SHORT-TERM CONGESTION PREDICTION ON FREEWAYS

Giovanni Huisken[*] and Frans Tillema
Centre for Transport Studies, Department of Civil Engineering, University of Twente,
P.O. Box 217, 7500 AE, Enschede, The Netherlands
Phone: +31-53-489 2543, Fax: +31-53-489 4040, Email: g.huisken@ctw.utwente.nl

Abstract

The paper describes a comparative analysis on several methods (the naïve method, multi-linear regression, time series analysis, multi-layer feedforward neural networks, radial basis function neural networks, Elman neural networks, self-organising map neural networks, fuzzy logic) that can be used to come to short-term congestion prediction. The field data sets that were used were acquired on two locations during the morning and evening peak periods. The research results show that both the multi-layer feedforward and the Elman neural networks outperform the other methods. The paper concludes with remarks on the importance to have detectors upstream of the prediction point and to keep in mind what kind of congestion has to be predicted.

Keywords: Congestion prediction; Time series analysis; Neural networks; Fuzzy logic; Freeways
Topic area: C3 Traffic Control

1. Introduction

Dynamic Traffic Management (DTM) is regarded as a highly effective tool to fight congestion, especially within the short-term and middle-long-term range. Nowadays, the deployment of DTM-measures is predominantly *reactive*. However, by bringing DTM-measures *proactive* into action the effectiveness could be increased. This in turn can only be done (effectively, at least) if we have reliable information regarding the near-future traffic state, especially whether or not it is going to be congested. In short: there is need for short-term congestion prediction.

This paper describes a comparative analysis of several methods that can be used to come to short-term congestion prediction. All used methods can be described as 'data driven', i.e. the model development is largely dependent on the acquired data sets. The models are tested on morning and evening peak data sets acquired on two different locations. However, we will not only analyse the performances of the methods, we will also reflect on lessons learned through this research.

1.1 State-of-the-art

If we check the literature we can find very few articles that have congestion prediction as research subject. One of the exceptions is an article by Dougherty *et al.* (1993), however, in this paper only one method (Artificial Neural Networks – ANNs) was used. Exceptions to the above are a few studies that compare multi linear regression, time series analysis, fuzzy logic and artificial neural networks (Huisken, 2000; Huisken and Coffa, 2000; Huisken and van Maarseveen, 2000).

There is, however, a large body of research on short-term traffic flow prediction. Subject of research is usually the prediction of traffic intensities; see e.g. Fahgri and Hua

[*] Current affiliation: MuConsult, P.O. Box 2054, 3800 CB Amersfoort, The Netherlands.
Phone: +31-33-465 5054, Fax: +31-33-461 4021, Email: g.huisken@muconsult.nl

(1992), Smith and Demetsky (1995), Florio and Mussone (1996), Ho and Ioannou (1996), Van der Voort *et al.* (1996), Dia (2001) and Yin *et al.* (2002). The many papers in this field have one thing in common: if comparisons are made between methods the number of methods is usually limited to two. Then again, some exceptions can be found (Huisken, 2001; Huisken, 2003; Huisken and van Maarseveen, *under review*).

Another related area of research is that of short-term travel time prediction. Again, time series analysis and artificial neural networks seem to be the most favourite choices of method (You and Kim, 2000; Van Lint *et al.*, 2002; Huisken and Van Berkum, 2003).

The above-referenced research shows that artificial neural networks (i.e. the *multi-layer-feedforward* neural networks) gave predominantly the best results with respect to congestion prediction as well as traffic intensity prediction. However, no studies on multiple locations could be found.

1.3 Outline

The next section gives a condensed overview of the used methods and is followed by a section that describes the data set acquisition sites. The section thereafter informs on the model development processes. The paper continues with a section that contains the comparative analysis, while the last section ends with the conclusions of the research.

2. Methods

In this section the used methods shall be described very briefly. There are references included that give more comprehensive information for those who are interested.

2.1 The naïve method

The naïve method is not a sophisticated prediction method; it can better be described as a 'do-nothing-scenario' method. So, this rather naïve method assumes that during the look-ahead period (i.e. the prediction horizon) the state of the (traffic) system (with regard to congestion) is stationary. In other words: this method predicts that observation Y at time t will remain the same during the look-ahead period and therefore is equal to Y at time $t = T$, with T being the prediction horizon. Mathematically the naïve method can be noted as:

$$Y_{t+T} = Y_t \qquad (1)$$

2.2 Multi-linear regression (MLR)

Multi-linear regression (Bickel en Jackson, 1977) is a fairly simple method to describe observations Y that are linearly depending on variables θ and white noise e with mean zero and variance σ^2 (Gaussian disturbances) and can be mathematically noted as:

$$Y_i = \theta_0 + c_{1i} \cdot \theta_1 + c_{2i} \cdot \theta_2 + \ldots c_{ki} \cdot \theta_k + v_i \qquad (2)$$

If we set the noise to zero we can rewrite (2) and c can be found by:

$$c = \left(\theta^T \cdot \theta\right)^{-1} \cdot \theta^T \cdot Y \qquad (3)$$

With the parameters c determined, they can then be used to estimate new observations Y given the new variables θ. MLR is the most common regression technique and is usually used to model linear systems.

2.3 Auto regression moving average (ARMA) time series analysis

The Auto Regression Moving Average (ARMA) time series analysis method (Box and Jenkins, 1976) is widely used as a time dependent prediction method. Given F observations X_0, X_1, …, X_{F-1}, ARMA(f, g) time series processes can be written in the form:

$$X_t = \mu + a_1 X_{t-1} + a_2 X_{t-2} + \cdots + a_f X_{t-f} + b_0 \varepsilon_t + b_1 \varepsilon_{t-1} + \cdots + b_g \varepsilon_{t-g} \qquad (4)$$

for $t = f, f+1, \ldots$ and with ε is white noise with mean zero and variance σ^2.

The parameters μ, a_1, a_2, …, a_f, b_0, b_1, …, b_g, and σ^2 can usually be estimated by deriving least squares and maximum likelihood estimators. Formula (4) can be rewritten as:

$$RX = M\varepsilon + PX^0 + \mu I \quad , \quad X = \begin{bmatrix} X_f \\ X_{f+1} \\ \cdots \\ X_{F-1} \end{bmatrix} \quad X^0 = \begin{bmatrix} X_0 \\ X_1 \\ \cdots \\ X_{f-1} \end{bmatrix} \quad \varepsilon = \begin{bmatrix} \varepsilon_{f-g} \\ \varepsilon_{f-g+1} \\ \cdots \\ \varepsilon_{F-1} \end{bmatrix} \qquad (5)$$

with R, M, and P representing matrices containing the a and b parameters and I being the unity vector we can minimise (5) to obtain least squares estimation and maximise (5) to obtain maximum likelihood estimation. ARMA time series analysis is usually used to model processes that can be measured consistently in time.

2.4 Multi-layer feedforward (MLF) neural networks

Artificial neural networks (ANNs) are based upon biological neural networks - like the human brain - by mimicking their architectural structure and information processing in a simplified manner. They both consist of building blocks or processing elements called neurons that are highly interconnected making the networks parallel information-processing systems. The architecture of ANNs is usually made up of several layers; each layer containing neurons. The neurons have a transfer function that gives them specific characteristics. Although the ANNs are rudimentary imitations of biological ones, they are to some extend capable of tasks such as pattern recognition, perception and motor control, which are considered poorly performed and highly processor time inefficient by conventional linear processing, whereas they seem to be done with ease by e.g. the human brain. Neural networks are also known to be robust and to have the capability to capture highly non-linear mappings between input and output (Haykin, 1994).

Muli-Layer Feedforward (MLF) neural networks are so-called ‘supervised’ ANNs meaning that during calibration (the ‘learning process’) every time that an input is presented (a ‘pattern’) also a desired output is provided (the ‘target’). The neurons are equipped with a sigmoid transfer function and ‘weights’ that are adaptable. If the ANN-generated output deviates from the target this deviation (‘error’) acts as a measure for weight adjustment. All weights are changed in such a manner (the ‘learning rule’) that the total of all presented inputs generates a smaller summed error during the next iteration. MLF neural networks are by far the most used ANNs.

2.5 Radial basis function (RBF) neural networks

Radial Basis Function (RBF) neural networks (Powell, 1988) are supervised neural networks that seem similar to MLF neural networks. The difference, however, can be found in the construction of the usually three-layered network. The input layer is made up of neurons with a linear transfer function. The second layer (the hidden layer) which has to be of a high enough dimension has a different purpose than that of the MLF neural networks; this can be seen in that the transfer function is not sigmoid but Gaussian. The output layer supplies the response of the network to the activation patterns applied to the input layer. RBF networks are most often used to deal with approximation problems in a multidimensional space.

2.6 Elman neural networks

Elman neural networks (Elman, 1990) are partial recurrent neural networks, also known as time-space neural networks, where information coming from neurons of the hidden layer serve as additional input in the next time step, the so-called ‘context’. Due to the recurrent character of the network it can be translated as networks that use information from the ‘past’ - if vectors are put in as time step vectors. Their learning rules are somewhat similar to the ones of the ANNs described before.

2.7 Self organising map (SOM) neural networks

Self-Organising Map (SOM) neural networks (Kohonen, 1997) are unsupervised neural network systems that intend to optimise their free parameters according to statistical regularities of the input (training) data and map these input vectors onto a two-dimensional (usually either rectangular or hexagonal) lattice structure. When input is presented to the network all neurons of the lattice structure are stimulated and the neuron with the biggest activation is declared 'winner'. The weights between the input neurons and the winning neuron are strengthened, as well as, but not as strong as, the weights connecting the input neurons to the neighbouring neurons. After the network is tuned and new input data are offered the data are mapped onto the lattice area that has the most statistical similarity to the training data. Self-Organising Maps are also known as Kohonen maps and are most often used to deal with classification problems.

2.8 Fuzzy logic

Fuzzy Logic (FL) can be described as ‘computing with words’ (Zadeh, 1965) and is suited for dealing with complex optimisation problems with many constraints and objectives, with unclear input information and vague decision criteria. This category contains many problems in the fields of traffic and transportation (Teodorović, 1999).

Fuzzy Set Theory, as it is often also called, is an extension of ordinary set theory, i.e. elements are not just ‘member’ or ‘no member’ of a certain set but can have a degree of membership (between 0 and 1). A fuzzy system consists of fuzzification, inference and defuzzification.

The fuzzification process (figure 1) is designed to assign degrees of membership (μ) to crisp input values (x_0, y_0) by means of lookup in one or several membership functions (A_1, A_2, B_1, B_2) These membership functions correspond with linguistic terms that apply to input variables and should be partially overlapping and sufficiently wide to allow for noise in measurements. Not only input but also output variables are fuzzified (C_1, C_2).

The inference method uses a fuzzy rule base (or knowledge base as it is sometimes called) that is made up of logical operations and is based on expert opinions, operator experience, and/or system knowledge. The rules are given according to the following format:

IF <premise 1> AND/OR <premise 2> AND/OR <premise 3> … THEN <consequence>

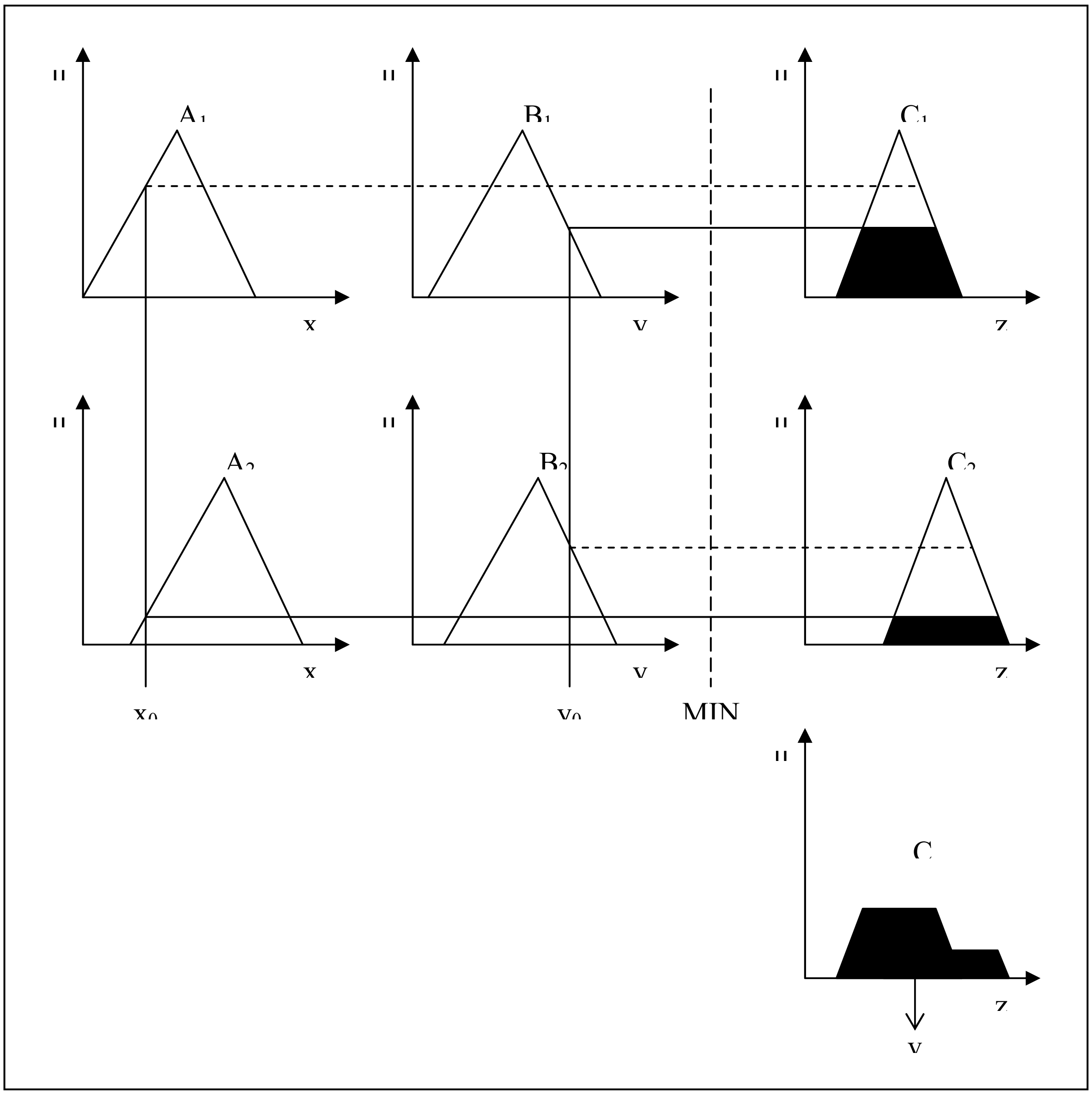

Figure1: Fuzzy logic process

Because a fuzzy rule based system consists of a set of fuzzy rules with partially overlapping conditions, a particular input to the system often triggers multiple fuzzy rules. Therefore a method is needed to combine the inference results of these rules. This is accomplished typically by mapping all fuzzy conclusions onto one variable (z).

Finally, the obtained fuzzy output variables need to be translated into a crisp output value. This process is called defuzzyfication and is established by using a method, e.g. the centre of gravity approach, which allocates a crisp value to the final outcome (v).

FL is used extensively as a control method in various applications, e.g. for determining traffic light green times (Niittymäki, 2001) or motor control in photo cameras.

3. Data acquisition

Field data sets were gathered on two locations: around cloverleaf Beekbergen (motorways A1/A50 – figure 2) and cloverleaf Hoevelaken (motorways A1/A28 – figure 3).

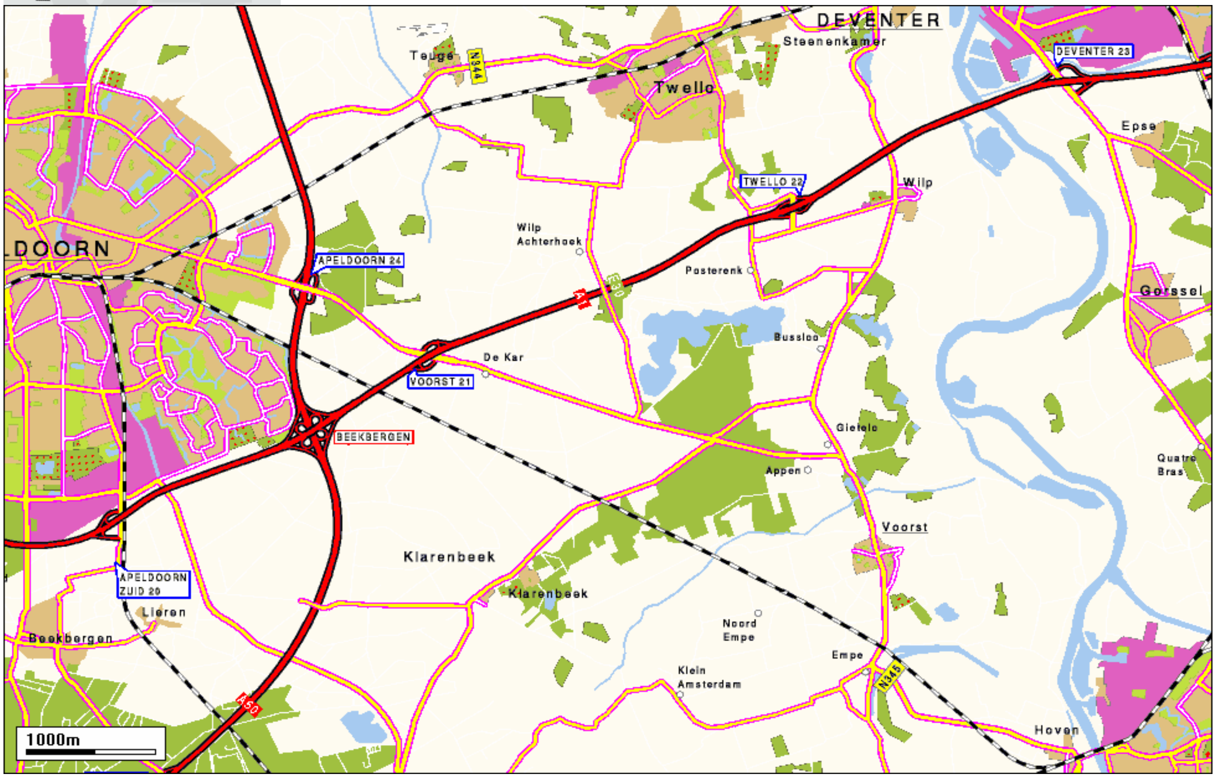

Figure 2: Data acquisition location Beekbergen

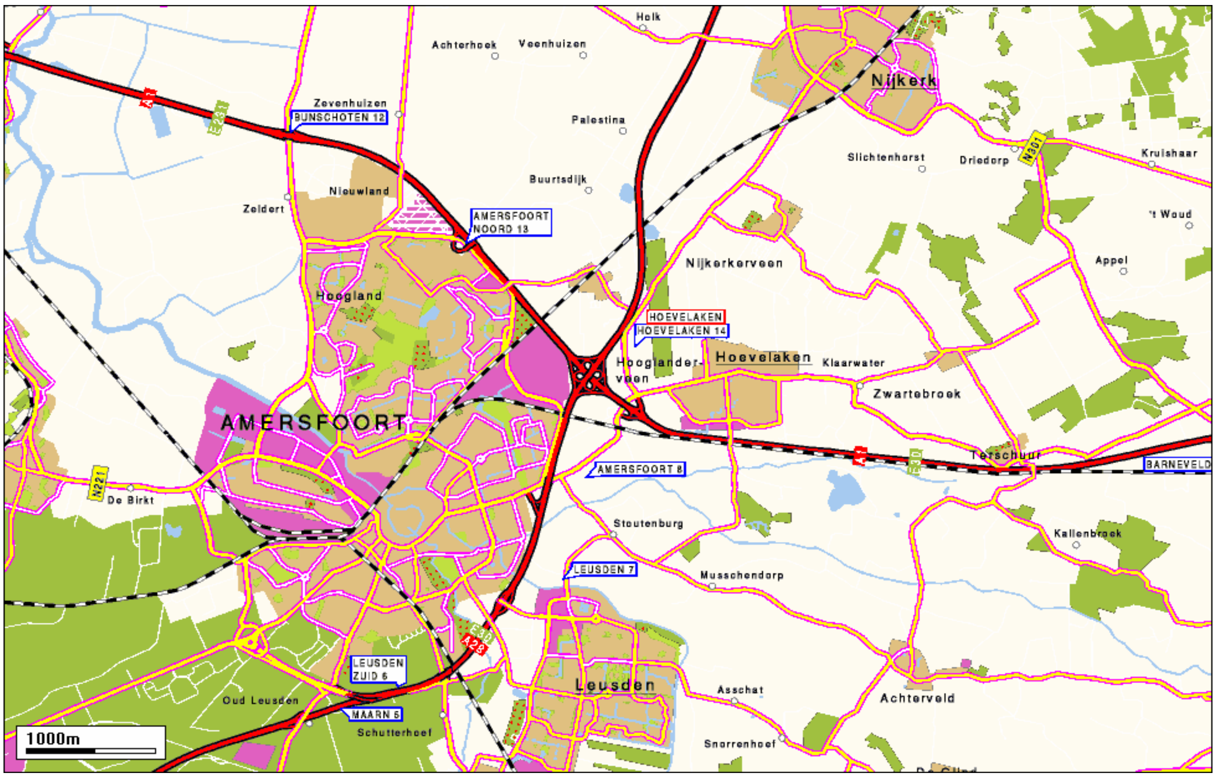

Figure 3: Data acquisition location Hoevelaken

Data was collected through the MoniCa – Monitoring Casco – system by means of dual induction loops during the month May of the year 2001. The data sets contain 1-minute aggregated data of intensity, mean speed, standard deviation of mean speed, occupancy, a reliability indicator, and a congestion indicator. The first three parameters are given in three categories: vehicles with a length up to 5.1 meters, vehicles with a length between 5.1 and 12.5 meters and vehicles longer than 12.5 meters. De congestion indicator flag is turned on when 5 vehicles subsequently pass a detector with a speed below 35 km/h and the flag is turned off again when 5 vehicles pass with a speed above 50 km/h.

4. Pre-processing

Obviously, we want to assess the models based on the location were the data was acquired. This means that we have 2 data sets: the Beekbergen data set and the Hoevelaken data set. For every location the morning and evening peak bottleneck detectors ('target detectors') were identified (being the detectors that usually were the first ones to detect congestion). Since we are interested in predicting congestion it makes sense to remove periods that are likely to not include congestion. Also, there is the reliability issue. Not all dual loop detectors report with 100% reliability all the time. Therefore unreliable data that can be identified by the reliability indicator has to be removed and substituted with the last known reliable data.

As was mentioned before in section 1, all prediction methods that are used are data driven. The implication of this is that we need the data to calibrate the models. However, we also need the data to validate the models. This can cause problems because we have to decide on which part of the data set will be used to calibrate the models and which part will be used to validate the models. A proper solution to this problem is to use cross-validation, i.e. we divide the data set in sub-sets and use a part to calibrate the model. Then we use the remaining part to validate the model. After this exercise we follow the same routine but then with another sub-set as validation set and repeat this until all sub-sets are used as validation set. The average of all sub-set validations then is used for establishing the performance of the models.

So, in order to perform cross-validation we need equally sized sub-sets. Since the data was acquired during the month of May of 2001 we can divide the set in four one-week sub-sets. First, all weekend days were removed. Since we were left with 4 Mondays, 5 Tuesdays, 5 Wednesdays, 5 Thursdays, and 4 Fridays the Tuesday, Wednesday, and Thursday that contained the most unreliable data were thrown out. This left us with 4 sets of one working week. Each day's peak period was filtered out; a peak period is defined as 300 minutes (morning peak from 06:00 hours – 10:59 hours and the evening peak from 15:00 hours – 19:59 hours). This resulted in 4 equally sized data sub-sets (1 working week per morning/evening peak period).

In order to build data sets that contain data that is relevant we need to identify the bottlenecks, i.e. the dual loop detector that usually will be the first one to detect congestion. The congestion indicator data reported by this detector will then be used as target data. We also need the identification of this detector to select relevant detectors in the sense that they are located either upstream or downstream in the vicinity of the bottleneck detector. In that way we can filter out data coming from detectors that are located on a lane on which traffic is driving in the opposite direction.

Finally, we also have to normalise data (to the [0, 1] domain; this procedure is necessary for ANNs to accept input values) so that every model receives the same input data during the calibration and evaluation process.

5. Model development

The variables present in each data sub-set and thus the input variables of the models are: intensity, mean speed, standard deviation of the mean speed (all variables for each vehicle category) and occupancy. These variables are provided by each dual induction loop. This results in (# of loops) * 10 = # of input variables. Obviously, for ARMA time series analysis temporal data coming from the target detector were used as input. For the remaining methods spatial data was used as input.

As output variable we used the binary congestion indicator ('congestion' or 'no congestion') gathered by the target detectors and it was shifted for 5, 10, 15, 20, 25 and 30 minutes in time to estimate the prediction performances of the models at these prediction horizons.

We will now describe the model development process for the morning peak data set of Beekbergen. The models for the remaining 3 peak periods have been developed in an analogue manner.

5.1 The naïve model

The model development of the data set into naïve models is rather simple. We only use data collected from the bottleneck detector. To make it even simpler, we only use the congestion indicator data. Data is then normalised by translation 'no congestion' into 0 and 'congestion' into 1. Each data sub-set now holds 1500 numbers and can be regarded as a vector containing 1500 units. The index number is linked to a particular minute of a working day (location 1 is Monday 6:00 pm). By shifting the index numbers with T spaces to the left we obtain the target vector with a prediction horizon of T minutes. An example:

Origin = [0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, …]

The index is shifted by 5 spaces and then becomes

Target = [0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …]

which is now the target vector with a prediction horizon of 5 minutes.

If we now subtract the Origin from the Target and sum the absolute value of the resulting vector we find the number of minutes that are in error when Origin is used as a model to predict congestion of the bottleneck with a 5- minute horizon. This process can be done for any number of minutes keeping in mind that the prediction horizon should not exceed the shortest 'no congestion' period after 6:00 and before 11:00 in the morning. This procedure is applied to every week sub-set.

5.2 The MLR model

Input data for the MLR model is collected through all relevant dual induction loops, i.e. all induction loops upstream of the bottleneck and two detectors downstream, just in case of back spilled congestion. For the morning peak period of the Beekbergen location this means data gathered by 23 detectors. This results in 1-minute time slices of 23 detectors * 10 variables = 230 input variables. Since there is 1 output variable (the congestion indicator of the bottleneck detector) we obtain four input matrices of 230 * 1500 [500 minutes peak period for 5 week days] elements and four target matrices of 1 * 1500 elements representing four work day weeks.

Three input matrices are combined (placed behind each other) to a resulting 230 * 4500 matrix and the accompanying target matrices undergo the same procedure resulting in a 1 * 4500 matrix. These matrices are then used according to formula (3) and so we obtain the resulting matrix *c* that holds the multiplier indices that is used together with the remaining input matrix (or evaluation matrix). If we then multiply the evaluation matrix with *c* the result should ideally be the same as the remaining target matrix. The resulting matrix is subtracted (element by element) from the target matrix and this produces an error matrix that is used to determine the model's error. This procedure is then repeated four times (cross-evaluation) for each input matrix to serve as evaluation matrix.

5.3 The ARMA time series analysis model

Input data for the ARMA(f, g) model is collected through the bottleneck detector. The ARMA model uses spatial data from the bottleneck detector to calibrate the model, by making use of formula (5). The identification of the lag and order index that will provide the best results is a time consuming task. In order to come to the identification of the proper lag and order index we constructed a data test set that consisted of random chosen vectors (week minutes) of data coming from every one of the data input sets and accompanying data target set (with several prediction horizons). We used this test set to assess the calibrated ARMA models resulting in figure 4 and thus revealing the optimum at order 8 and lag 0. The elements of the test set were left out during the evaluation process described in the next section.

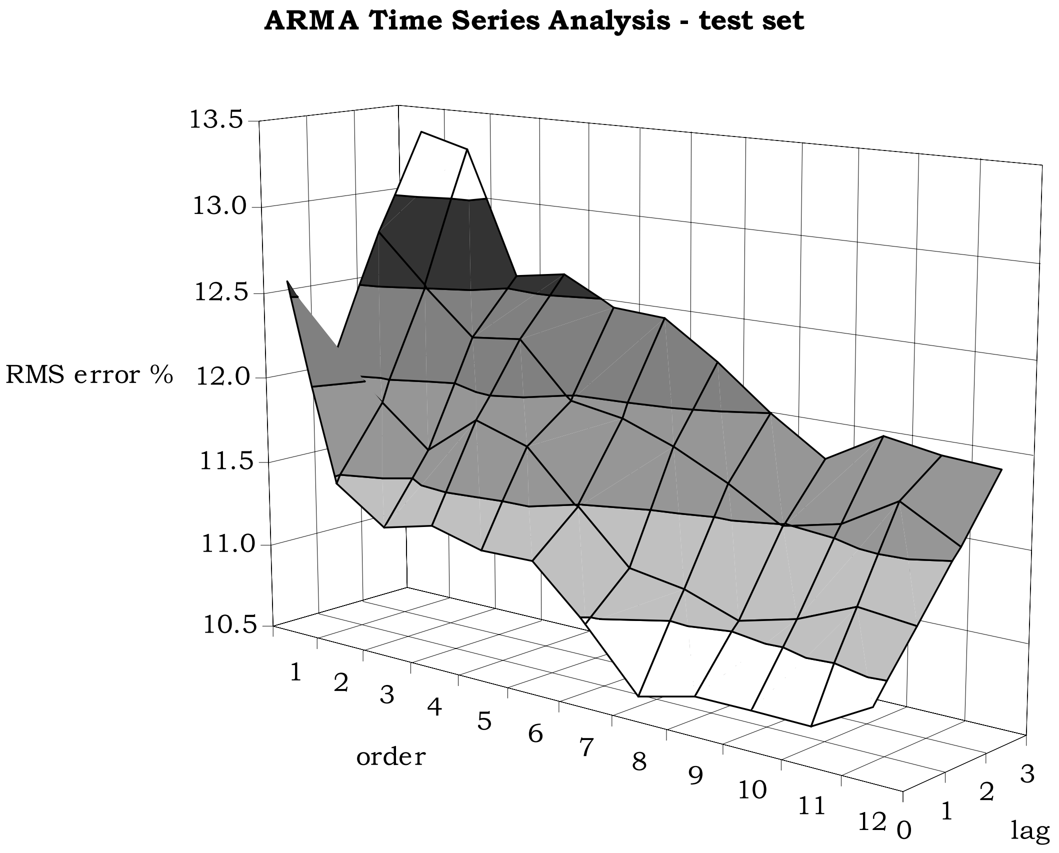

Figure 4: ARMA time series analysis - RMS error percentage of the test set depending on order and lag

5.4 The MLF model

The input data subsets are also used to evaluate the MLF method. The model is build by using the spatial input variables (23 induction loop detectors * 10 variables = 230 variables). These 230 input variables are mapped onto the 1 output variable, the target. The target is the congestion indicator of the bottleneck detector shifted in time as described in paragraph 5.1. To find the optimum number of learning epochs and the optimum neural network architecture (i.e. the optimum number of hidden neurons) we used the vectors that hold the same target elements as the test set as described in the previous paragraph. By varying these 2 parameters we obtained figure 5 showing that the optimum was found at 30 learning epochs with a 230-6-1 neural network architecture. Again, the elements of the test set were left out during the evaluation process.

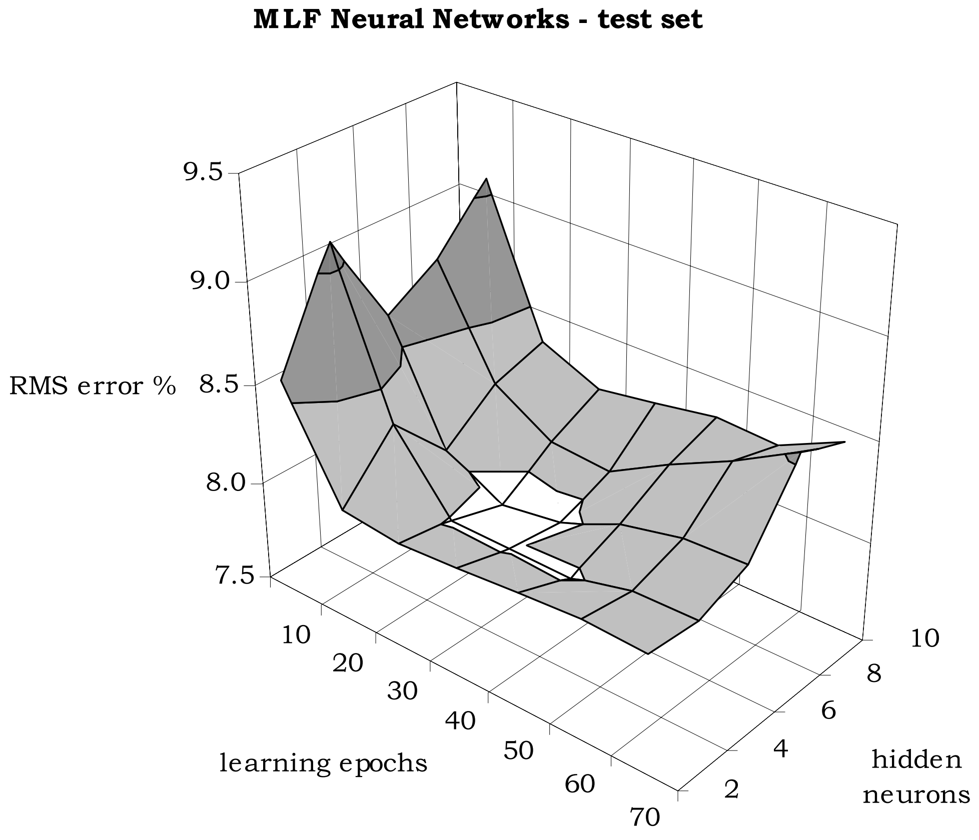

Figure 5: MLF neural networks - RMS error percentage of the test set depending on learning epochs and hidden neurons

5.5 The RBF model

The model development of the RBF model is identical to that of the MLF model. The test set model runs resulted in figure 6 and gave as optimum 70 learning epochs at an 230-6-1 neural network architecture.

5.6 The Elman model

The model development of the Elman model is identical to that of the previous artificial neural network model. The test set model runs with the Elman models showed an optimum at 80 learning epochs by using a 230-2-1 neural network architecture, as can be seen in figure 7.

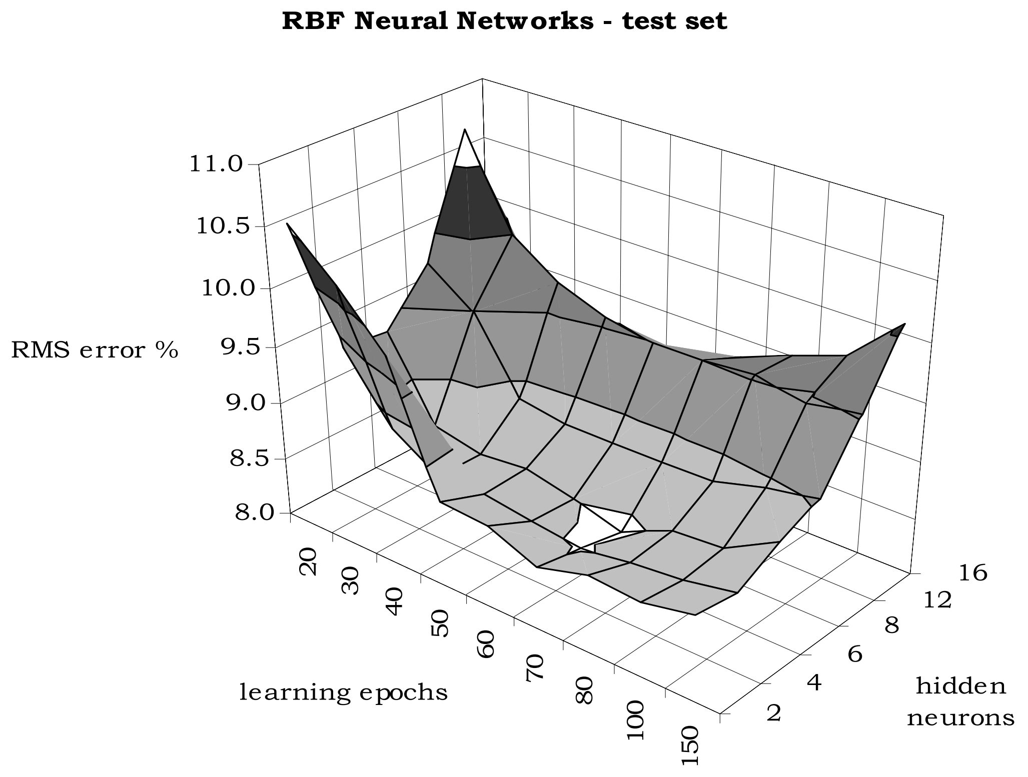

Figure 6: RBF neural networks - RMS error percentage of the test set depending on learning epochs and hidden neurons

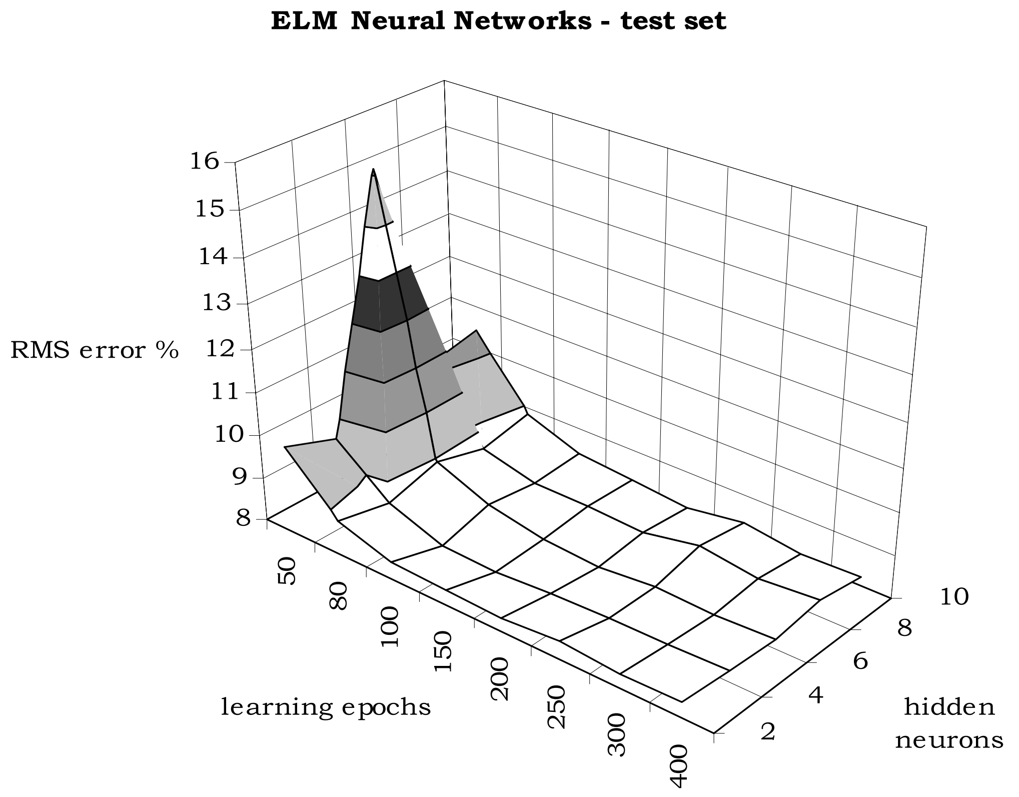

Figure 7: ELM neural networks - RMS error percentage of the test set depending on learning epochs and hidden neurons

5.7 The SOM model

The model development of a SOM model is completely different from the previous procedures; because the learning phase is unsupervised, the network is built on statistical clustering that is hidden within the data set. Therefore, one can only use the targets to establish how well the model is able to see ‘congestion’ as one particular cluster. Since there are many parameters of how to develop such a model (number of categories, one or two-dimensional [hexagonal or square] etc.) we will only provide the model that seemed to approach ‘congestion’ as good as possible within one cluster. The architecture of this SOM model was 230-10 neurons.

5.8 The fuzzy logic model

We chose to use the ANFIS (Adaptive-Network-based Fuzzy Inference System) model (Jang, 1993) that uses adaptive rules to come to the optimum shape of the membership functions. The drawback of this method is that computation time increases exponentially with increasing number of membership functions. Practical restraints resulted in the selection of 4 membership functions as a maximum; we selected the 3 variables of the first vehicle category plus occupancy. The model variable that was left to be optimised was the number of learning epochs and again we used the test set described in 5.3. Several runs resulted into figure 8 where the optimum was identified at 125 learning epochs.

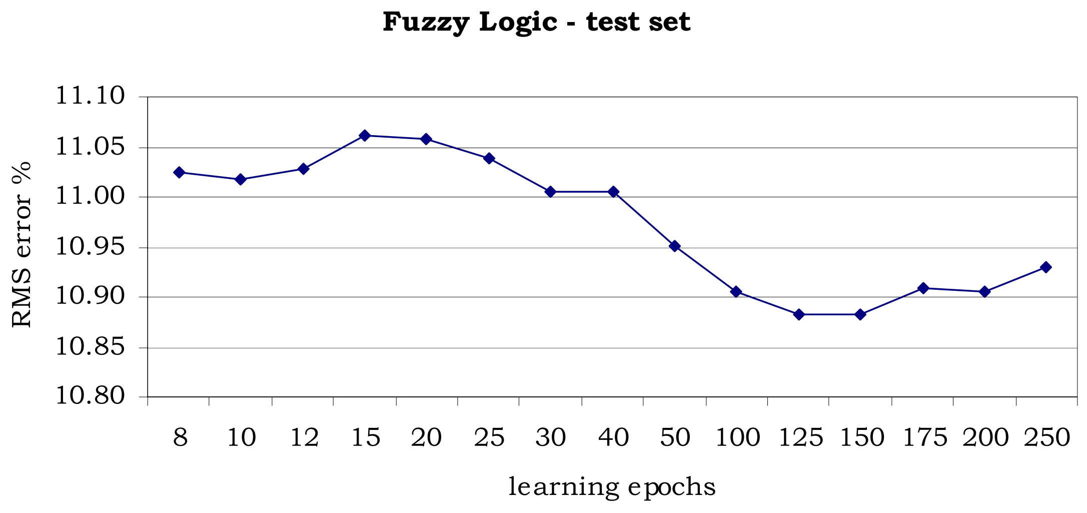

Figure 8: Fuzzy Logic - RMS error percentage of the test set depending on learning epochs

6. Analysis

The performances of the methods were determined by comparing the through the models generated output with the congestion indicator of the target detector. When errors were made they were either classified as false alarm (predicting ‘congestion’ when in fact there was no congestion) or error (predicting ‘no congestion’ when there was congestion).

For validation cross-evaluation was used; the method’s performances were evaluated with 1 data sub-set per peak period per location while the remaining 3 data sub-sets per peak period per location were used to calibrate or train/test the model parameters. This procedure was executed for each data sub-set and the so-obtained results were averaged giving the final performances measured in error percentages over time, i.e. (# minutes / total minutes) * 100%.

Figures 4 – 7 display the results of the model performances. Each figure shows the error percentage per method that is grouped per prediction horizon.

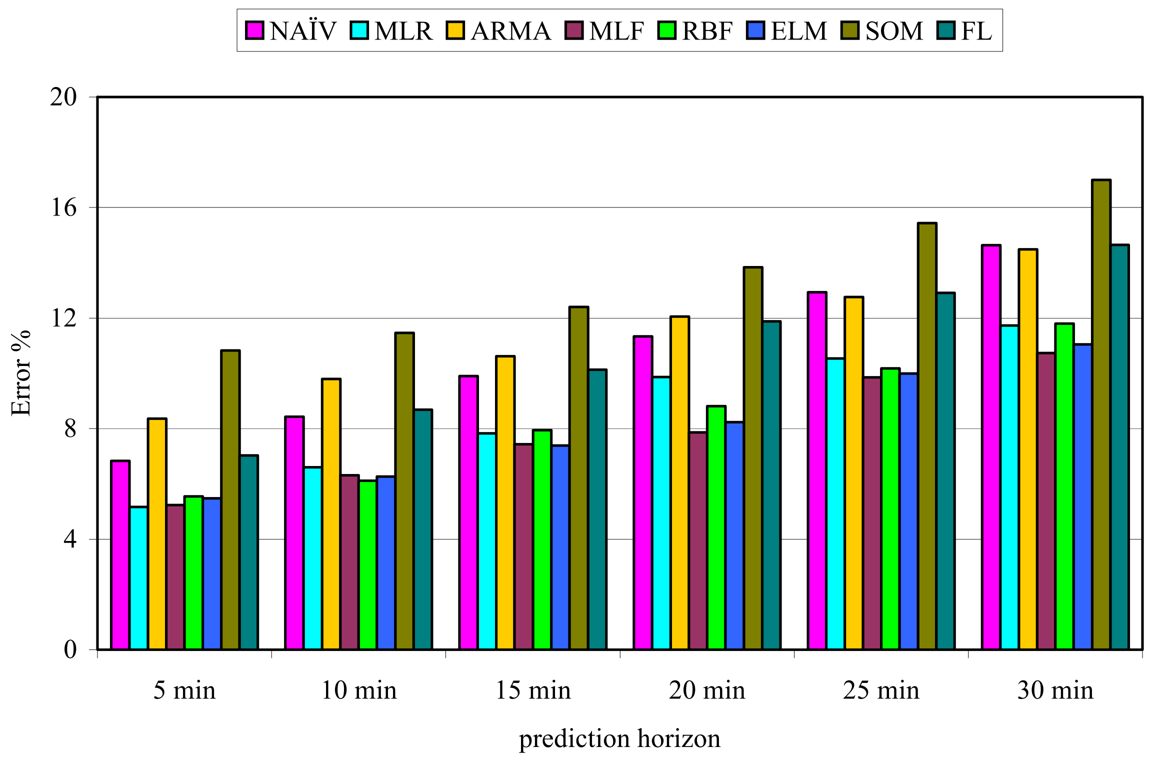

Figure 9: Error % per method per prediction horizon during the morning peak period on location Beekbergen

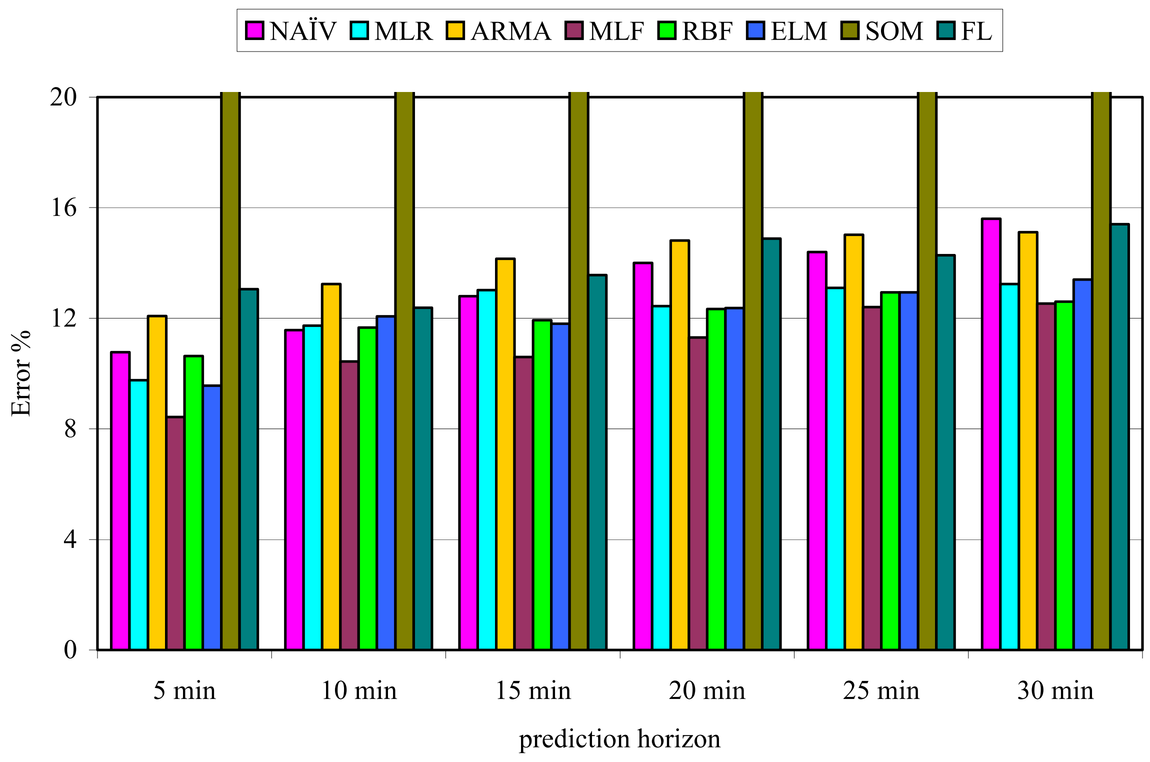

Figure 10: Error % per method per prediction horizon during the evening peak period on location Beekbergen

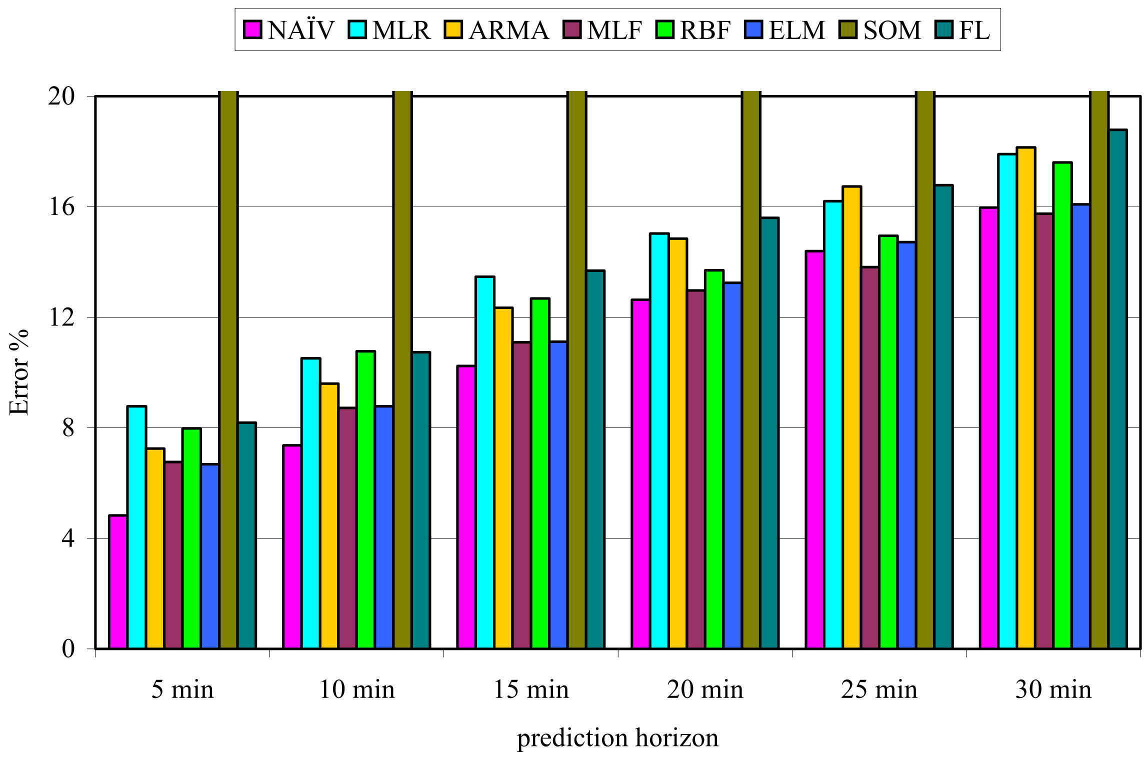

Figure 11: Error % per method per prediction horizon during the morning peak period on location Hoevelaken

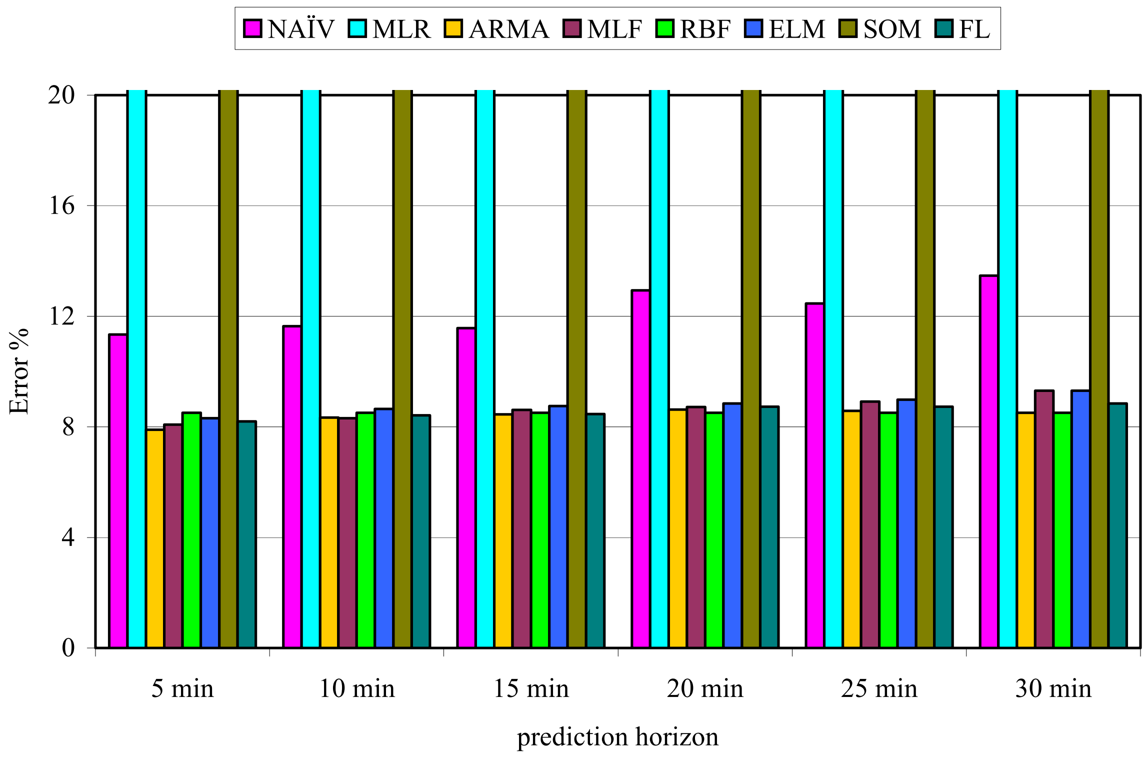

Figure 12: Error % per method per prediction horizon during the evening peak period on location Hoevelaken

The morning peak figures show that the errors increase with increasing prediction horizon; this is logical keeping in mind that the models have to predict congestion on the minute accurate. The MLF neural networks and the Elman neural networks produce the best results under these circumstances.

What is the matter with the evening peak results? Closer inspection of the Beekbergen evening peak reveals that only 2 detectors are present upstream to the target detector. This means that very little information on traffic that has to pass the target detector is available and because of the distance that these 2 detectors lie upstream (about 1½ km) the prediction horizon is very short.

Closer analysis of the Hoevelaken evening peak learns that only roughly 8 % of the time the target detector is congested. This might not be a problem but the severity increases when we see how the congestion is detected: the congestion indicator flips on and off every few minutes. This is due to stop-and-go traffic and this kind of congestion is very difficult to predict within 1-minute accuracy. The models are calibrated with the data set and when they output 'no congestion' all the time the overall value gives the best results, i.e. about 8%. This is supported when we have a look at the 'false alarm' versus 'error' percentages: the 'best' models give almost no 'false alarm' and about the same 8% 'error'.

7. Conclusions

Of the methods that were compared (the naïve method, MLR, ARMA time series analysis, MLF ANNs, RBF ANNs, Elman ANNs, SOM ANNs, FL) the MLF ANNs and Elman ANNs produce the best results. Another conclusion is that in order to come to meaningful congestion prediction it is necessary to have access to information acquired by detectors upstream of the bottleneck detector. It is also important to be aware of what kind of congestion has to be predicted; if it is stop-and-go traffic congestion this has proven to be extremely difficult (at least if one wants to predict congestion with 1-minute accuracy).

In conclusion we have to make some critical notes with respect to the comparison: the ARMA method and the Fuzzy Logic method use different input data then the remaining methods. While the ARMA method uses data from previous time frames ('historical data') the remaining methods also use data from detectors upstream of the bottleneck detector, i.e. they use information relating to vehicles that have not passed the bottleneck detector ('future data'). The Fuzzy Logic method only uses data of the present from the bottleneck detector and not even the complete 10 variables. However, it is inherent to the ARMA method that it makes use of historical data and as far as the Fuzzy Logic ANFIS method is concerned we were bound (for now) by computational restriction.

Acknowlegdements

The authors wish to acknowledge the Transport Research Centre residing under the Dutch Ministry of Transport, Public Works and Water Management for providing the field data.

References

Bickel, P.J. and Jackson, T., 1977. Mathematical Statistics: Basic Ideas and Selected Topics, Holden-Day, Inc., Oakland, CA, USA.

Box, G.E.P. and Jenkins, G.M., 1970. Time Series Analysis, Forecasting and Control, Holden-Day, San Francisco, CA, USA.

Dia, H., 2001. An object-oriented neural network approach to short-term traffic forecasting, European Journal of Operational Research, 131(2) 253 – 261.

Dougherty, M.S., Kirby, H.R. and Boyle, R.D., 1993. The use of neural networks to recognise and predict traffic congestion, Traffic Engineering and Control, 34 (6) 311 – 314.

Elman, J.L., 1990. Finding structure in time, Cognitive Science, 14 179 – 211.

Faghri, A. and Hua, J., 1992. Evaluation of artificial neural network applications in transportation engineering, Transportation Research Record, 1358 71 – 80.

Florio L. and Mussone L., 1996. Neural-network models for classification and forecasting of freeway traffic flow stability, Control Engineering Practice, 4 (2) 153 – 164.

Haykin, S., 1994. Neural networks: a comprehensive foundation, Prentice-Hall.

Ho, F.-S. and Ioannou, P., 1996. Traffic Flow Modeling and Control Using Artificial Neural Networks, IEEE Control Systems, 16 (5) 16 – 26.

Huisken, G., 2000. Short-Term Congestion Forecasting: time series versus fuzzy sets, In: Proceedings South-African Transport Conference - Action in Transport for the new Millenium, 17 - 20 July 2000, Pretoria, South Africa.

Huisken, G., 2001. Short-term forecasting of traffic flow on freeways, In: Proceedings 9th World Conference on Transportation Research, Seoul, South Korea.

Huisken, G., 2003. Soft-Computing Techniques Applied to Short-Term Traffic Flow Forecasting, System Analysis, Modelling, Simulation 43 (2) 165 – 173.

Huisken, G. and Coffa, A., 2000. Short-Term Congestion Prediction: comparing time series with neural networks, IEE Conference Publication 472 60 – 63.

Huisken, G. and Van Berkum, E.C., 2003. A Comparative Analysis of Short-Range Travel Time Prediction Methods, In: Preprints of the 82nd Annual Meeting of the Transportation Research Board, Washington, DC, USA.

Huisken, G. and Van Maarseveen, M.F.A.M., 2000. Congestion prediction on motorways: a comparative analysis, In: Proceedings 7th World Congress on Intelligent Transportation Systems, 6-9 November 2000, Turin, Italy.

Huisken, G. and Van Maarseveen, M.F.A.M., *under review*. A Comparative Analysis of Short-term Traffic Flow Forecasting Methods, Submitted to: IEEE Transactions on Intelligent Transportation Studies.

Jang, J.-S.R., 1993. ANFIS: Adaptive-Network-based Fuzzy Inference Systems, IEEE Transactions on Systems, Man, and Cybernetics 23(3) 665 – 685.

Niittymäki, J., 2001. Installation and experiences of field testing a fuzzy signal controller, European Journal of Operational Research, 131 (2) 273 – 281.

Kohonen, T., 1982. Self-organized formation of topological correct feature maps, Biological Cybernetics, 43 59 – 69.

Powell, M.J.D., 1988. Radial basis function approximations to polynomials, in: Numerical Analysis 1987 Proceedings, pp. 223 – 241, Dundee, UK.

Smith B. L. and Demetsky M. J., 1995, Short-Term Traffic Flow Prediction: Neural Network Approach, Transportation Research Record, 1453 98 – 104.

Teodorović, D., 1999. Fuzzy logic systems for transportation engineering: the state of the art, Transportation Research Part A, 33 (5) 337 – 364.

Van der Voort, M.C., Dougherty, M.S. and Watson, S.M., 1996. Combining Kohonen maps with ARIMA time series models to forecast traffic flow, Transportation Research Part C, 4 (5) 307 – 318.

Van Lint, H., Hoogendoorn, S.P., and Van Zuylen, H.J., 2002. Robust freeway travel time prediction with state-space neural networks, In: Proceedings of the 13th Mini-EURO Conference, Bari, Italy.

Yin, H., Wong, S. C., Xu, J. and Wong, C. K., 2002. Urban traffic flow prediction using a fuzzy-neural approach, Transportation Research Part C, 10 (2) 85 – 98.

You J. and Kim, T.J., 2000. Development and evaluation of a hybrid travel time forecasting model, Transportation Research Part C, 8 (1-6), 231 – 256.

Zadeh, L.A., 1965. Fuzzy sets, Information and Control, 8 338 – 353.