * Xout[2] +
    (float)(-0.09376654) * Xout[3] + (float)(-0.054549322) * Xout[4] +
    (float)(-0.31562617) * Xout[5] + (float)(0.12599145) * Xout[6] +
    (float)(-0.1048751) * Xout[7] + (float)(-0.17034683) * Xout[8] +
    (float)(-0.21874747) * Xout[9] + (float)(-0.50566018) * Xout[10] +
    (float)(0.071527988) * Xout[11] + (float)(-0.023923784) * Xout[12] +
    (float)(0.12305678) * Xout[13] + (float)(-0.10094954) * Xout[14] +
    (float)(0.021821555) * Xout[15] + (float)(0.011320068) * Xout[16] +
    (float)(-0.060499124) * Xout[17] + (float)(0.23596604) * Xout[18] +
    (float)(0.088493764) * Xout[19] + (float)(-0.19103557) * Xout[20] +
    (float)(0.12294304) * Xout[21] + (float)(0.19326495) * Xout[22] +
    (float)(0.3334755) * Xout[23] + (float)(-0.33368164) * Xout[24] +
    (float)(0.18300989) * Xout[25] + (float)(-1.2668388) * Xout[26] +
    (float)(-1.3534703) * Xout[27] + (float)(0.31445384) * Xout[28] +
    (float)(-0.060470361) * Xout[29] + (float)(-1.072058) * Xout[30] +
    (float)(-0.81146157) * Xout[31] + (float)(1.2040931) * Xout[32] +
    (float)(0.20241591) * Xout[33] + (float)(0.87953722) * Xout[34] +
    (float)(0.6056127) * Xout[35] + (float)(-0.89499015) * Xout[36] +
    (float)(0.44661835) * Xout[37] + (float)(0.4531039) * Xout[38] +
    (float)(1.5543835) * Xout[39] + (float)(-0.010512762) * Xout[40] +
    (float)(0.78583872) * Xout[41] + (float)(-0.93994778) * Xout[42] +
    (float)(-0.85553789) * Xout[43] + (float)(0.25198886) * Xout[44] +
    (float)(0.16974337) * Xout[45] + (float)(-0.022152375) * Xout[46] +
    (float)(-0.10255505) * Xout[47] + (float)(-0.030970838) * Xout[48] +
    (float)(-0.11070077) * Xout[49];
Xout[52] = 1.0 / (1.0 + exp( -Xout[52] ));

/* Generating code for PE 3 in layer 3 */
Xout[53]  = (float)(0.13949104) + (float)(0.10189875) * Xout[2] +
    (float)(-0.55026507) * Xout[3] + (float)(0.21085982) * Xout[4] +
    (float)(1.1541545) * Xout[5] + (float)(0.36357248) * Xout[6] +
    (float)(0.39584774) * Xout[7] + (float)(-0.046483487) * Xout[8] +
    (float)(0.5676592) * Xout[9] + (float)(-0.70274276) * Xout[10] +
    (float)(0.26898298) * Xout[11] + (float)(0.00020120759) * Xout[12] +
```

```
    (float)(-0.39827794) * Xout[13] + (float)(0.08736553) * Xout[14] +
    (float)(0.14117919) * Xout[15] + (float)(-0.025063034) * Xout[16] +
    (float)(-0.040395781) * Xout[17] + (float)(0.42403224) * Xout[18] +
    (float)(-0.92850965) * Xout[19] + (float)(0.13591525) * Xout[20] +
    (float)(0.17193057) * Xout[21] + (float)(-0.39501804) * Xout[22] +
    (float)(0.58620834) * Xout[23] + (float)(0.12553707) * Xout[24] +
    (float)(0.26471254) * Xout[25] + (float)(-1.0157481) * Xout[26] +
    (float)(-0.24690114) * Xout[27] + (float)(0.11004148) * Xout[28] +
    (float)(0.44747251) * Xout[29] + (float)(0.13314532) * Xout[30] +
    (float)(0.088985085) * Xout[31] + (float)(-1.1841747) * Xout[32] +
    (float)(-0.21754375) * Xout[33] + (float)(0.51002938) * Xout[34] +
    (float)(-0.29807806) * Xout[35] + (float)(1.2135274) * Xout[36] +
    (float)(-0.42891556) * Xout[37] + (float)(0.3979947) * Xout[38] +
    (float)(0.40132305) * Xout[39] + (float)(-0.19484264) * Xout[40] +
    (float)(-0.72129041) * Xout[41] + (float)(-0.3358019) * Xout[42] +
    (float)(-0.44215018) * Xout[43] + (float)(0.2317975) * Xout[44] +
    (float)(-1.371945) * Xout[45] + (float)(0.097398855) * Xout[46] +
    (float)(-0.013195818) * Xout[47] + (float)(0.01290418) * Xout[48] +
    (float)(0.12439603) * Xout[49];
Xout[53] = 1.0 / (1.0 + exp( -Xout[53] ));

/* Generating code for PE 4 in layer 3 */
Xout[54]  = (float)(-0.20105506) + (float)(0.22468172) * Xout[2] +
    (float)(-0.6927318) * Xout[3] + (float)(0.4091163) * Xout[4] +
    (float)(0.47609633) * Xout[5] + (float)(0.16533674) * Xout[6] +
    (float)(-0.22636612) * Xout[7] + (float)(0.069191441) * Xout[8] +
    (float)(-0.10517604) * Xout[9] + (float)(-0.25750095) * Xout[10] +
    (float)(0.0064930827) * Xout[11] + (float)(0.19565172) * Xout[12] +
    (float)(0.56902945) * Xout[13] + (float)(-0.11431003) * Xout[14] +
    (float)(-0.2028603) * Xout[15] + (float)(-0.023412565) * Xout[16] +
    (float)(-0.051798113) * Xout[17] + (float)(0.069247685) * Xout[18] +
    (float)(-0.87400419) * Xout[19] + (float)(-0.1078627) * Xout[20] +
    (float)(0.61493742) * Xout[21] + (float)(-0.21227697) * Xout[22] +
    (float)(-0.63795429) * Xout[23] + (float)(0.055845942) * Xout[24] +
    (float)(-0.0023771757) * Xout[25] + (float)(0.012703186) * Xout[26] +
    (float)(-0.53479058) * Xout[27] + (float)(0.14113513) * Xout[28] +
    (float)(0.66652876) * Xout[29] + (float)(0.63246948) * Xout[30] +
    (float)(1.4532244) * Xout[31] + (float)(1.1156636) * Xout[32] +
    (float)(-2.5521057) * Xout[33] + (float)(0.56124854) * Xout[34] +
    (float)(-0.14273176) * Xout[35] + (float)(0.12417153) * Xout[36] +
    (float)(-0.27993402) * Xout[37] + (float)(-0.23759179) * Xout[38] +
    (float)(-0.044650033) * Xout[39] + (float)(0.14766996) * Xout[40] +
    (float)(1.4994899) * Xout[41] + (float)(-0.19950646) * Xout[42] +
    (float)(-0.31506014) * Xout[43] + (float)(-0.11455775) * Xout[44] +
    (float)(0.62934107) * Xout[45] + (float)(-0.075667903) * Xout[46] +
    (float)(-0.058001682) * Xout[47] + (float)(-0.065247625) * Xout[48] +
    (float)(-0.12719113) * Xout[49];
Xout[54] = 1.0 / (1.0 + exp( -Xout[54] ));

/* Generating code for PE 5 in layer 3 */
Xout[55]  = (float)(1.2739719) + (float)(0.12358622) * Xout[2] +
    (float)(-0.0011564116) * Xout[3] + (float)(-0.5352124) * Xout[4] +
    (float)(-0.9896313) * Xout[5] + (float)(0.25093728) * Xout[6] +
    (float)(-0.58066022) * Xout[7] + (float)(-0.6361075) * Xout[8] +
    (float)(-0.10442054) * Xout[9] + (float)(-0.21724081) * Xout[10] +
    (float)(-0.13287511) * Xout[11] + (float)(-0.034477122) * Xout[12] +
    (float)(0.1293897) * Xout[13] + (float)(0.54324669) * Xout[14] +
    (float)(0.50855917) * Xout[15] + (float)(0.64062589) * Xout[16] +
    (float)(0.66025239) * Xout[17] + (float)(0.62916249) * Xout[18] +
    (float)(0.16684332) * Xout[19] + (float)(-0.40169331) * Xout[20] +
    (float)(0.038857784) * Xout[21] + (float)(0.30590743) * Xout[22] +
    (float)(-0.29211655) * Xout[23] + (float)(-0.72319937) * Xout[24] +
    (float)(-0.1232236) * Xout[25] + (float)(0.47761947) * Xout[26] +
    (float)(-0.88616043) * Xout[27] + (float)(-1.6025869) * Xout[28] +
    (float)(0.14755373) * Xout[29] + (float)(-1.5308527) * Xout[30] +
    (float)(-0.74065667) * Xout[31] + (float)(2.0079441) * Xout[32] +
    (float)(-0.6584484) * Xout[33] + (float)(0.24768226) * Xout[34] +
    (float)(0.23630543) * Xout[35] + (float)(-0.79035771) * Xout[36] +
    (float)(-1.5325606) * Xout[37] + (float)(-0.600079) * Xout[38] +
    (float)(-0.55377084) * Xout[39] + (float)(-0.67059439) * Xout[40] +
    (float)(-0.71625412) * Xout[41] + (float)(0.57290006) * Xout[42] +
    (float)(-0.74958003) * Xout[43] + (float)(-0.083881944) * Xout[44] +
    (float)(-0.034407057) * Xout[45] + (float)(0.68412584) * Xout[46] +
    (float)(0.60689795) * Xout[47] + (float)(0.5736751) * Xout[48] +
    (float)(0.59332556) * Xout[49];
Xout[55] = 1.0 / (1.0 + exp( -Xout[55] ));

/* Generating code for PE 6 in layer 3 */
Xout[56]  = (float)(0.56586242) + (float)(-0.36933059) * Xout[2] +
    (float)(-0.66693872) * Xout[3] + (float)(0.40064433) * Xout[4] +
    (float)(0.73532015) * Xout[5] + (float)(-0.93871421) * Xout[6] +
    (float)(0.35009968) * Xout[7] + (float)(0.50690699) * Xout[8] +
    (float)(0.45120305) * Xout[9] + (float)(-0.22918911) * Xout[10] +
    (float)(-0.034334045) * Xout[11] + (float)(0.5671075) * Xout[12] +
```

```
    (float)(-0.030097445) * Xout[13] + (float)(0.35256785) * Xout[14] +
    (float)(0.29476064) * Xout[15] + (float)(0.24095441) * Xout[16] +
    (float)(0.27967) * Xout[17] + (float)(-0.52735549) * Xout[18] +
    (float)(-0.034091577) * Xout[19] + (float)(1.0079882) * Xout[20] +
    (float)(0.38028258) * Xout[21] + (float)(-2.5052698) * Xout[22] +
    (float)(-0.44483191) * Xout[23] + (float)(-0.15290855) * Xout[24] +
    (float)(0.32594562) * Xout[25] + (float)(0.72212088) * Xout[26] +
    (float)(0.40015146) * Xout[27] + (float)(1.3505418) * Xout[28] +
    (float)(-0.021075681) * Xout[29] + (float)(-0.25375205) * Xout[30] +
    (float)(-1.5281478) * Xout[31] + (float)(-0.82626092) * Xout[32] +
    (float)(-0.86877435) * Xout[33] + (float)(-0.97092295) * Xout[34] +
    (float)(0.23536199) * Xout[35] + (float)(-0.26608816) * Xout[36] +
    (float)(-0.94527662) * Xout[37] + (float)(-1.3477619) * Xout[38] +
    (float)(-0.095986016) * Xout[39] + (float)(0.38609573) * Xout[40] +
    (float)(0.48803288) * Xout[41] + (float)(-1.0740809) * Xout[42] +
    (float)(0.44735703) * Xout[43] + (float)(0.213048) * Xout[44] +
    (float)(-1.152295) * Xout[45] + (float)(0.34040087) * Xout[46] +
    (float)(0.18813749) * Xout[47] + (float)(0.33993649) * Xout[48] +
    (float)(0.27205342) * Xout[49];
Xout[56] = 1.0 / (1.0 + exp( -Xout[56] ));

/* Generating code for PE 7 in layer 3 */
Xout[57]  = (float)(-0.29485977) + (float)(0.22406372) * Xout[2] +
    (float)(0.32174391) * Xout[3] + (float)(-0.63177186) * Xout[4] +
    (float)(0.0073282174) * Xout[5] + (float)(0.1586794) * Xout[6] +
    (float)(0.10309328) * Xout[7] + (float)(-0.078165293) * Xout[8] +
    (float)(-0.012613529) * Xout[9] + (float)(0.16958965) * Xout[10] +
    (float)(-0.040859964) * Xout[11] + (float)(-0.00047394569) * Xout[12]
     + (float)(0.51511228) * Xout[13] + (float)(-0.2753855) * Xout[14] +
    (float)(-0.15141223) * Xout[15] + (float)(-0.15668197) * Xout[16] +
    (float)(-0.24348216) * Xout[17] + (float)(-0.070271224) * Xout[18] +
    (float)(0.15415566) * Xout[19] + (float)(-0.5239898) * Xout[20] +
    (float)(0.74738663) * Xout[21] + (float)(-0.021152984) * Xout[22] +
    (float)(-0.252498) * Xout[23] + (float)(-0.47565472) * Xout[24] +
    (float)(0.46268404) * Xout[25] + (float)(0.71692646) * Xout[26] +
    (float)(-0.87645817) * Xout[27] + (float)(0.027768299) * Xout[28] +
    (float)(0.45354754) * Xout[29] + (float)(0.5184868) * Xout[30] +
    (float)(1.4414446) * Xout[31] + (float)(-1.0185164) * Xout[32] +
    (float)(-1.2611992) * Xout[33] + (float)(0.94681245) * Xout[34] +
    (float)(0.80429518) * Xout[35] + (float)(-0.48626092) * Xout[36] +
    (float)(0.22194217) * Xout[37] + (float)(0.053984258) * Xout[38] +
    (float)(-0.10032623) * Xout[39] + (float)(-0.3497102) * Xout[40] +
    (float)(0.38406181) * Xout[41] + (float)(0.1493817) * Xout[42] +
    (float)(-0.70406449) * Xout[43] + (float)(0.21372479) * Xout[44] +
    (float)(-1.0924242) * Xout[45] + (float)(-0.17655207) * Xout[46] +
    (float)(-0.08479514) * Xout[47] + (float)(-0.1493815) * Xout[48] +
    (float)(-0.2257648) * Xout[49];
Xout[57] = 1.0 / (1.0 + exp( -Xout[57] ));

/* Generating code for PE 8 in layer 3 */
Xout[58]  = (float)(-1.5923505) + (float)(0.18126073) * Xout[2] +
    (float)(0.048812006) * Xout[3] + (float)(-0.39372683) * Xout[4] +
    (float)(1.2156512) * Xout[5] + (float)(0.59015393) * Xout[6] +
    (float)(-0.4295682) * Xout[7] + (float)(-0.32028252) * Xout[8] +
    (float)(0.88752508) * Xout[9] + (float)(-0.46801144) * Xout[10] +
    (float)(0.53548294) * Xout[11] + (float)(0.31162575) * Xout[12] +
    (float)(-0.62029624) * Xout[13] + (float)(-0.74793446) * Xout[14] +
    (float)(-0.70039487) * Xout[15] + (float)(-0.776003) * Xout[16] +
    (float)(-0.77511573) * Xout[17] + (float)(0.14190286) * Xout[18] +
    (float)(-0.68547916) * Xout[19] + (float)(-0.10005734) * Xout[20] +
    (float)(0.93411827) * Xout[21] + (float)(-0.40749294) * Xout[22] +
    (float)(1.6403668) * Xout[23] + (float)(-0.12842295) * Xout[24] +
    (float)(0.76891935) * Xout[25] + (float)(0.275316) * Xout[26] +
    (float)(0.70664173) * Xout[27] + (float)(0.10833843) * Xout[28] +
    (float)(0.26961005) * Xout[29] + (float)(0.65789753) * Xout[30] +
    (float)(-0.26605067) * Xout[31] + (float)(0.97291052) * Xout[32] +
    (float)(1.1553342) * Xout[33] + (float)(0.37810156) * Xout[34] +
    (float)(-0.038276654) * Xout[35] + (float)(0.90224564) * Xout[36] +
    (float)(-0.061171401) * Xout[37] + (float)(0.003520912) * Xout[38] +
    (float)(1.4937876) * Xout[39] + (float)(0.051848672) * Xout[40] +
    (float)(-0.96272814) * Xout[41] + (float)(1.372381) * Xout[42] +
    (float)(-0.29946288) * Xout[43] + (float)(0.50876653) * Xout[44] +
    (float)(0.6017341) * Xout[45] + (float)(-0.69839787) * Xout[46] +
    (float)(-0.69324684) * Xout[47] + (float)(-0.75035655) * Xout[48] +
    (float)(-0.83586276) * Xout[49];
Xout[58] = 1.0 / (1.0 + exp( -Xout[58] ));

/* Generating code for PE 9 in layer 3 */
Xout[59]  = (float)(0.5416944) + (float)(0.15000497) * Xout[2] +
    (float)(-0.12338259) * Xout[3] + (float)(-0.31689623) * Xout[4] +
    (float)(0.159587) * Xout[5] + (float)(-0.11926581) * Xout[6] +
    (float)(-0.055541813) * Xout[7] + (float)(-0.48028594) * Xout[8] +
    (float)(-0.076991759) * Xout[9] + (float)(-0.42255327) * Xout[10] +
    (float)(-0.1861871) * Xout[11] + (float)(-0.075230286) * Xout[12] +
```

```
    (float)(-0.1782358) * Xout[13] + (float)(0.22479539) * Xout[14] +
    (float)(0.33376971) * Xout[15] + (float)(0.21946605) * Xout[16] +
    (float)(0.15987192) * Xout[17] + (float)(-0.33082253) * Xout[18] +
    (float)(0.65715367) * Xout[19] + (float)(0.36685103) * Xout[20] +
    (float)(0.71706665) * Xout[21] + (float)(-0.79971886) * Xout[22] +
    (float)(0.18586251) * Xout[23] + (float)(-0.19874921) * Xout[24] +
    (float)(0.016538784) * Xout[25] + (float)(-0.53400767) * Xout[26] +
    (float)(0.10623104) * Xout[27] + (float)(-1.0225739) * Xout[28] +
    (float)(0.036707461) * Xout[29] + (float)(0.44361988) * Xout[30] +
    (float)(0.20848507) * Xout[31] + (float)(0.38812453) * Xout[32] +
    (float)(0.42509636) * Xout[33] + (float)(0.37347594) * Xout[34] +
    (float)(-0.75318253) * Xout[35] + (float)(0.040563457) * Xout[36] +
    (float)(-1.1420361) * Xout[37] + (float)(-2.0562191) * Xout[38] +
    (float)(-0.2450887) * Xout[39] + (float)(-0.050812438) * Xout[40] +
    (float)(-0.44245818) * Xout[41] + (float)(-0.54716414) * Xout[42] +
    (float)(0.65176606) * Xout[43] + (float)(-0.027478792) * Xout[44] +
    (float)(0.6727007) * Xout[45] + (float)(0.3155956) * Xout[46] +
    (float)(0.33797863) * Xout[47] + (float)(0.23575915) * Xout[48] +
    (float)(0.16986759) * Xout[49];
Xout[59] = 1.0 / (1.0 + exp( -Xout[59] ));

/* Generating code for PE 10 in layer 3 */
Xout[109]  = (float)(-1.3895786) + (float)(0.063122712) * Xout[2] +
    (float)(-0.21582234) * Xout[3] + (float)(0.5492965) * Xout[4] +
    (float)(0.92746276) * Xout[5] + (float)(0.17635921) * Xout[6] +
    (float)(-0.37018704) * Xout[7] + (float)(0.22885533) * Xout[8] +
    (float)(1.1699438) * Xout[9] + (float)(-0.17973369) * Xout[10] +
    (float)(0.3108983) * Xout[11] + (float)(0.20651363) * Xout[12] +
    (float)(0.82667094) * Xout[13] + (float)(-0.57625371) * Xout[14] +
    (float)(-0.6304335) * Xout[15] + (float)(-0.69396967) * Xout[16] +
    (float)(-0.61044341) * Xout[17] + (float)(0.88602322) * Xout[18] +
    (float)(0.11514997) * Xout[19] + (float)(0.55787128) * Xout[20] +
    (float)(0.40139416) * Xout[21] + (float)(-0.97742152) * Xout[22] +
    (float)(-1.0412339) * Xout[23] + (float)(0.39499709) * Xout[24] +
    (float)(1.2295721) * Xout[25] + (float)(0.17401868) * Xout[26] +
    (float)(0.47933024) * Xout[27] + (float)(0.38328195) * Xout[28] +
    (float)(2.1356871) * Xout[29] + (float)(0.0079797367) * Xout[30] +
    (float)(-1.1368124) * Xout[31] + (float)(0.68952698) * Xout[32] +
    (float)(0.045440998) * Xout[33] + (float)(-0.049017463) * Xout[34] +
    (float)(0.96961218) * Xout[35] + (float)(0.698246) * Xout[36] +
    (float)(-0.59503067) * Xout[37] + (float)(0.63887215) * Xout[38] +
    (float)(-0.7319122) * Xout[39] + (float)(0.42402995) * Xout[40] +
    (float)(-0.36838815) * Xout[41] + (float)(-0.14430378) * Xout[42] +
    (float)(0.022293054) * Xout[43] + (float)(0.49231437) * Xout[44] +
    (float)(-0.028518299) * Xout[45] + (float)(-0.7461279) * Xout[46] +
    (float)(-0.64732462) * Xout[47] + (float)(-0.66799039) * Xout[48] +
    (float)(-0.67912275) * Xout[49];
Xout[109] = 1.0 / (1.0 + exp( -Xout[109] ));

/* Generating code for PE 11 in layer 3 */
Xout[108]  = (float)(0.65389645) + (float)(0.33503312) * Xout[2] +
    (float)(-0.089007258) * Xout[3] + (float)(0.10890045) * Xout[4] +
    (float)(-1.3216935) * Xout[5] + (float)(-0.12574467) * Xout[6] +
    (float)(-0.11352976) * Xout[7] + (float)(0.1747409) * Xout[8] +
    (float)(-0.10143614) * Xout[9] + (float)(0.15215366) * Xout[10] +
    (float)(-0.14394636) * Xout[11] + (float)(0.060874913) * Xout[12] +
    (float)(0.61284304) * Xout[13] + (float)(0.45705763) * Xout[14] +
    (float)(0.41922095) * Xout[15] + (float)(0.31419244) * Xout[16] +
    (float)(0.41155437) * Xout[17] + (float)(0.17697504) * Xout[18] +
    (float)(-1.0042534) * Xout[19] + (float)(-0.01706785) * Xout[20] +
    (float)(-0.81615114) * Xout[21] + (float)(0.41915947) * Xout[22] +
    (float)(-0.6959694) * Xout[23] + (float)(-0.40278295) * Xout[24] +
    (float)(0.72472626) * Xout[25] + (float)(0.24050766) * Xout[26] +
    (float)(-0.38085449) * Xout[27] + (float)(-0.54467082) * Xout[28] +
    (float)(0.130595) * Xout[29] + (float)(-1.5171425) * Xout[30] +
    (float)(-0.26581243) * Xout[31] + (float)(-2.0576243) * Xout[32] +
    (float)(0.55165535) * Xout[33] + (float)(-0.59858763) * Xout[34] +
    (float)(-0.25994617) * Xout[35] + (float)(-1.1683228) * Xout[36] +
    (float)(0.5564127) * Xout[37] + (float)(-0.71469492) * Xout[38] +
    (float)(-0.35802168) * Xout[39] + (float)(-0.48882803) * Xout[40] +
    (float)(1.3641639) * Xout[41] + (float)(0.46255839) * Xout[42] +
    (float)(-0.65269339) * Xout[43] + (float)(-0.51163357) * Xout[44] +
    (float)(1.1725081) * Xout[45] + (float)(0.31346831) * Xout[46] +
    (float)(0.28724936) * Xout[47] + (float)(0.41426948) * Xout[48] +
    (float)(0.31229255) * Xout[49];
Xout[108] = 1.0 / (1.0 + exp( -Xout[108] ));

/* Generating code for PE 0 in layer 4 */
Xout[60]  = (float)(-1.1271948) + (float)(-0.81842512) * Xout[50] +
    (float)(-1.3834889) * Xout[51] + (float)(-0.022965293) * Xout[52] +
    (float)(0.81243747) * Xout[53] + (float)(0.38154218) * Xout[54] +
    (float)(0.90772903) * Xout[55] + (float)(-0.72471118) * Xout[56] +
    (float)(-0.30400932) * Xout[57] + (float)(0.58435428) * Xout[58] +
    (float)(-0.29048774) * Xout[59] + (float)(0.47321105) * Xout[108] +
```

```
    (float)(1.3796418) * Xout[109];
Xout[60] = 1.0 / (1.0 + exp( -Xout[60] ));

/* Generating code for PE 1 in layer 4 */
Xout[61]  = (float)(0.99051261) + (float)(-0.30540881) * Xout[50] +
    (float)(0.90528154) * Xout[51] + (float)(0.78356338) * Xout[52] +
    (float)(-2.2607591) * Xout[53] + (float)(-3.1666543) * Xout[54] +
    (float)(-0.44249922) * Xout[55] + (float)(-0.23646095) * Xout[56] +
    (float)(2.5246212) * Xout[57] + (float)(-0.91401523) * Xout[58] +
    (float)(1.2091639) * Xout[59] + (float)(-1.7077878) * Xout[108] +
    (float)(1.0161073) * Xout[109];
Xout[61] = 1.0 / (1.0 + exp( -Xout[61] ));

/* Generating code for PE 2 in layer 4 */
Xout[62]  = (float)(0.45626181) + (float)(1.130173) * Xout[50] +
    (float)(-1.7073213) * Xout[51] + (float)(-1.1245573) * Xout[52] +
    (float)(0.89064527) * Xout[53] + (float)(1.4729378) * Xout[54] +
    (float)(-0.5808937) * Xout[55] + (float)(0.19245362) * Xout[56] +
    (float)(-3.3339746) * Xout[57] + (float)(-1.4942347) * Xout[58] +
    (float)(-0.74518216) * Xout[59] + (float)(0.086931542) * Xout[108] +
    (float)(0.80059516) * Xout[109];
Xout[62] = 1.0 / (1.0 + exp( -Xout[62] ));

/* Generating code for PE 3 in layer 4 */
Xout[63]  = (float)(-1.9432629) + (float)(-0.30953941) * Xout[50] +
    (float)(1.2693883) * Xout[51] + (float)(0.76719898) * Xout[52] +
    (float)(-0.37819272) * Xout[53] + (float)(-0.45898786) * Xout[54] +
    (float)(-1.3339252) * Xout[55] + (float)(-0.26667818) * Xout[56] +
    (float)(1.8682129) * Xout[57] + (float)(1.8805428) * Xout[58] +
    (float)(0.49274552) * Xout[59] + (float)(0.014423536) * Xout[108] +
    (float)(0.73427594) * Xout[109];
Xout[63] = 1.0 / (1.0 + exp( -Xout[63] ));

/* Generating code for PE 4 in layer 4 */
Xout[64]  = (float)(2.0163431) + (float)(0.38312757) * Xout[50] +
    (float)(0.051329836) * Xout[51] + (float)(0.0044544959) * Xout[52] +
    (float)(-0.31182596) * Xout[53] + (float)(0.94831932) * Xout[54] +
    (float)(-0.54690707) * Xout[55] + (float)(-2.4414914) * Xout[56] +
    (float)(-1.4124751) * Xout[57] + (float)(-0.50987464) * Xout[58] +
    (float)(-1.7826247) * Xout[59] + (float)(0.334185) * Xout[108] +
    (float)(-1.4672284) * Xout[109];
Xout[64] = 1.0 / (1.0 + exp( -Xout[64] ));

/* Generating code for PE 5 in layer 4 */
Xout[65]  = (float)(-0.71147937) + (float)(-1.0970083) * Xout[50] +
    (float)(0.52948028) * Xout[51] + (float)(0.22390811) * Xout[52] +
    (float)(0.47199774) * Xout[53] + (float)(-0.25515306) * Xout[54] +
    (float)(-0.15084545) * Xout[55] + (float)(-0.02536888) * Xout[56] +
    (float)(-0.22640531) * Xout[57] + (float)(2.6865695) * Xout[58] +
    (float)(0.34129471) * Xout[59] + (float)(-0.64093232) * Xout[108] +
    (float)(-2.070827) * Xout[109];
Xout[65] = 1.0 / (1.0 + exp( -Xout[65] ));

/* Generating code for PE 6 in layer 4 */
Xout[66]  = (float)(1.6143047) + (float)(0.81114674) * Xout[50] +
    (float)(-0.56400937) * Xout[51] + (float)(-0.21637194) * Xout[52] +
    (float)(-0.098756105) * Xout[53] + (float)(0.81337875) * Xout[54] +
    (float)(-1.0847054) * Xout[55] + (float)(-0.4039582) * Xout[56] +
    (float)(-1.8428776) * Xout[57] + (float)(-1.501519) * Xout[58] +
    (float)(-0.77188599) * Xout[59] + (float)(-0.4604736) * Xout[108] +
    (float)(0.35490516) * Xout[109];
Xout[66] = 1.0 / (1.0 + exp( -Xout[66] ));

/* Generating code for PE 7 in layer 4 */
Xout[67]  = (float)(-2.5193746) + (float)(-0.71139127) * Xout[50] +
    (float)(0.55171955) * Xout[51] + (float)(0.13719322) * Xout[52] +
    (float)(-0.23672403) * Xout[53] + (float)(-0.17397298) * Xout[54] +
    (float)(-0.47456336) * Xout[55] + (float)(-0.12127858) * Xout[56] +
    (float)(1.3360059) * Xout[57] + (float)(1.7202868) * Xout[58] +
    (float)(0.55110395) * Xout[59] + (float)(0.77017182) * Xout[108] +
    (float)(1.5245388) * Xout[109];
Xout[67] = 1.0 / (1.0 + exp( -Xout[67] ));

/* Generating code for PE 8 in layer 4 */
Xout[68]  = (float)(1.5630575) + (float)(-0.23494618) * Xout[50] +
    (float)(-2.0493004) * Xout[51] + (float)(-1.3465009) * Xout[52] +
    (float)(-2.3500924) * Xout[53] + (float)(0.46151495) * Xout[54] +
    (float)(0.5551433) * Xout[55] + (float)(1.235092) * Xout[56] +
    (float)(0.45352343) * Xout[57] + (float)(1.8041968) * Xout[58] +
    (float)(-1.7768726) * Xout[59] + (float)(0.35839537) * Xout[108] +
    (float)(-1.3124775) * Xout[109];
Xout[68] = 1.0 / (1.0 + exp( -Xout[68] ));

/* Generating code for PE 9 in layer 4 */
```

```
Xout[69]  = (float)(0.34848964) + (float)(-2.2175453) * Xout[50] +
    (float)(0.080968171) * Xout[51] + (float)(-1.7334762) * Xout[52] +
    (float)(-0.38086399) * Xout[53] + (float)(-0.016748372) * Xout[54] +
    (float)(-0.3129648) * Xout[55] + (float)(-0.96740031) * Xout[56] +
    (float)(-0.77607948) * Xout[57] + (float)(2.7698929) * Xout[58] +
    (float)(0.14756824) * Xout[59] + (float)(0.014937705) * Xout[108] +
    (float)(0.58674985) * Xout[109];
Xout[69] = 1.0 / (1.0 + exp( -Xout[69] ));

/* Generating code for PE 10 in layer 4 */
Xout[70]  = (float)(1.2931837) + (float)(1.1293036) * Xout[50] +
    (float)(-1.1445518) * Xout[51] + (float)(1.3071111) * Xout[52] +
    (float)(-1.5132684) * Xout[53] + (float)(0.36326271) * Xout[54] +
    (float)(-1.8802327) * Xout[55] + (float)(2.6150167) * Xout[56] +
    (float)(-0.80144083) * Xout[57] + (float)(-0.49410975) * Xout[58] +
    (float)(-2.1637807) * Xout[59] + (float)(-0.76418853) * Xout[108] +
    (float)(-0.62683147) * Xout[109];
Xout[70] = 1.0 / (1.0 + exp( -Xout[70] ));

/* Generating code for PE 11 in layer 4 */
Xout[71]  = (float)(-3.0252392) + (float)(-0.30334255) * Xout[50] +
    (float)(0.73941272) * Xout[51] + (float)(-0.030317988) * Xout[52] +
    (float)(0.19043158) * Xout[53] + (float)(0.3451598) * Xout[54] +
    (float)(-0.5137074) * Xout[55] + (float)(-1.0549057) * Xout[56] +
    (float)(0.35269666) * Xout[57] + (float)(0.54933143) * Xout[58] +
    (float)(1.120172) * Xout[59] + (float)(0.94736534) * Xout[108] +
    (float)(2.381901) * Xout[109];
Xout[71] = 1.0 / (1.0 + exp( -Xout[71] ));

/* Generating code for PE 12 in layer 4 */
Xout[72]  = (float)(-1.6351501) + (float)(-0.035657734) * Xout[50] +
    (float)(0.60131836) * Xout[51] + (float)(-1.0153981) * Xout[52] +
    (float)(-0.96337998) * Xout[53] + (float)(2.0109179) * Xout[54] +
    (float)(-2.7939463) * Xout[55] + (float)(-1.1227628) * Xout[56] +
    (float)(1.077229) * Xout[57] + (float)(0.41827017) * Xout[58] +
    (float)(1.1275314) * Xout[59] + (float)(-2.0361631) * Xout[108] +
    (float)(0.6591363) * Xout[109];
Xout[72] = 1.0 / (1.0 + exp( -Xout[72] ));

/* Generating code for PE 13 in layer 4 */
Xout[73]  = (float)(-0.83080274) + (float)(1.2367696) * Xout[50] +
    (float)(1.1079992) * Xout[51] + (float)(-2.0837095) * Xout[52] +
    (float)(0.10090644) * Xout[53] + (float)(1.1603016) * Xout[54] +
    (float)(-2.6968729) * Xout[55] + (float)(-1.8496203) * Xout[56] +
    (float)(1.2509276) * Xout[57] + (float)(-0.054938078) * Xout[58] +
    (float)(1.6586423) * Xout[59] + (float)(-0.22862975) * Xout[108] +
    (float)(-0.45432207) * Xout[109];
Xout[73] = 1.0 / (1.0 + exp( -Xout[73] ));

/* Generating code for PE 14 in layer 4 */
Xout[74]  = (float)(-1.4831536) + (float)(0.89368427) * Xout[50] +
    (float)(0.29253733) * Xout[51] + (float)(0.84712416) * Xout[52] +
    (float)(-3.1981816) * Xout[53] + (float)(3.4082227) * Xout[54] +
    (float)(2.4743552) * Xout[55] + (float)(-0.061327793) * Xout[56] +
    (float)(-3.1025629) * Xout[57] + (float)(1.3641064) * Xout[58] +
    (float)(0.041832387) * Xout[59] + (float)(-2.9901614) * Xout[108] +
    (float)(0.45638043) * Xout[109];
Xout[74] = 1.0 / (1.0 + exp( -Xout[74] ));

/* Generating code for PE 15 in layer 4 */
Xout[75]  = (float)(1.1997241) + (float)(2.1578417) * Xout[50] +
    (float)(-0.80038494) * Xout[51] + (float)(0.24383093) * Xout[52] +
    (float)(-0.84625733) * Xout[53] + (float)(-5.0739422) * Xout[54] +
    (float)(-1.9571033) * Xout[55] + (float)(-1.2238055) * Xout[56] +
    (float)(-1.0995946) * Xout[57] + (float)(0.66278946) * Xout[58] +
    (float)(2.556469) * Xout[59] + (float)(1.4499192) * Xout[108] +
    (float)(2.0539217) * Xout[109];
Xout[75] = 1.0 / (1.0 + exp( -Xout[75] ));

/* Generating code for PE 16 in layer 4 */
Xout[76]  = (float)(-0.31402117) + (float)(-0.56115401) * Xout[50] +
    (float)(-5.0220456) * Xout[51] + (float)(1.0307035) * Xout[52] +
    (float)(0.79192179) * Xout[53] + (float)(0.92074883) * Xout[54] +
    (float)(0.1491286) * Xout[55] + (float)(-1.0890059) * Xout[56] +
    (float)(1.3366988) * Xout[57] + (float)(0.42141479) * Xout[58] +
    (float)(1.5278354) * Xout[59] + (float)(-0.28532648) * Xout[108] +
    (float)(1.0869534) * Xout[109];
Xout[76] = 1.0 / (1.0 + exp( -Xout[76] ));

/* Generating code for PE 17 in layer 4 */
Xout[77]  = (float)(-0.4453508) + (float)(-0.48765531) * Xout[50] +
    (float)(1.4618384) * Xout[51] + (float)(1.0828806) * Xout[52] +
    (float)(-1.6182721) * Xout[53] + (float)(-2.4736428) * Xout[54] +
    (float)(-0.64924866) * Xout[55] + (float)(-0.34230852) * Xout[56] +
```

```
    (float)(2.3691278) * Xout[57] + (float)(-0.24325132) * Xout[58] +
    (float)(-0.15207697) * Xout[59] + (float)(-1.0746771) * Xout[108] +
    (float)(1.638507) * Xout[109];
Xout[77] = 1.0 / (1.0 + exp( -Xout[77] ));

/* Generating code for PE 18 in layer 4 */
Xout[78]  = (float)(-0.0093993209) + (float)(1.2652856) * Xout[50] +
    (float)(-0.8570205) * Xout[51] + (float)(-1.1121503) * Xout[52] +
    (float)(1.623265) * Xout[53] + (float)(0.65648186) * Xout[54] +
    (float)(-0.15524225) * Xout[55] + (float)(0.32887942) * Xout[56] +
    (float)(-2.4795859) * Xout[57] + (float)(-1.4936676) * Xout[58] +
    (float)(-0.51489735) * Xout[59] + (float)(-0.51418811) * Xout[108] +
    (float)(0.96810508) * Xout[109];
Xout[78] = 1.0 / (1.0 + exp( -Xout[78] ));

/* Generating code for PE 19 in layer 4 */
Xout[79]  = (float)(-0.77337676) + (float)(-0.44863424) * Xout[50] +
    (float)(0.65776843) * Xout[51] + (float)(0.95053434) * Xout[52] +
    (float)(-0.93394405) * Xout[53] + (float)(0.06248771) * Xout[54] +
    (float)(-1.4685494) * Xout[55] + (float)(-0.40164199) * Xout[56] +
    (float)(1.3384471) * Xout[57] + (float)(1.324085) * Xout[58] +
    (float)(-0.11933818) * Xout[59] + (float)(0.51301754) * Xout[108] +
    (float)(0.58442467) * Xout[109];
Xout[79] = 1.0 / (1.0 + exp( -Xout[79] ));

/* Generating code for PE 20 in layer 4 */
Xout[80]  = (float)(3.137306) + (float)(-0.95458806) * Xout[50] +
    (float)(0.18669866) * Xout[51] + (float)(0.43128395) * Xout[52] +
    (float)(0.068867706) * Xout[53] + (float)(-0.29709685) * Xout[54] +
    (float)(-0.89955783) * Xout[55] + (float)(-2.2190347) * Xout[56] +
    (float)(-0.83802128) * Xout[57] + (float)(-0.25351721) * Xout[58] +
    (float)(-3.2831914) * Xout[59] + (float)(-0.67998296) * Xout[108] +
    (float)(-0.10428108) * Xout[109];
Xout[80] = 1.0 / (1.0 + exp( -Xout[80] ));

/* Generating code for PE 21 in layer 4 */
Xout[81]  = (float)(-1.3307847) + (float)(-0.68005109) * Xout[50] +
    (float)(0.10318635) * Xout[51] + (float)(1.949532) * Xout[52] +
    (float)(-0.10627366) * Xout[53] + (float)(0.11747685) * Xout[54] +
    (float)(-0.75109428) * Xout[55] + (float)(0.26546109) * Xout[56] +
    (float)(-0.1810772) * Xout[57] + (float)(2.4917076) * Xout[58] +
    (float)(-0.46360749) * Xout[59] + (float)(0.051633526) * Xout[108] +
    (float)(-1.8230093) * Xout[109];
Xout[81] = 1.0 / (1.0 + exp( -Xout[81] ));

/* Generating code for PE 22 in layer 4 */
Xout[82]  = (float)(2.6315839) + (float)(0.7565732) * Xout[50] +
    (float)(-0.95833892) * Xout[51] + (float)(0.19239159) * Xout[52] +
    (float)(0.19948715) * Xout[53] + (float)(0.2337243) * Xout[54] +
    (float)(-1.3877039) * Xout[55] + (float)(0.15337388) * Xout[56] +
    (float)(-1.2589613) * Xout[57] + (float)(-2.2221169) * Xout[58] +
    (float)(-1.1016883) * Xout[59] + (float)(-1.4852508) * Xout[108] +
    (float)(1.100075) * Xout[109];
Xout[82] = 1.0 / (1.0 + exp( -Xout[82] ));

/* Generating code for PE 23 in layer 4 */
Xout[83]  = (float)(-1.8382864) + (float)(-0.3411918) * Xout[50] +
    (float)(-0.025781045) * Xout[51] + (float)(0.31228745) * Xout[52] +
    (float)(-0.99119544) * Xout[53] + (float)(0.82920599) * Xout[54] +
    (float)(-0.65316337) * Xout[55] + (float)(-0.50539988) * Xout[56] +
    (float)(0.81932914) * Xout[57] + (float)(1.2416919) * Xout[58] +
    (float)(0.86348158) * Xout[59] + (float)(1.4143608) * Xout[108] +
    (float)(1.1326553) * Xout[109];
Xout[83] = 1.0 / (1.0 + exp( -Xout[83] ));

/* Generating code for PE 24 in layer 4 */
Xout[84]  = (float)(0.30048013) + (float)(0.54728049) * Xout[50] +
    (float)(-1.4993926) * Xout[51] + (float)(-2.0250094) * Xout[52] +
    (float)(-1.6045873) * Xout[53] + (float)(0.55398297) * Xout[54] +
    (float)(1.2700931) * Xout[55] + (float)(0.87798905) * Xout[56] +
    (float)(0.62644106) * Xout[57] + (float)(1.825107) * Xout[58] +
    (float)(-1.9769086) * Xout[59] + (float)(0.58303523) * Xout[108] +
    (float)(-0.76937056) * Xout[109];
Xout[84] = 1.0 / (1.0 + exp( -Xout[84] ));

/* Generating code for PE 25 in layer 4 */
Xout[85]  = (float)(2.2584329) + (float)(-1.8088914) * Xout[50] +
    (float)(-1.322167) * Xout[51] + (float)(-0.23244019) * Xout[52] +
    (float)(-1.7197626) * Xout[53] + (float)(0.17660409) * Xout[54] +
    (float)(-1.2876936) * Xout[55] + (float)(-0.76772326) * Xout[56] +
    (float)(-0.6087507) * Xout[57] + (float)(1.073369) * Xout[58] +
    (float)(1.8392125) * Xout[59] + (float)(-0.72801536) * Xout[108] +
    (float)(-0.69041973) * Xout[109];
Xout[85] = 1.0 / (1.0 + exp( -Xout[85] ));
```

```
/* Generating code for PE 26 in layer 4 */
Xout[86]  = (float)(-0.12115224) + (float)(1.2860204) * Xout[50] +
   (float)(-0.68727338) * Xout[51] + (float)(0.17316988) * Xout[52] +
   (float)(-0.26247919) * Xout[53] + (float)(0.10292512) * Xout[54] +
   (float)(-0.41605964) * Xout[55] + (float)(1.9672805) * Xout[56] +
   (float)(0.069597267) * Xout[57] + (float)(-0.056802392) * Xout[58] +
   (float)(-1.5673718) * Xout[59] + (float)(-0.77722275) * Xout[108] +
   (float)(-0.46560061) * Xout[109];
Xout[86] = 1.0 / (1.0 + exp( -Xout[86] ));

/* Generating code for PE 27 in layer 4 */
Xout[87]  = (float)(-2.1286206) + (float)(0.038134307) * Xout[50] +
   (float)(0.87354302) * Xout[51] + (float)(0.50962937) * Xout[52] +
   (float)(-0.76444685) * Xout[53] + (float)(0.69365746) * Xout[54] +
   (float)(-1.1442401) * Xout[55] + (float)(-0.95135128) * Xout[56] +
   (float)(-0.24921077) * Xout[57] + (float)(0.21229799) * Xout[58] +
   (float)(0.87801582) * Xout[59] + (float)(1.2346022) * Xout[108] +
   (float)(1.5355805) * Xout[109];
Xout[87] = 1.0 / (1.0 + exp( -Xout[87] ));

/* Generating code for PE 28 in layer 4 */
Xout[88]  = (float)(-2.4577863e-005) + (float)(1.2992728e-005) * Xout[58];
Xout[88] = 1.0 / (1.0 + exp( -Xout[88] ));

/* Generating code for PE 29 in layer 4 */
Xout[89]  = (float)(-2.8107917e-005) + (float)(1.3743494e-005) * Xout[53]
    + (float)(1.0417771e-005) * Xout[55];
Xout[89] = 1.0 / (1.0 + exp( -Xout[89] ));

/* Generating code for PE 30 in layer 4 */
Xout[90]  = (float)(0.0001893545) + (float)(-2.9530063e-005) * Xout[50] +
   (float)(1.4433194e-005) * Xout[51] + (float)(2.3297041e-005) * Xout[52]
    + (float)(-7.4419615e-005) * Xout[53] +
   (float)(-5.8972055e-005) * Xout[54] +
   (float)(-3.5120465e-005) * Xout[55] +
   (float)(-2.8683868e-005) * Xout[56] +
   (float)(3.0169651e-005) * Xout[57] + (float)(-0.00010326075) * Xout[58]
    + (float)(-3.9426686e-005) * Xout[59] +
   (float)(-5.0844203e-005) * Xout[108] +
   (float)(1.6017189e-005) * Xout[109];
Xout[90] = 1.0 / (1.0 + exp( -Xout[90] ));

/* Generating code for PE 31 in layer 4 */
Xout[91]  = (float)(0.00021251041) + (float)(-5.3825112e-005) * Xout[50] +
   (float)(-4.2433396e-005) * Xout[51] +
   (float)(4.0391315e-005) * Xout[52] + (float)(-0.00010584209) * Xout[53]
    + (float)(1.0023429e-005) * Xout[54] +
   (float)(-9.2874398e-005) * Xout[55] +
   (float)(-4.1116495e-005) * Xout[57] +
   (float)(-6.103763e-005) * Xout[58] +
   (float)(-2.8993813e-005) * Xout[59] +
   (float)(-1.119095e-005) * Xout[108] +
   (float)(-6.2168881e-005) * Xout[109];
Xout[91] = 1.0 / (1.0 + exp( -Xout[91] ));

/* Generating code for PE 32 in layer 4 */
Xout[92]  = (float)(0.26969635) + (float)(-0.30974364) * Xout[50] +
   (float)(-2.517801) * Xout[51] + (float)(0.28506058) * Xout[52] +
   (float)(0.32746753) * Xout[53] + (float)(1.162149) * Xout[54] +
   (float)(-0.89783609) * Xout[55] + (float)(-1.0002921) * Xout[56] +
   (float)(-0.3454769) * Xout[57] + (float)(0.063768186) * Xout[58] +
   (float)(1.1734737) * Xout[59] + (float)(0.34612843) * Xout[108] +
   (float)(0.069671661) * Xout[109];
Xout[92] = 1.0 / (1.0 + exp( -Xout[92] ));

/* Generating code for PE 33 in layer 4 */
Xout[93]  = (float)(0.97986376) + (float)(-1.5181413) * Xout[50] +
   (float)(0.61071473) * Xout[51] + (float)(1.8992558) * Xout[52] +
   (float)(-2.6065402) * Xout[53] + (float)(-1.5113795) * Xout[54] +
   (float)(-0.6471867) * Xout[55] + (float)(0.02046635) * Xout[56] +
   (float)(2.3483627) * Xout[57] + (float)(0.057492416) * Xout[58] +
   (float)(-0.16072766) * Xout[59] + (float)(-0.43656364) * Xout[108] +
   (float)(-1.1315476) * Xout[109];
Xout[93] = 1.0 / (1.0 + exp( -Xout[93] ));

/* Generating code for PE 34 in layer 4 */
Xout[94]  = (float)(-1.0722711) + (float)(1.5044731) * Xout[50] +
   (float)(-0.5833627) * Xout[51] + (float)(-0.95837814) * Xout[52] +
   (float)(1.9107828) * Xout[53] + (float)(0.091979191) * Xout[54] +
   (float)(-0.098079793) * Xout[55] + (float)(0.54660648) * Xout[56] +
   (float)(-2.2612064) * Xout[57] + (float)(-1.308275) * Xout[58] +
   (float)(0.016765656) * Xout[59] + (float)(-0.92056769) * Xout[108] +
   (float)(1.7252015) * Xout[109];
Xout[94] = 1.0 / (1.0 + exp( -Xout[94] ));
```

```
/* Generating code for PE 35 in layer 4 */
Xout[95]  = (float)(0.020033343) + (float)(-0.94125074) * Xout[50] +
    (float)(0.27962023) * Xout[51] + Xout[52] +
    (float)(-1.5500464) * Xout[53] + (float)(0.65965164) * Xout[54] +
    (float)(-1.8283548) * Xout[55] + (float)(-0.56391716) * Xout[56] +
    (float)(0.76049888) * Xout[57] + (float)(1.5675797) * Xout[58] +
    (float)(-0.023777883) * Xout[59] + (float)(1.1918838) * Xout[108] +
    (float)(-0.67139995) * Xout[109];
Xout[95] = 1.0 / (1.0 + exp( -Xout[95] ));

/* Generating code for PE 36 in layer 4 */
Xout[96]  = (float)(1.3026482) + (float)(-1.5718399) * Xout[50] +
    (float)(-0.84302801) * Xout[51] + (float)(0.98278356) * Xout[52] +
    (float)(1.6147796) * Xout[53] + (float)(-0.61733937) * Xout[54] +
    (float)(-0.039766923) * Xout[55] + (float)(-1.5104465) * Xout[56] +
    (float)(1.194056) * Xout[57] + (float)(-0.77516675) * Xout[58] +
    (float)(-3.2540019) * Xout[59] + (float)(-1.3176019) * Xout[108] +
    (float)(1.7704582) * Xout[109];
Xout[96] = 1.0 / (1.0 + exp( -Xout[96] ));

/* Generating code for PE 37 in layer 4 */
Xout[97]  = (float)(-0.50115377) + (float)(-1.2574297) * Xout[50] +
    (float)(0.69005173) * Xout[51] + (float)(0.97507006) * Xout[52] +
    (float)(-0.1685814) * Xout[53] + (float)(-0.29004422) * Xout[54] +
    (float)(-0.19388328) * Xout[55] + (float)(-0.39527857) * Xout[56] +
    (float)(-0.17465004) * Xout[57] + (float)(2.2017305) * Xout[58] +
    (float)(-0.97406936) * Xout[59] + (float)(0.15925157) * Xout[108] +
    (float)(-1.5618716) * Xout[109];
Xout[97] = 1.0 / (1.0 + exp( -Xout[97] ));

/* Generating code for PE 38 in layer 4 */
Xout[98]  = (float)(1.5504092) + (float)(0.13156244) * Xout[50] +
    (float)(-0.45470008) * Xout[51] + (float)(-0.03258229) * Xout[52] +
    (float)(0.92437667) * Xout[53] + (float)(-0.17690143) * Xout[54] +
    (float)(-0.6620695) * Xout[55] + (float)(0.23948748) * Xout[56] +
    (float)(-0.67032462) * Xout[57] + (float)(-2.1200576) * Xout[58] +
    (float)(-0.64275318) * Xout[59] + (float)(-1.3591479) * Xout[108] +
    (float)(1.5086142) * Xout[109];
Xout[98] = 1.0 / (1.0 + exp( -Xout[98] ));

/* Generating code for PE 39 in layer 4 */
Xout[99]  = (float)(-1.3422076) + (float)(-0.19107366) * Xout[50] +
    (float)(-0.84976107) * Xout[51] + (float)(0.90883416) * Xout[52] +
    (float)(-2.1653829) * Xout[53] + (float)(1.7798616) * Xout[54] +
    (float)(-1.4378302) * Xout[55] + (float)(-0.54379606) * Xout[56] +
    (float)(-0.33565909) * Xout[57] + (float)(1.5350535) * Xout[58] +
    (float)(1.0401212) * Xout[59] + (float)(2.332197) * Xout[108] +
    (float)(-0.4805471) * Xout[109];
Xout[99] = 1.0 / (1.0 + exp( -Xout[99] ));

/* Generating code for PE 40 in layer 4 */
Xout[100]  = (float)(-0.17228021) + (float)(1.0267595) * Xout[50] +
    (float)(-0.32913649) * Xout[51] + (float)(-2.3633208) * Xout[52] +
    (float)(-0.83040869) * Xout[53] + (float)(-0.45237222) * Xout[54] +
    (float)(1.9302666) * Xout[55] + (float)(0.21804175) * Xout[56] +
    (float)(0.37375265) * Xout[57] + (float)(2.205935) * Xout[58] +
    (float)(-2.0602682) * Xout[59] + (float)(0.040478695) * Xout[108] +
    (float)(0.24739593) * Xout[109];
Xout[100] = 1.0 / (1.0 + exp( -Xout[100] ));

/* Generating code for PE 41 in layer 4 */
Xout[101]  = (float)(1.2131093) + (float)(-1.1169457) * Xout[50] +
    (float)(-0.42688763) * Xout[51] + (float)(-0.24864078) * Xout[52] +
    (float)(-1.1827255) * Xout[53] + (float)(0.64675373) * Xout[54] +
    (float)(-1.2052728) * Xout[55] + (float)(-0.82416272) * Xout[56] +
    (float)(-0.057926528) * Xout[57] + (float)(0.4857721) * Xout[58] +
    (float)(0.5657509) * Xout[59] + (float)(0.10218118) * Xout[108] +
    (float)(-1.5153761) * Xout[109];
Xout[101] = 1.0 / (1.0 + exp( -Xout[101] ));

/* Generating code for PE 42 in layer 4 */
Xout[102]  = (float)(-1.766288) + (float)(2.3454351) * Xout[50] +
    (float)(0.24242234) * Xout[51] + (float)(-0.25429481) * Xout[52] +
    (float)(0.739815) * Xout[53] + (float)(-0.69349694) * Xout[54] +
    (float)(1.434274) * Xout[55] + (float)(1.3433131) * Xout[56] +
    (float)(0.66296631) * Xout[57] + (float)(0.64785177) * Xout[58] +
    (float)(-1.7816589) * Xout[59] + (float)(-0.80198276) * Xout[108] +
    (float)(0.43305841) * Xout[109];
Xout[102] = 1.0 / (1.0 + exp( -Xout[102] ));

/* Generating code for PE 43 in layer 4 */
Xout[103]  = (float)(-0.45019552) + (float)(-0.40654105) * Xout[50] +
    (float)(0.32625028) * Xout[51] + (float)(0.66520005) * Xout[52] +
    (float)(-1.4969305) * Xout[53] + (float)(1.3141861) * Xout[54] +
```

```
    (float)(-1.9023379) * Xout[55] + (float)(-1.0899962) * Xout[56] +
    (float)(-1.1103041) * Xout[57] + (float)(0.43716505) * Xout[58] +
    (float)(0.47580844) * Xout[59] + (float)(1.3565735) * Xout[108] +
    (float)(0.12094656) * Xout[109];
Xout[103] = 1.0 / (1.0 + exp( -Xout[103] ));

/* Generating code for PE 44 in layer 4 */
Xout[104]  = (float)(1.0193326e-005) * Xout[51] +
    (float)(-1.234505e-005) * Xout[54] + (float)(1.1479333e-005) * Xout[55]
     + (float)(1.2623273e-005) * Xout[57] +
    (float)(1.3918387e-005) * Xout[109];
Xout[104] = 1.0 / (1.0 + exp( -Xout[104] ));

/* Generating code for PE 45 in layer 4 */
Xout[105]  = (float)(-0.00044823103) + (float)(9.0591944e-005) * Xout[50]
     + (float)(1.8353496e-005) * Xout[51] +
    (float)(-7.1812625e-005) * Xout[52] + (float)(0.00018278712) * Xout[53]
     + (float)(6.3663509e-005) * Xout[54] +
    (float)(0.00014655043) * Xout[55] + (float)(2.7393486e-005) * Xout[56]
     + (float)(0.0001975622) * Xout[58] +
    (float)(7.1318253e-005) * Xout[59] +
    (float)(9.5640149e-005) * Xout[108] +
    (float)(4.9502469e-005) * Xout[109];
Xout[105] = 1.0 / (1.0 + exp( -Xout[105] ));

/* Generating code for PE 46 in layer 4 */
Xout[106]  = (float)(0.00023040116) + (float)(-5.2382205e-005) * Xout[50]
     + (float)(-2.9611108e-005) * Xout[51] +
    (float)(4.1685456e-005) * Xout[52] + (float)(-0.00010552297) * Xout[53]
     + (float)(-1.0615638e-005) * Xout[54] +
    (float)(-8.3810912e-005) * Xout[55] +
    (float)(-2.4085066e-005) * Xout[57] +
    (float)(-8.1487422e-005) * Xout[58] +
    (float)(-3.4434139e-005) * Xout[59] +
    (float)(-2.685517e-005) * Xout[108] +
    (float)(-5.0365397e-005) * Xout[109];
Xout[106] = 1.0 / (1.0 + exp( -Xout[106] ));

/* Generating code for PE 47 in layer 4 */
Xout[107]  = (float)(-3.4915927e-005) + (float)(1.6623082e-005) * Xout[50]
     + (float)(2.2298404e-005) * Xout[51] +
    (float)(-1.0412085e-005) * Xout[52] +
    (float)(2.3294422e-005) * Xout[53] +
    (float)(-2.0708705e-005) * Xout[54] +
    (float)(2.9503291e-005) * Xout[55] + (float)(2.6054226e-005) * Xout[57]
     + (float)(2.9213143e-005) * Xout[109];
Xout[107] = 1.0 / (1.0 + exp( -Xout[107] ));

/* De-scale and write output from network */
Yout[0] = Xout[60] * (0.5) + (4.29);
Yout[1] = Xout[61] * (2.1) + (0.95999998);
Yout[2] = Xout[62] * (0.08) + (0.029999999);
Yout[3] = Xout[63] * (5.6799984) + (28.530001);
Yout[4] = Xout[64] * (1.0599999) + (4.3099999);
Yout[5] = Xout[65] * (2.03) + (1.02);
Yout[6] = Xout[66] * (0.059999999) + (0.050000001);
Yout[7] = Xout[67] * (5.0599995) + (25.82);
Yout[8] = Xout[68] * (2.54) + (4.9400001);
Yout[9] = Xout[69] * (1.9300001) + (1.3);
Yout[10] = Xout[70] * (0.059999999) + (0.059999999);
Yout[11] = Xout[71] * (3.460001) + (22.82);
Yout[12] = Xout[72] * (4.6799994) + (4.3000002);
Yout[13] = Xout[73] * (2.9800003) + (2.3);
Yout[14] = Xout[74] * (0.02) + (0.0099999998);
Yout[15] = Xout[75] * (4.0699997) + (23.559999);
Yout[16] = Xout[76] * (0.30000019) + (4.4099998);
Yout[17] = Xout[77] * (2.3799999) + (0.92000002);
Yout[18] = Xout[78] * (0.089999998) + (0.029999999);
Yout[19] = Xout[79] * (7.5799999) + (26.969999);
Yout[20] = Xout[80] * (0.96000004) + (4.3800001);
Yout[21] = Xout[81] * (1.6000001) + (1.04);
Yout[22] = Xout[82] * (0.059999999) + (0.050000001);
Yout[23] = Xout[83] * (5.7999992) + (24.870001);
Yout[24] = Xout[84] * (3.25) + (5.3800001);
Yout[25] = Xout[85] * (2.0100001) + (1.17);
Yout[26] = Xout[86] * (0.059999999) + (0.059999999);
Yout[27] = Xout[87] * (3.9899998) + (22.27);
Yout[28] = Xout[88];
Yout[29] = Xout[89];
Yout[30] = Xout[90];
Yout[31] = Xout[91];
Yout[32] = Xout[92] * (0.31999969) + (4.3400002);
Yout[33] = Xout[93] * (1.8200001) + (0.87);
Yout[34] = Xout[94] * (0.099999996) + (0.029999999);
```

```
    Yout[35] = Xout[95] * (6.5899982) + (28.120001);
    Yout[36] = Xout[96] * (0.9000001) + (4.3299999);
    Yout[37] = Xout[97] * (2.03) + (1);
    Yout[38] = Xout[98] * (0.059999999) + (0.050000001);
    Yout[39] = Xout[99] * (4.8499985) + (26.120001);
    Yout[40] = Xout[100] * (3.79) + (5.4200001);
    Yout[41] = Xout[101] * (2.7800001) + (1.37);
    Yout[42] = Xout[102] * (0.050000001) + (0.050000001);
    Yout[43] = Xout[103] * (4.6499996) + (23.700001);
    Yout[44] = Xout[104];
    Yout[45] = Xout[105];
    Yout[46] = Xout[106];
    Yout[47] = Xout[107];
    return( 0 );
}
```