Abstract—Fuzzy Associative Conjuncted Maps (FASCOM) is a fuzzy neural network that represents information by conjuncting fuzzy sets and associates them through a combination of unsupervised and supervised learning. The network first quantizes input and output feature maps using fuzzy sets. They are subsequently conjuncted to form antecedents and consequences, and associated to form fuzzy if-then rules. These associations are learnt through a learning process consisting of three consecutive phases. First, an unsupervised phase initializes based on information density the fuzzy membership functions that partition each feature map. Next, a supervised Hebbian learning phase encodes synaptic weights of the input-output associations. Finally, a supervised error reduction phase fine-tunes the fine-tunes the network and discovers the varying influence of an input dimension across output feature space. FASCOM was benchmarked against other prominent architectures using data taken from three nonlinear data estimation tasks and a real-world road traffic density prediction problem. The promising results compiled show significant improvements over the state-of-the-art for all four data prediction tasks.

I. INTRODUCTION

Neural networks and fuzzy systems represent key methodologies in soft computing [1]. Neural Networks take inspiration from brain functions and connectionist structures to be highly capable in learning and adaptation, while fuzzy systems mimic human-like knowledge representation and reasoning. Each methodology possesses its own share of drawbacks. On one hand, neural networks are typically black boxes and have a problem of being opaque. On the other hand, fuzzy systems face design problems such as determining membership functions, as well as identifying and inferring fuzzy rules.

By integrating neural networks with fuzzy logic, the architecture’s transparency can significantly improve. Conversely, design problems faced by fuzzy systems may be alleviated through brain-inspired techniques, such as self-organization. Thus, the hybridization of neural networks with fuzzy systems results in powerful tools for machine learning and logic inference because their strengths are complimentary and are hence able to maximize the desirable properties of both methodologies. Such hybrid intelligent systems are widely known as neuro-fuzzy systems or fuzzy neural networks (the name used henceforth).

Fuzzy neural networks are generally grouped into two distinctively different paradigms based on how they represent fuzzy if-then rules. The first paradigm is called the linguistic fuzzy model (e.g. Mamdani model [2]), whereby both the antecedent and consequence are represented using fuzzy sets. The second is known as the precise fuzzy model (e.g. TSK model [3]–[5]) in which the antecedent is a fuzzy set while the consequence is expressed as linear equations [6], [7]. Linguistic fuzzy models excel in interpretability while lacking in accuracy, while the inverse is true for precise fuzzy models (see [7], [8] for discussion).

This paper proposes the Fuzzy Associative Conjuncted Maps (FASCOM) architecture, which is a variant of a fuzzy linguistic model, whereby a fuzzy if-then rule $R_k$ is represented in the form of

$$R_k : \text{if } x_1 \text{ is } A_{k,1} \text{ and } \ldots \text{ and } x_I \text{ is } A_{k,I} \text{ then } y_1 \text{ is } C_{k,1} \text{ and } \ldots \text{ and } y_S \text{ is } C_{k,S}$$

(1)

where $X = [x_1, \ldots, x_I]^T$ and $Y = [y_1, \ldots, y_S]^T$ are the input and output vectors respectively, $A_{k,i}, i \in \{1 \ldots I\}$ and $C_{k,s}, s \in \{1 \ldots S\}$ represent linguistic labels of input and output linguistic variables, while $I$ and $S$ are the number of antecedents and consequences respectively. Each rule is weighted by $v_k$ denoting the strength of the $R_k$.

The current research direction in fuzzy neural networks is to learn, modify and infer fuzzy rules based on past experience. This is achieved using either unsupervised and/or supervised learning techniques to identify, learn and adjust fuzzy if-then rules.

In unsupervised learning approaches [9]–[12], algorithms are used to identify fuzzy rules before applying neural network techniques to adjust rules without the need for a priori outputs. However, due to the heavy reliance on the training data, non-representative data may lead to ill defined rules and hence producing an inaccurate models.

For supervised learning approaches [13], [14], rules are identified by mapping input-output data pairs. However, a drawback is that the semantics of the fuzzy neural network remains opaque, contradicting the original conceived objective of combining fuzzy logic with neural networks.

FASCOM is encoded through a learning process consisting of one unsupervised learning phase followed by two consecutive supervised learning phases. The initial phase involves the unsupervised initialization of membership functions of fuzzy sets of data dimensions. This is followed by a supervised Hebbian learning phase to determine the synaptic weights between associated nodes in the network. The final supervised phase fine-tunes the network by reducing the error produced by the system. The entire learning process consisting of the three learning phases will be described in detail in Section III and the contributions of the individual phases to the accuracy of data prediction is discussed in V-A.4.
The following contributions of the proposed architecture are described in this paper:

1) Improving the precision of output prediction by using uniform information density topology over one with a proportional distribution of fuzzy sets in feature space.
2) Using a novel supervised error reduction algorithm to fine-tune the network and discover the varying influence of an input dimension across output space.
3) Improvements in accuracy over the state-of-the-art for a real-world road traffic density prediction problem.

II. THE FASCOM ARCHITECTURE

The FASCOM architecture (Fig. 1) has six layers, with each layer performing a specific fuzzy operation. The inputs and outputs are vectors \( X = [x_1, \ldots, x_i, \ldots, x_J]^T \) and \( Y = [y_1, \ldots, y_k, \ldots, y_S]^T \), where \( I \) and \( S \) denote the number of input and output linguistic variables respectively. In layer 1, \( \text{input linguistic node } IF_i \), represents the \( i \)th input linguistic variable of input \( x_i \). In layer 2, \( \text{input label node } \text{IL}_{i,j} \), represents the \( j \)th input linguistic label of the \( i \)th input linguistic variable. \( \text{IL}_{i,j} \) nodes corresponding to the same input \( x_i \) are grouped in \( \text{input map } \text{IM}_i \). Layer 3 consists of \( \text{antecedent nodes } A_{i,j} \) grouped into \( \text{input conjuncted maps } \text{ICM}_i \). \( A_{k,l} \) represents either a single antecedent that is mapped from an \( \text{IM} \), or multiple antecedents which are a conjunction of two or more \( \text{IL}_{i,j} \) nodes from different \( \text{IM} \). In layer 4, \( \text{consequence node } C_{q,p} \) represents either a single consequence or multiple consequences, and nodes are organized into \( \text{output conjuncted maps} \). Layer 5 consists of \( \text{output maps } \text{OM}_s \), each containing \( \text{output label nodes } \text{OL}_{s,r} \), with each node symbolizing the \( r \)th linguistic label of the \( s \)th output variable. In layer 6, \( \text{output linguistic nodes } \text{OD}_s \), denotes the \( s \)th output linguistic variable of output \( y_s \).

![Diagram of the FASCOM architecture](image-url)

Fig. 1. Six-layered structure of the Fuzzy Associative Conjuncted Maps (FASCOM) architecture.
In the hybrid associative memory, every ICMk is connected to every OCMj via a (ICMk, OCMj) association, each representing a fully connected two-layered heteroassociative network. In a (ICMk, OCMj) network, when \( A_{k,l} \) is linked with \( C_{q,p} \), a fuzzy \( \text{if-then} \) rule is formed between them (i.e., if \( A_{k,l} \) then \( C_{q,p} \)). Memory between two nodes \( A_{k,l} \) and \( C_{q,p} \) is stored as synaptic weight \( w_{k,l,q,p} \), while modulatory weight \( m_{k,l,q,p} \) signifies the connection strength between ICMk and OCMj. As a result, a rule is weighted by resultant weight \( v_{k,l,q,p} \), which is computed as

\[
v_{k,l,q,p} = m_{k,l,q,p} \cdot w_{k,l,q,p}.
\] 

(2)

As a convention, the output nodes and maps are denoted using Z with the subscripts specifying its origin. For example, \( Z_{\text{IF}_i} \) represents the output of node \( \text{IF}_i \), and \( Z_{\text{IM}_i} \) denotes the output vector of nodes in \( \text{IM}_i \). All outputs from a layer are propagated to the corresponding inputs at the next layer with unity link-weight \( \lambda = 1.0 \) where a connection exists, except for the connections between layers 3 and 4 where they are weighted by \( v_{k,l,q,p} \).

A. Layer 1: Input Fuzzifier

A \( \text{IF}_i \) node in layer 1 fuzzifies the input into a fuzzy membership function given by

\[
Z_{\text{IF}_i} = \mu(x_i)
\] 

(3)

where \( \mu(x) \in [0, 1] \) is the membership function of \( \text{IF}_i \). If a fuzzy input is presented to a node in this layer, the node simply redirects it as the output of the node (i.e., \( Z_{\text{IF}_i} = x_i \)). Two fuzzifying functions were identified: Gaussian \( G(\cdot) \), and Laplacian of Gaussian \( \text{LoG}(\cdot) \). The \( \text{LoG}(\cdot) \) function produces similar effects as lateral inhibition in the biological neural circuitry [15], which is used to increase signal discrimination by improving feature contrast.

B. Layer 2: Single Antecedent

In layer 2, a \( \text{IL}_{i,j} \) node computes the fuzzy subsethood measure between \( \mu_{i,j}(\cdot) \) and \( Z_{\text{IF}_i} \). Adopting the minimum T-norm operator for the intersection operation, \( Z_{\text{IL}_{i,j}} \) can be approximated as (4), where \( Z_{\text{IF}_i} \in [0, 1] \) and \( Z_{\text{IL}_{i,j}} \in [-1, 0] \) are positive and negative components of \( Z_{\text{IF}_i} \), respectively, and \( \mu_{i,j}(\cdot) \) is the membership function of input label \( \text{IL}_{i,j} \).

C. Layer 3: Multiple Antecedent

In layer 3, output \( Z_{\text{AK}_{k,l}} \in [-1, 1] \) is known as the firing strength, resulting from the conjunction of \( \text{IL}_{i,j} \) nodes. Using the algebraic product T-norm operator, \( Z_{\text{AK}_{k,l}} \) is given by

\[
Z_{\text{AK}_{k,l}} = \prod_{i=1}^{I} \prod_{j=1}^{J} \lambda_{\text{IL}_{i,j}} \cdot Z_{\text{IL}_{i,j}}
\] 

(5)

where \( \lambda_{\text{IL}_{i,j}} \cdot A_{k,l} \) is a link weight of 1 for a connection between \( \text{IL}_{i,j} \) and \( A_{k,l} \), and is otherwise 0.

D. Layer 4: Multiple Consequence

For layer 4, output \( Z_{\text{OL}_{i,j}} \in [-1, 1] \) of \( \text{OL}_{i,j} \) is its activation level \( \phi_{\text{OL}_{i,j}} \), which is obtained through the recalling process to be explained in Section IV, and given by

\[
Z_{\text{OL}_{i,j}} = \phi_{\text{OL}_{i,j}}.
\] 

(6)

E. Layer 5: Single Consequence

In layer 5, \( Z_{\text{OL}_{i,j}} \in [-1, 1] \) is the maximum activation level of connected \( \text{OL}_{i,j} \) nodes, defined as

\[
Z_{\text{OL}_{i,j}} = \max_{x \in q,y \in p} \left( \lambda_{\text{OL}_{i,j}} \cdot Z_{\text{OL}_{i,j}} \right)
\] 

(7)

where \( \lambda_{\text{OL}_{i,j}} \cdot \text{OL}_{i,j} \) is a link weight of 1 if \( \text{OL}_{i,j} \) is connected to \( \text{OL}_{i,j} \), and is 0 otherwise.

F. Layer 6: Output Defuzzifier

Finally, layer 6 defuzzifies the fuzzy outputs from layer 5 to produce crisp outputs. The center of area defuzzification scheme \( \text{COA} \) is applied for every \( \text{OD}_i \) as follows

\[
Z_{\text{OD}_i} = \text{COA} \left( \sum_{j=1}^{R_s} Z_{\text{OL}_{i,j}} \right)
\] 

(8)

where \( \sum_{j=1}^{R_s} Z_{\text{OL}_{i,j}} \) is the aggregated output membership function.

III. PHASES IN THE LEARNING PROCESS

The learning process (Fig. 2) consists of three consecutive phases: 1) unsupervised membership function initialization, 2) supervised Hebbian learning [16], and 3) supervised error reduction. During learning, the output layers 4 to 6 are functionally similar to input layers 1 to 3, which are fuzzification for layer 6 (Section II-A), subhood calculation for layer 5 (Section II-B), and conjunction formation for layer 4 (Section II-C). For clarity, an alternate symbol \( \bar{Z} \) is used to represent an output obtained via this reverse propagation process.

Fig. 2. Phases in the learning process (highlighted in gray).
A. Unsupervised Membership Function Initialization

A linguistic label for a particular linguistic variable, is represented by a neuron in a feature map. In FASCOM, feature maps may be scalar or cyclic [17] while membership functions of neurons may be discrete, rectangular or Gaussian. The first two result in crisp or classical sets while the third results in fuzzy sets. The number of neurons in a feature map is fixed according to the number of labels of a linguistic variable.

The initialization of membership functions involves a three-stages. First, the centroids of membership functions of these neurons are then placed evenly across the feature space represented by the feature map. Next, a membership function (i.e. discrete, rectangular, or Gaussian) is applied for each neuron. Finally, uniform information density equalization is performed.

The third stage of equalization based on information density results in a disproportion in neuron allocation. This is inspired by biological sensory maps, whereby more neurons are allocated to areas with higher usage or sensitivity requirements [18]–[22]. In this case, FASCOM allocates neurons to optimize information density [23] of the training data across feature space (Fig. 3d). This can be achieved by first forming a histogram of data points and smoothing it using a Gaussian smoothing function (Fig. 3a). This is followed by equalizing the histogram (Fig. 3b) and mapping the equalized scale to the feature map (Fig. 3c). With more neurons allocated to the areas with higher information concentration, the output will be more responsive and precise.

B. Supervised Hebbian Learning

The second stage of the learning process is a hybrid of Hebbian learning [16], whereby the simultaneous stimulation of two connected neurons results in an increase of synaptic efficacy between them. Assuming neurons $i$ and $j$ are respective $A_{k,j}$ and $C_{q,p}$ nodes in $ICM_k$ and $OCM_p$, then the change in synaptic weight $\Delta w_{i,j}$ between $i$ and $j$ can be computed based on their firing strengths $Z_i$ and $Z_j$ as follows:

$$\Delta w_{i,j} = \begin{cases} 0 & \text{if } (Z_i \leq 0 \text{ and } Z_j \leq 0), \\ |\Delta w_{i,j}|_{\text{max}} \cdot Z_i Z_j & \text{otherwise.} \end{cases}$$  

where $|\Delta w_{i,j}|_{\text{max}}$ is the maximum possible absolute change in synaptic weight. $\Delta w_{i,j}$ is aggregated with the current $w_{i,j}$ to form the new synaptic weight, as in

$$w_{i,j} = w_{i,j} + \Delta w_{i,j}.$$  

$|\Delta w_{i,j}|_{\text{max}}$ is determined by the mechanisms of synaptic plasticity, forgetting and long-term facilitation [15] result in the following concepts:

1) The effects of both synaptic plasticity and forgetting are approximated as constants through time. Through mutual cancellation, the approximated combined effect is a DC component.
2) The effect of long-term facilitation is initially high, but decays as the memory strengthened. When the memory is strong, the effect of facilitation is insignificant.
3) The initial effect of long-term facilitation is very much greater than that of the resultant effect of synaptic plasticity and forgetting.

A decaying sigmoid function is used to describe the effects of these three concepts, defined by

$$|\Delta w_{i,j}|_{\text{max}} = -\frac{k_{\text{LTF}}}{1 + \exp(-a|w_{i,j} - c|)} + k_{\text{LTF}} + k_{\text{DC}}$$  

where $k_{\text{LTF}}$ is the initial effect of long-term facilitation, $k_{\text{DC}}$ is the resultant DC component from synaptic plasticity and forgetting, while $a$ and $c$ define the slope of the function.

The process of modifying synaptic weights is performed for all training samples, and may result in memories being stored to be strong like long-term memory or weak like short-term memory.

C. Supervised Error Reduction

After Hebbian learning, the supervised error reduction phase is performed to fine-tune the network. The varying influences of an input dimension across output feature space is discovered stored as modulatory weights $m_{k,q,p} \in \mathbb{R}^{\text{nonneg}}$.

Table I lists the symbols used to explain the algorithm shown in Algorithm 1. Modulatory weights are first initialized to 1.0 (line 1). Temporary modulatory weights are then assigned for every training input-output pair $n$ (line 4). Next, memory is recalled using the process described in Section IV (line 5) and an initial error is obtained for each output node (line 6). Next, using only one input conjuncted map $ICM_k$ in each iteration of $t$, memory is again recalled (line 8) and a new error for each output node is computed (line 9). The difference between the new and initial errors is computed (line 10) and based on this, the temporary modulatory weights for the training instance $n$ are modified (line 11). After processing all $ICM$s for all training samples, the modulatory weights are updated (line 14) and the entire
process is repeated until a terminating condition is met. The algorithm terminates when: 1) the maximum number of epoch is reached, or 2) the total change in error falls below a low threshold (i.e. $\sum_{q,p=1}^{Q} \sum_{p=1}^{P} \Delta E_{u,p}^{(k)} \approx 0$).

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m_{k,q,p}$</td>
<td>Modulatory weight between ICM$<em>k$ and C$</em>{q,p}$</td>
</tr>
<tr>
<td>$u_{k,q,p}^{(n)}$</td>
<td>Temporary value of $m_{k,q,p}$ for the nth training input-output pair</td>
</tr>
<tr>
<td>$U_{\text{mod}}$</td>
<td>Factor that updates temporary modulatory weights $u_{k,q,p}^{(n)}$</td>
</tr>
<tr>
<td>$N$</td>
<td>No. of training input-output pairs</td>
</tr>
<tr>
<td>$Z_{L3} = [Z_{L3}^{(1)}, ..., Z_{L3}^{(N)}]$</td>
<td>Training input vector</td>
</tr>
<tr>
<td>$Z_{L4} = [Z_{L4}^{(1)}, ..., Z_{L4}^{(N)}]$</td>
<td>Training output vector</td>
</tr>
<tr>
<td>$Z_{L3}^{(n)} = [\phi_{OCM_1}^{Z_{L3}^{(n)}}, ..., \phi_{OCM_q}^{Z_{L3}^{(n)}}]$</td>
<td>nth training input vector</td>
</tr>
<tr>
<td>$Z_{L4}^{(n)} = [\phi_{OCM_1}^{Z_{L4}^{(n)}}, ..., \phi_{OCM_q}^{Z_{L4}^{(n)}}]$</td>
<td>nth training output vector</td>
</tr>
<tr>
<td>$\phi_{OCM_q} = [\phi_{C_{q,1}}, ..., \phi_{C_{q,P}}]$</td>
<td>Propagated output of layer 4</td>
</tr>
<tr>
<td>$\phi_{OCM_q}$</td>
<td>Activates level of C$_{q,p}$</td>
</tr>
<tr>
<td>$\xi_{q,p}^{\text{init}}$</td>
<td>New squared-error for C$_{q,p}$</td>
</tr>
<tr>
<td>$\xi_{q,p}^{\text{new}}$</td>
<td>New squared-error for C$_{q,p}$</td>
</tr>
<tr>
<td>$\Delta E_{C_{q,p}}$</td>
<td>Scaled error change for C$_{q,p}$</td>
</tr>
</tbody>
</table>

Algorithm 1: Supervised error reduction algorithm

$U_{\text{mod}}$ defined in (12), determines the magnitude of change in $u_{k,q,p}^{(n)}$. If the error decreases when using only ICM$_k$, it means that ICM$_k$ is important for output prediction and $u_{k,q,p}^{(n)}$ is strengthened. When the opposite occurs, ICM$_k$ is insignificant and $u_{k,q,p}^{(n)}$ should be reduced.

IV. RECALLING PROCESS

In the recalling process (Fig. 4), based on the input presented to layer 1 of the encoded network, input functions of layers 1 to 3 are performed (Sections II-A, II-B and II-C). The memory recall process then occurs in the hybrid associative memory. Finally, the output functions of layers 4 to 6 are performed (Sections II-D, II-E and II-F) and the estimated output is recalled. Here, we explain the signaling processes of neurons within the hybrid associative memory.

V. DATA PREDICTION EXPERIMENTS

Four data prediction tasks were used to benchmark our architecture against other existing architectures. They include three separate sets of data from Nakaniishi’s nonlinear estimation tasks [24] and real-world dataset for predicting the density of highway traffic [25].

1520 2008 International Joint Conference on Neural Networks (IJCNN 2008)
A. Nakanishi’s Nonlinear Estimation Tasks

The Nakanishi’s dataset [24] consists of three examples of real-world nonlinear estimation tasks. The tasks for data prediction are namely: 1) a nonlinear system, 2) the human operation of a chemical plant, and 3) the daily stock price of a stock in a stock market.

Based on these three experiments, FASCOM’s performance based on data prediction accuracy was benchmarked against Hebb-RR [27], SVM [26], RSPOP [8], DENFIS [28], POP-CRI [29], ANFIS [30], [31] and EFuNN [32]. Two performance measures are used for this evaluation, namely the mean squared error (MSE) and the Pearson product-moment correlation coefficient (R).

1) Example A: Nonlinear System: FASCOM was used to model a nonlinear system given by

\[ y = (1 + x_1^2 + x_2^3)^2: (1 \leq x_1, x_2 \leq 5) \]

The dataset consists of four input variables \((x_1, x_2, x_3, x_4)\) and one output variable \((y)\), whereby only \(x_1, x_2\) are useful and \(x_3, x_4\) are irrelevant.

The mean of modulatory weights discovered was computed across the output feature space. It was found that the input conjuncted maps ICMs representing \(x_3, x_4\) and \(x_5 \land x_4\) exerted no influence on the outcome of \(y\) and could be discarded. This is similar to the results obtained by [8], [24], [27]. For this example task, FASCOM was most accurate when compared to other benchmarked architectures.

2) Example B: Chemical Plant Operation: This example involves the human operation of a chemical plant, whereby five inputs representing monomer concentration \((x_1)\), charge of monomer concentration \((x_2)\), monomer flow rate \((x_3)\), and local temperatures inside the plant \((x_4, x_5)\), are used to estimate the set point for monomer flow rate \((y)\).

Similar to [24], FASCOM discarded \(x_2, x_3, x_4\) and \(x_5\). As a comparison, [8] discarded \(x_1, x_2, x_5\), while [27] discarded all but \(x_3\). Also, we deduce that the set point for monomer flow rate \((y)\) depends mainly on the combination of monomer flow rate \((x_3)\) and monomer concentration \((x_1)\) (i.e. \(x_3 \land x_1\)). For this example, with a MSE of \(8.170 \times 10^3\) and near perfect R value, FASCOM’s performance was significantly better than the state-of-the-art.

3) Example C: Stock Price Forecasting: The prediction of the price of a stock \(y\) for this example is performed with the ten inputs, which are the past and present moving averages over a middle period \((x_1, x_2)\), past and present separation ratios with respect to moving average over both short and middle periods \((x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10})\), present change of moving average over both short and long periods \((x_{5}, x_9)\) and past and present price changes \((x_{6}, x_{7})\).

The supervised error reduction algorithm significantly reduced inputs corresponding to \(x_1, x_2, x_3, x_4, x_5, x_6, x_8, x_9, x_{10}\). [24] discarded inputs \(x_1, x_2, x_3, x_4, x_7, x_9, x_{10}\), \(x_8\) discarded only \(x_2, x_3, x_6, x_7\), and \(x_{10}\). Interestingly, all methods did not discard \(x_5\), which means the present separation ratio with respect to moving average over a middle period \((x_1)\) is a critical component to predict stock prices. Similar to the previous two examples, FASCOM outperformed all other benchmarked architectures.

4) Discussion: The three tasks were also used to analyze the effects of three different experimental initializations of the learning process:

1) using all three phases in the learning process
2) omitting membership function initialization (phase 1)
3) omitting error reduction (phase 3)

For each task, each initialization’s MSE was computed and normalized with respect to the highest MSE amongst the three initializations. For all three tasks, the comparison between the three initializations indicates that the inclusion of both phases 1 and 3 produced a MSE that is significantly lower than when either one was omitted (Fig. 5). This shows that both phases 1 and 3 are crucial in improving the accuracy of data prediction and their existence within the learning process is justified.

Fig. 5. Comparison between the three different experimental initializations.
B. Highway Traffic Density Prediction

The raw traffic data [25], [33] was collected for three straight lanes and two exit lanes, at site 29 located at exit 5 along the east bound direction of the Pan Island Expressway (PIE) in Singapore, using loop detectors embedded beneath the road surface (Fig. 6). Data spanning a period of six days from September 5 to 10, 1996, for the three straight lanes (i.e. lanes 1 to 3) were considered for this experiment. The dataset has four input attributes representing the normalized time and the traffic density of each of the three lanes.

Fig. 6. Photograph of site 29 where the traffic data was collected.

The purpose of this experiment is to model the traffic flow trend and subsequently produce predictions of the traffic density of each lane at time \( t + \tau \), where \( \tau = 5, 15, 30, 45, 60 \) min is the prediction time interval. Three cross-validation groups (CV1, CV2 and CV3) of training and test sets were used for evaluation purposes [34]. The mean squared error (MSE) and the Pearson product-moment correlation coefficient (R) were computed for each predictions run. From the example in Fig. 7, we observe that the prediction accuracy decreases as the time interval \( \tau \) increases.

To evaluate the accuracy of prediction, the “Avg MSE” indicator was computed by taking the average MSE across all 45 prediction runs (3 lanes, 5 time intervals and 3 cross-validation groups). Also, the “Var” indicator reflecting the consistency of predictions over different time intervals across the three lanes was computed by taking the change in the mean of R from \( \tau = 5\) min to \( \tau = 60\) min expressed as a percentage of the former. This was then averaged across all three lanes to produce “Avg Var”.

The results of traffic density prediction was compared to Hebb-RR [27], SVM [26], RSPop [8], POP-CRI [29], DENFIS [28], GeSoFNN [34] and EFuNN [32]. FASCOM significantly outperformed all other architectures based on the results as shown in Fig. 8, with the best combination of “Avg MSE” (0.098) and “Avg Var” (19.0%) as compared to other architectures. The results indicates that the output prediction by FASCOM are both highly accurate and consistent over different time intervals. This is desirable.

Fig. 7. Traffic density prediction of lane 1 using CV1.

Fig. 8. FASCOM outperforms other benchmarked architectures for highway traffic density prediction.
VI. CONCLUSIONS
This paper proposed the Fuzzy Associative Conjuncted Maps (FASCOM) fuzzy neural network, which represents information using conjuncted fuzzy sets and learns associations between them through three consecutive unsupervised and supervised phases in the learning process. During the first phase of unsupervised membership function initialization, neurons are allocated to optimize information density. For the supervised learning phase, the single pass Hebbian learning performed is based on the memory mechanisms of synaptic plasticity, forgetting and long-term facilitation. Finally, the final supervised phase involves fine-tuning the network using the supervised error reduction algorithm, which uses an error comparison to determine the influence of an input dimension across output space.

In the series of experiments performed, we showed that each of the phases contributed to the overall performance of the architecture. We were also able to demonstrate FASCOM’s effectiveness in performing nonlinear data prediction on a variety of real-world problems of different natures and consisting of noisy data, such as nonlinear system modeling, chemical plant operation, stock price forecasting and traffic density prediction. For every experiment, FASCOM produced the best data prediction accuracy when benchmarked against existing architectures.

REFERENCES

2008 International Joint Conference on Neural Networks (IJCNN 2008) 1523