Online Self-Adaptive Cellular Neural Network Architecture for Robust Time-Series Forecast

Article *in* IEEE Transactions on Neural Networks and Learning Systems · January 2016

Ahmad Haj Mosa
Alpen-Adria-Universität Klagenfurt
16 PUBLICATIONS **9** CITATIONS
SEE PROFILE

Kyandoghere Kyamakya
Alpen-Adria-Universität Klagenfurt
186 PUBLICATIONS **1,339** CITATIONS
SEE PROFILE

Mouhannad Ali
Alpen-Adria-Universität Klagenfurt
11 PUBLICATIONS **2** CITATIONS
SEE PROFILE

Fadi Al Machot
Christian-Albrechts-Universität zu Kiel
38 PUBLICATIONS **45** CITATIONS
SEE PROFILE

Online Self-Adaptive Cellular Neural Network Architecture for Robust Time-Series Forecast.

Ahmad Haj Mosa*, Kyandoghere Kyamakya*, Mouhannad Ali*, and Fadi Al Machot*
*Universitaet Klagenfurt, Institute of Smart System Technologies

***Abstract*—This paper is submited for publication on IEEE Transcations for Neural Networks and Learning System (ID: TNNLS-2016-P-6504) This paper presents a novel online adaptive neuro-computing framework for a robust time-series forecast. The proposed framework does mimic the human mind's biological two-thinking model. Our mind makes decisions/calculations using a two-connected system. A first system, System 1, the so-called the intuitive system, makes decisions based on our experience. A second system, System 2, the controller system, does control the decisions of System 1 by either modifying or trusting them. Similarly to the human mind's two-systems model, this paper proposes an artificial framework consisting of two cellular neural network (CNN) systems. The first CNN processor does represent the intuitive system and we call it Intuitive-CNN. The Second CNN processor does represent the controller system, which is called Controller-CNN. Both are connected within a general framework that we name OSA-CNN. The proposed framework is extensively tested, validated and benchmarked with the best state-of-the-art related methods, while involving real field time-series data. Multiple scenarios are considered: traffic flow data extracted from the PeMs traffic database and the 111 time-series collected from the so-called NN3 competitions. The novel OSA-CNN concept does remarkably highly outperform the state-of-the-art competing methods regarding both performance and universality.**

***Index Terms*—Time-series forecast (TFS), Cellular neural networks (CNN), Echo state network (ESN), Model reference neural network adaptive control (MRNNAC), Recursive particle swarm optimization (RPSO), Online self-adaptive CNN (OSA-CNN)**

I. Introduction

The importance of time-series in science and technology is extremly high nowadays. It is a fact that that time-series forecasting (TSF) is widely utilized in several areas such as engineering, science, and finances [1], [2], [3]. The high interest for time-series analysis and forecasting is motivated by the need for understanding and predicting the future of various technical, social, and natural systems. In this perspective several forecasting methods have been developed. Some of which rely on either linear or nonlinear statistical models. One of these promising forecasting concepts do involve artificial neural networks (ANN). Over the past decades, time-series forecasting has been predominately implemented using statistical analysis based methods. One prominent example is the autoregressive integrated moving averages (ARIMA) [4], [5] which has been used by scientists for several years. However, regarding both the emerging technologies and the high complexity of much recent data, artificial neural networks have been preferred and have become a serious processing alternative in time-series forecasting. The fact that artificial neural networks are characterized by a clearly superior performance in classification and regression problems in various practical domains has made them to be a preferred method for time-series forecasting [6].

Forecasting is without any doubt a major challenge that characterizes most global data analysis. That is, accurate and timely flow of information is needed for individuals or experts to make the relevant decisions regarding a certain phenomenon under observation. It is easier to make a decision based data or knowledge on a future evolution of the system. However, bid data can be very challenging due to the uncertainty of the related variables [7]. Thus, time-series forecasting will be helpful for making a calculated decision and predicting the future trend of the variables. The phenomena that are used to produce time-series can sometimes be unknown and the information available for forecasting is by large limited to the past values of the time-series [8]. Therefore, it is essential and vital to utilize the most relevant number of previous values, termed lags, when undertaking a time-series forecasting. This is the gap that ANN is mostly developed to close.
This paper is organized as follows: Section II address the related work. Section III provides background information related to the techniques that are involved in our method. Section IV presents the OSA-CNN model. Section **??** explains the learning algorithm used for OSA-CNN. Section VI shows the obtained results while using two different benchmark online data sets. Finally, a conclusion is given in Section VII.

II. Related Works

Compared to earlier works especially to some of the existing statistical forecasting methods, neural network approaches have proven several outstanding characteristics that include the following ones: (a) artificial neural networks can use both linear and nonlinear data; (b) artificial neural networks are non-parametric, that is, ANNs do not need an explicit underlying model if compared to other forecasting methods; and (c) ANNs are very flexible and universal and hence can be used for more complicated and complex data structures/models. The above listed characteristics make ANNs to be regarded as the best current methods/concepts for time-series forecasting. Several empirical studies have shown a significantly superior performance of ANNs over other statistical forecasting tools. The superiority measure/benchmarking has been mainly based

Corresponding author: Ahmad Haj Mosa a research assistant at Alpen Adria universitt Klagenfurt, Institute of Smart System Technologies (email: Ahmad.hajmosa@aau.at

on either a single or a small set of time-series. A special and very prominent feature of ANN is its ability to combine both long and short-term forecasting approaches to provide a better system-model with highest performance potential [9].

A special type of ANN, which is the recurrent neural network (RNN), has been a breakthrough in the scientific field related to time-series forecasting. Through recurrent neural networks data analysis and forecasting has taken tremendous steps ahead and hence enhanced scientific decision making has been enabled. However, ANN does also face several limitations, which have been revealed by several studies. For instance, according to [10], in some cases, ANN have depicted some inferiority regarding performance when compared to traditional statistical methods. The inconsistency of the performance of artificial neural networks, as revealed by several studies, is attributed to the inherent requirement of a sufficient number of training samples needed to insure that the ANN can be adequately trained. Also, several real-world applications occur during a short time period; there it becomes inadequate for artificial neural networks to reveal the underlying pattern and structure. This leads to poor prediction performance for RNNs. From this perspective and in order to increase the universality performance of ANN, Rahman (2016) [11] and Weizhong (2012) [10] suggested an ensemble architecture of multi-layered neural networks MLP and generalized regression neural network (GRNN) respectively. Both presented a promising universality performance by fusing multi ANN predictors. Another trending type of ANN is the Echo State Network (ESN) [12], [13], [14] which benefits from its big-size hidden layer to model well any complex nonlinear time-series. It is also very fast to be trained thanks to the random generation of hidden layer weights followed by a simple LSE (Least Square Estimation) for the output layer.

III. BACKGROUND KNOWLEDGE

In this section, we briefly provide background information related to the four basic techniques that are involved as building bricks in the design of the OSA-CNN (online self-adaptive cellular neural network) architecture concept which is later described in this paper.

A. The Two-Systems Model of Cognitive Processes

The difference between conscious and unconscious processes (of the human mind) in decision-making is one of the key questions for which social scientists have conducted many studies to find appropriate answers. In the 2011 best-selling book *Thinking, Fast and Slow*, the Nobel Prize winner Daniel Kahneman answered that question by concluding that there are two different systems the brain uses to form thoughts[15]. These two systems are the following:

* **System 1**: is a fast, intuitive, unconscious and emotion-based decision-making system.
* **System 2**: is a slow, conscious and calculation-based decision-making system.

In other words, System 1 does involve both our intuition and past experiences. System 1 may give a correct decision if the related case is neither critical nor new to the system.

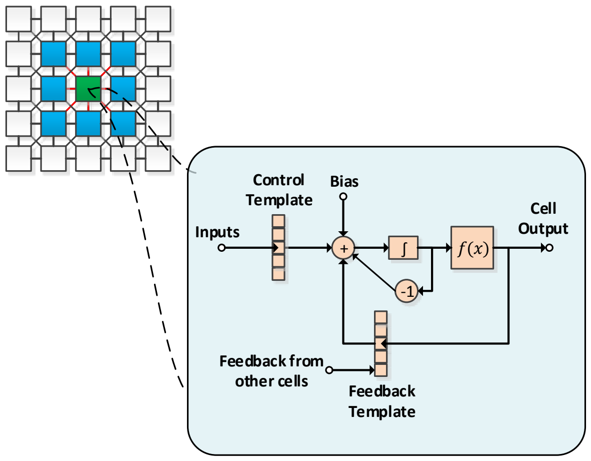


Fig. 1. The cellular neural network architecture as provided in [16]. The state model of a single CNN cell is given.

If the case is critical, new or unexperienced by the brain, System 1 will have a high likelihood of arriving at a wrong decision. In this case, System 2 will interfere with and control System 1 in order to ensure a more accurate outcome. This model of thinking inspired us to build an artificial framework for time-series prediction that mimics the two-systems human psychological system. Our novel framework consists of two systems as well. Similarly to the human mind, System 1 of OSA-CNN predicts the future value of the time-series. Also, System 2 of OSA-CNN controls System 1 and drives it to deliver a better performance. More details about the OSA-CNN artificial framework will be addressed later in Section IV.

B. Cellular Neural Networks

A Cellular neural network (CNN) is a universel system modeler expressed by a system of differential equations that expresses the relationship between close/neighboring nonlinear units. CNN has been suggested by Chua and Yang in the year 1988 [17]. It incorporates good features of both ANN and cellular automata (CA); this makes it more promising compared to both earlier approaches. However, despite the ability to integrate the qualities of both CA and ANN it does differ from CA and ANN by its nonlinear dynamical representation of the coupling amongst cells associated with its property of local connectivity. Due to its parallelism capability, CNN has a significant processing capacity and does provide easy and flexible implementability in either software or hardware or even in an analog circuit or in hybrid constellations of all the lastly named target platforms. More importantly, CNN can be embraced as a part of digital platforms (see CNN chips [18], [19]) or emulated on top of digital platforms like FPGA and GPU [20], [21]. Recently, reconfigurable analog platforms like FPAA (field programmable analog array[22]) have offered a further realization alternative. The generally proposed state

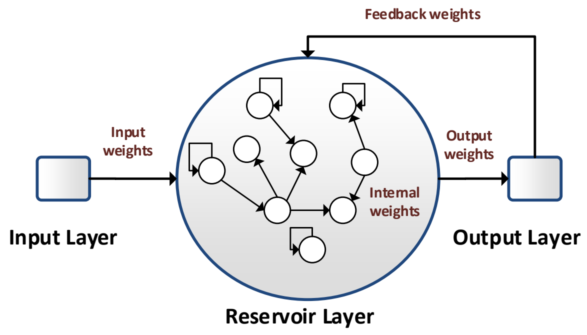


Fig. 2. The echo state network architecture. Inputs, Reservoir and Output layers are presented. The input layer feeds the reservoir which has a random internal connection between it inner cells. The reservoir feeds the output layer which also feeds back the reservoir [12]

equation of a 3×3 neighborhood CNN cell is given in (1) [16].

$$\frac{dx_i(t)}{dt} = -x_i(t) + \sum_{k=1}^{n} a_{k,i} y_k(t) + \sum_{k=1}^{m} b_{k,i} u_k(t) + I \tag{1}$$

where $x_i(t)$ and

$$y_i(t) = f(x_i(t)) = \frac{1}{2}(|x_i(t)+1| - |x_i(t)-1|) \tag{2}$$

do represent respectively the current system state and the output of the cell (i); $u_k(t)$ is the current kth input value, $A = a_{1,1} \cdots a_{n,n}$ is the feedback template, $b = b_{1,1} \cdots b_{m,n}$ is the control template, I is the cell bias; n and m are the number of cells and inputs respectively.

In Fig. 1, the CNN architecture is presented in the form of a graphical representation of (1). Generally, CNN is perceived as a continuous-time oscillator that is driven/excited by an external input. But discrete-time CNN model variants (DT-CNN) are also available [23][24]. For the continuous time CNN, the differential equation (1) needs to be solved in order to get the value of each cell state. In the DT-CNN case, a corresponding difference equation is considered instead.

C. Echo State Network

The echo state network (ESN) is a novel type of Recurrent Neural Networks (RNN) that has produced excellent performance in the forecasting of either nonlinear or non-Gaussian dynamical sequences. Primarily, ESN was designed by Jaeger back in the year 2001 with the objective of using an RNN with a large number of neurons to provide a desirable sparsity from which valuable information coming from the inputs can be better extracted [12]. Compared to other neural networks concepts, ESN does have the clear merits of enclosing finer details regarding the contribution of the past in a form that reflects the recent past at best. This feature becomes possible due to its large number of neurons along with the recurrent connections, which give it a short-term memory. Therefore, the challenge of finding the best time-delay of inputs (i.e. how much history to be considered in the input vector) is no more an issue, unlike previous neural network concepts. Another worth mentioning advantage of ESN is that it is simple to train. ESN has the following configurations which is also visualized in Fig.2:

1) Its first layer called the reservoir consists of nonlinear neurons. Virtually, in this layer, all neurons are connected with each other and with themselves (self-feedback).
2) The network inputs are respectively connected to each neuron of the reservoir layer. Also, each output of layer 1 neurons is connected to all reservoir layer neurons.
3) The output layer consists of a linear regressors neuron vector which is connected to the outputs of all hidden-layer reservoir neurons.
4) The network is trained in a first stage by a random initialization of both the reservoir inner/hidden-layer weights and also of the weights between INPUT layer and Reservoir Layer. Secondly, the reservoir-output weights are trained by using a regularized simple least-square method.

Despite many desirable features, traditional ESN concepts do experience two main limitations. These limitations include the issue of ill-conditioned solutions, that are usually related to linear regression or recursive least squares, which may lead to poor outcomes due to the large output weights that mostly affect the universality and the stability of the network. Furthermore, there is the problem of inputs uncertainty. Generally, uncertainty from the inputs is a very common case in industrial time-series prediction, where noise is a serious issue to be addressed [13].

D. Model Reference Neural Network Adaptive Control (MRNNAC)

Neural networks have been widely used in process control applications. One of the well-performing architectures was proposed by [25]. They introduced what is called the Model Reference Neural Network Adaptive Control (MRNNAC). Their proposed architecture consists of two connected networks. The first network (a Process Model Network) does model the dynamical process of the plant, while the second network (a Controller Network) controls the first one and, thus, drives it to a target point. Both networks are generally trained using one of the popular dynamic neural network training techniques [26][27]. In the following, we summarize/explain the configuration and later the training of both networks.

1) **Process Model Network**:
   a) It is used to predict the next process output.
   b) The current and delayed plant input, along with the delayed output, are the inputs to this network.
   c) A training set containing plant input-output measurements is used to train this network.
2) **Controller Network**:
   a) It is used to drive the process plant to a target set-point following a reference model, which models a particular trajectory that the plant output should follow to converge to the targeted set-point.

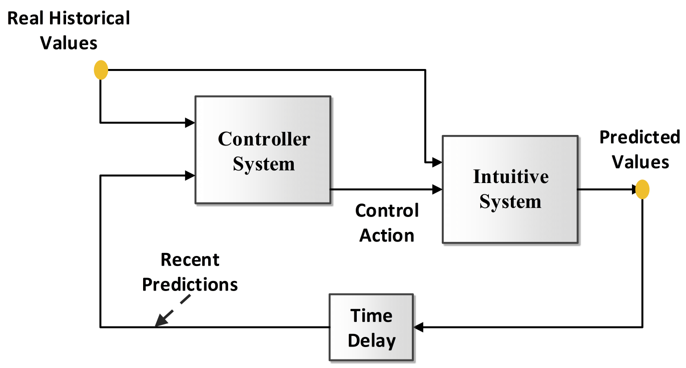


Fig. 3. The OSA-CNN Framework. The Intuitive-System takes the historical values to predict the future values of a time-series. The Controller-System reads and compares the historical values with the corresponding prediction values and manipulate the Intuitive-System output to improve the future performance.

b) The target set-point input (the reference value that the plant output should follow), along with the delayed outputs and inputs of the plant, are considered to be inputs to this network.
c) Once the process model network is trained, the controller network is trained by connecting it with the process model network. Then, a time-series of a varying set-point value is valued with its corresponding reference model to train the controller. In this training phase, only the controller network weights need to be learned because the process network weights have already been fixed from the first training phase (training of the process network).

IV. The OSA-CNN System Concept

OSA-CNN is built by integrating: a) a continuous-time nonlinear CNN oscillator system; b) the echo-state property and the easy training approach of ESN; and c) the theory of the two-systems' model of thinking. This combination of advantages makes OSA-CNN to be a precious architecture. As it is presented in Fig. 3, OSA-CNN consists of a two connected systems, mimicing the Two-Systems model of human cognitive processes. System 1 is the controller model (Controller-System). System 2 is the process model (Intuitive-System). Both systems are integrated similarly to MRNNAC. The role of the Intuitive-System is to predict the future value of a time-series given its historical records. The role of the Controller-System is then to assess the recent prediction performance of the Intuitive-System by comparing it with the corresponding real values. Accordingly it controls the Intuitive-System to improve the future prediction performance. In other words, the Controller-System drives the Intuitive-System in such a way that the recent prediction error is minimized. In the following, we address the data preparation needed for OSA-CNN along with a review of modeling details.

A. OSA-CNN Data Preparation

Given a time series $v(t)$, the forecasting goal is to predict $v(t)$ while considering the h historical values $v(t-k_1)...v(t-k_h)$ as shown in Fig 4(a), where k_i is a delay or lag. For modeling purposes, we treat the current and lagged values of the time-series as separate quantities denoted by $v_{k_0}(t)$ and $v_{k_1}(t)$ to $v_{k_h}(t)$ respectively. For modeling purposes, the current and lagged time-series are transformed into a series of quadratic pulses (i.e. step-functions) wich are delayed by one time-lag of the time series and which look as following (see Fig 4(b)):

a) We take each two samples of the original series, say $v_{k_0}(1)$ and $v_{k_0}(2)$, then we interpolate them with a square pulse that has the length Λ (interpolation size or the time-distance between two samples of the original time-series) which starts at $v_{k_0}(1)$ and ends at $v_{k_0}(2)$.
b) This square pulse has the constant value of $v_{k_0}(2)$ over all the interval of size Λ between lag-time (1) and lag-time (2).

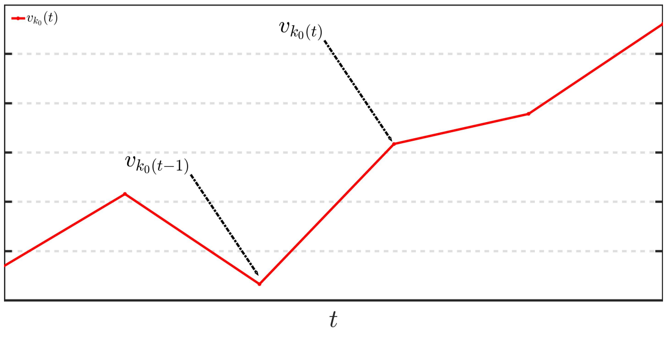
The aim of this interpolation process is to make the system (driven by the controller part) doing several prediction trials of the same target and eventually converges at the best possible prediction value. The resulted signals (i.e square pulses) are denoted by $vs_{k_0}(ti)$ and $vs_{k_1}(ti)$ to $vs_{k_h}(ti)$ respectively. Note that each of this signals has the length of Λ. In the next step, as in MRNNAC, we do need to obtain the reference series of the pulses series. A reference series is the same as the pulses series; however the difference is that the reference series consist of smooth steps instead of sharp square steps. The resulted smooth path (see $vr_j(ti)$ in Equation (3)) mimics the desired path that the plant should follow. The transformation from squares into smooth steps is generally done by using the following reference model, which is very known in the related literature[26], [28], [29], [30], [31]:

$$\frac{d^2 vr_j(ti)}{dti^2} = -\alpha vr_j(ti) + -\rho \frac{vr_j(ti)}{dti} + \gamma vs_j(ti) \tag{3}$$

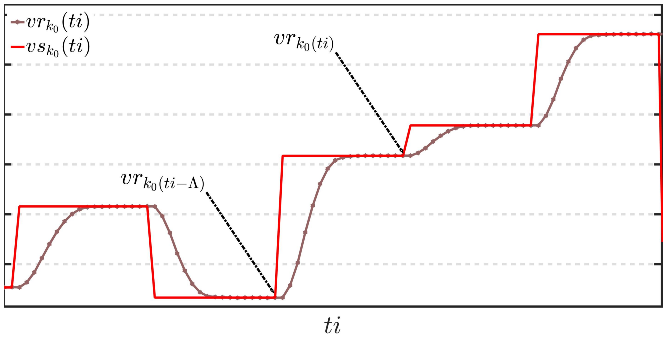
where $j = k_0$ to k_h, representing the current and lagged square steps series; α, ρ and γ are the feedback, damping and control coefficients that characterize the shape of the desired smooth path.These coefficients are selected empirically. The selected values are presented in the evaluation Section VI. Equation (3) is a second order differential equation that needs to be solved. Given a sequence/series of square pulses $vs_j(ti)$ and zero initial condition (at the very beginning oft he sequence) we solve (3) using the Matlab [32] Adams-Moulton solver [33] (ode113) solver with a variable step size between 0.05 and 0.1. Accordingly, the solution gives the reference smooth step series $vr_j(ti)$. Eventually, we get a set of two time-series which is used in the training and testing processes of OSA-CNN. These time series are:

a) The OSA-CNN input (square pulse) signals $vs_{k_1}(ti), \dots vs_{k_h}(ti)$: the step series of the historical values that are used as inputs for the prediction model.
b) The OSA-CNN output (smooth pulse) signals $vr_{k_0}(ti)$: the reference series of the future values that need to be predicted.

The output of OSA-CNN is denoted by $\hat{vr}_{k_0}(ti)$. The task of OSA-CNN is to minimize the difference between the input signals $vs_{k_1}(ti). \dots . vs_{k_h}(ti)$ and the history of the recent prediction values $\hat{vr}_{k_1}(ti), \dots \hat{vr}_{k_h}(ti)$.



(a)



(b)

Fig. 4. (a) An example of a time series $v_{k_0}(t)$ with two marked records $v_{k_0}(t)$ and $v_{k_0}(t-1)$ (b) The transformation of $v_{k_0}(t)$ into square pulses (red solid) and smooth reference (brown dotted) series which is constructed by solving (3) using the Matlab ode113 solver. It also shows the corresponding new position of $v_{k_0}(t)$ and $v_{k_0}(t-1)$ from (a) represented by $vr_{k_0(ti)}$ and $vr_{k_0(ti-\Lambda)}$ respectively, where Λ is the interpolation size or the number of interpolation points

B. OSA-CNN Model

As it is mentioned above, OSA-CNN consists of two connected systems: the Controller-System and the Intuitive-System. Each one of this two systems consists of a hidden and an output layer. The hidden layer is an ordinary cellular neural network as presented in Sub-section III-B. The output layer is nothing but a linear layer. Accordingly, OSA-CNN consists of four connected layers: Controller CNN (C-CNN), Controller Linear Regression (C-LR), Intuitive CNN (I-CNN) and Intuitive Linear Regression (I-LR) models. Figure 5 presents these four layers. As it is shown, the C-CNN is fed by the h lag reference square-pulses $vs_{k_1}(ti)..vs_{k_h}(ti)$ representing the recent real h values. It is also fed by the h lag predicted values $\hat{vr}_{k_1}(ti)..\hat{vr}_{k_h}(ti)$. The last input for C-CNN is the Controller-System output feedback $u(ti)$. C-CNN feeds forward the linear layer C-LR which does a linear mapping (weighted sum) to the Controller-System output. The I-CNN has two inputs, first one is again the h lag reference square pulse $vs_{k_1}(ti)..vs_{k_h}(ti)$ and the Intuitive-System output feedback $\hat{vr}_{k_0}(ti)$. The I-CNN feeds forward the linear layer I-LR which does a weighted sum of the I-CNN outputs. In addition, the control action for C-LR to I-LR is realized as an adaptive bias added to weighted sum in I-LR. In this way, the Controller-System adapts the Intuitive-System output by regulating the biases of the I-LR layer. In the following, we do address the modeling equation of both the first Controller-System and the Intuitive-CNN. Equations (4) (5) and (6) give the state, output models of C-CNN and C-LR respectively:

$$\frac{dx_i^{\textbf{c-cnn}}(ti)}{dti} = -x_i^{\textbf{c-cnn}}(ti) + \sum_{j=1}^{n} \boldsymbol{\omega}_{1_{i,j}}^{\textbf{c-cnn}} y_j^{\textbf{c-cnn}}(ti) + \sum_{j=1}^{h} \boldsymbol{\omega}_{2_{i,j}}^{\textbf{c-cnn}} vs_{k_j}(ti) + \sum_{j=1}^{h} \boldsymbol{\omega}_{3_{i,j}}^{\textbf{c-cnn}} \hat{vr}_{k_j}(ti) + \sum_{j=1}^{n} \boldsymbol{\omega}_{4_{i,j}}^{\textbf{c-cnn}} u_j(ti) + \boldsymbol{\varepsilon}_i^{\textbf{c-cnn}} \tag{4}$$

$$y_i^{\textbf{c-cnn}}(ti) = \frac{1}{2}(|x_i^{\textbf{c-cnn}} + 1| - |x_i^{\textbf{c-cnn}} - 1|) \tag{5}$$

$$\begin{aligned} u_1(ti) &= \sum_{i=1}^{n} \boldsymbol{\omega}_{1,i}^{\textbf{c-lr}} y_i^{\textbf{c-cnn}}(ti) + \boldsymbol{\varepsilon}_1^{\textbf{c-lr}} \\ u_2(ti) &= \sum_{i=1}^{n} \boldsymbol{\omega}_{2,i}^{\textbf{c-lr}} y_i^{\textbf{c-cnn}}(ti) + \boldsymbol{\varepsilon}_2^{\textbf{c-lr}} \\ &\vdots \\ u_n(ti) &= \sum_{i=1}^{n} \boldsymbol{\omega}_{n,i}^{\textbf{c-lr}} y_i^{\textbf{c-cnn}}(ti) + \boldsymbol{\varepsilon}_n^{\textbf{c-lr}} \end{aligned} \tag{6}$$

where $x_i^{\textbf{c-cnn}}(ti)$ and $y_i^{\textbf{c-cnn}}(ti)$ do represent respectively the system state and the output of the C-CNN ith cell; $u_i(ti)$ is the ith C-LR output; n and h are respectively the number of C-CNN cells and the number of input lags; $\boldsymbol{W}_1^{\textbf{c-cnn}} = (\boldsymbol{\omega}_{1_{1,1}}^{\textbf{c-cnn}} \ldots \boldsymbol{\omega}_{1_{n,n}}^{\textbf{c-cnn}})$ is the C-CNN feedback template; $\boldsymbol{W}_2^{\textbf{c-cnn}} = (\boldsymbol{\omega}_{2_{1,1}}^{\textbf{c-cnn}} \ldots \boldsymbol{\omega}_{2_{n,h}}^{\textbf{c-cnn}})$, $\boldsymbol{W}_3^{\textbf{c-cnn}} = (\boldsymbol{\omega}_{3_{1,1}}^{\textbf{c-cnn}} \ldots \boldsymbol{\omega}_{3_{n,h}}^{\textbf{c-cnn}})$ and $\boldsymbol{W}_4^{\textbf{c-cnn}} = (\boldsymbol{\omega}_{4_{1,1}}^{\textbf{c-cnn}} \ldots \boldsymbol{\omega}_{4_{n,n}}^{\textbf{c-cnn}})$ are the C-CNN control templates; $\boldsymbol{W}_i^{\textbf{c-lr}} = (\boldsymbol{\omega}_{i,1}^{\textbf{c-lr}} \ldots \boldsymbol{\omega}_{n,i}^{\textbf{c-lr}})$ is the ith C-LR template; $\boldsymbol{\varepsilon}^{\textbf{c-cnn}} = (\boldsymbol{\varepsilon}_1^{\textbf{c-cnn}} \ldots \boldsymbol{\varepsilon}_n^{\textbf{c-cnn}})$ and $\boldsymbol{\varepsilon}^{\textbf{c-lr}} = (\boldsymbol{\varepsilon}_1^{\textbf{c-lr}} \ldots \boldsymbol{\varepsilon}_n^{\textbf{c-lr}})$ are biases.

While, Equations (7), (8) and (9) give the state, output models of the I-CNN and I-LR respectively:

$$\frac{dx_i^{\textbf{i-cnn}}(ti)}{dti} = -x_i^{\textbf{i-cnn}}(ti) + \sum_{j=1}^{n} \boldsymbol{\omega}_{1_{i,j}}^{\textbf{i-cnn}} y_j^{\textbf{i-cnn}}(ti) + \sum_{j=1}^{h} \boldsymbol{\omega}_{2_{i,j}}^{\textbf{i-cnn}} vs_{k_j}(ti) + \boldsymbol{\omega}_{3_i}^{\textbf{i-cnn}} \hat{vr}_{k_0}(ti) + \boldsymbol{\varepsilon}_i^{\textbf{i-cnn}} \tag{7}$$

$$y_i^{\textbf{i-cnn}}(ti) = \frac{1}{2}(|x_i^{\textbf{i-cnn}} + 1| - |x_i^{\textbf{i-cnn}} - 1|) \tag{8}$$

$$\hat{vr}_{k_0}(ti) = \sum_{i=1}^{n} (\boldsymbol{\omega}_i^{\textbf{i-lr}} y_i^{\textbf{i-cnn}}(ti) + u_i(ti)) \tag{9}$$

where $x_i^{\textbf{i-cnn}}(ti)$ and $y_i^{\textbf{i-cnn}}(ti)$ do represent respectively the system state, the output of the I-CNN ith cell; n is the number of I-CNN cells; $\boldsymbol{W}_1^{\textbf{i-cnn}} = (\boldsymbol{\omega}_{1_{1,1}}^{\textbf{i-cnn}} \ldots \boldsymbol{\omega}_{1_{n,n}}^{\textbf{i-cnn}})$ is the I-CNN feedback template; $\boldsymbol{W}_2^{\textbf{i-cnn}} = (\boldsymbol{\omega}_{2_{1,1}}^{\textbf{i-cnn}} \ldots \boldsymbol{\omega}_{2_{n,h}}^{\textbf{i-cnn}})$ and $\boldsymbol{W}_3^{\textbf{i-cnn}} = (\boldsymbol{\omega}_{3_1}^{\textbf{i-cnn}} \ldots \boldsymbol{\omega}_{3_n}^{\textbf{i-cnn}})$ are the I-CNN control templates; $\boldsymbol{W}^{\textbf{i-lr}} = (\boldsymbol{\omega}_1^{\textbf{i-lr}} \ldots \boldsymbol{\omega}_n^{\textbf{i-lr}})$ is the I-LR template; $\varepsilon^{\textbf{i-cnn}} = (\varepsilon_1^{\textbf{i-cnn}} \ldots \varepsilon_n^{\textbf{i-cnn}})$ is a bias. To solve the OSA-CNN states equations we also use the Matlab [32] Adams-Moulton solver [33] (ode113) solver with a variable step size between 0.05 and 0.1. In the next section we will provide more details about the OSA-CNN configuration and training.

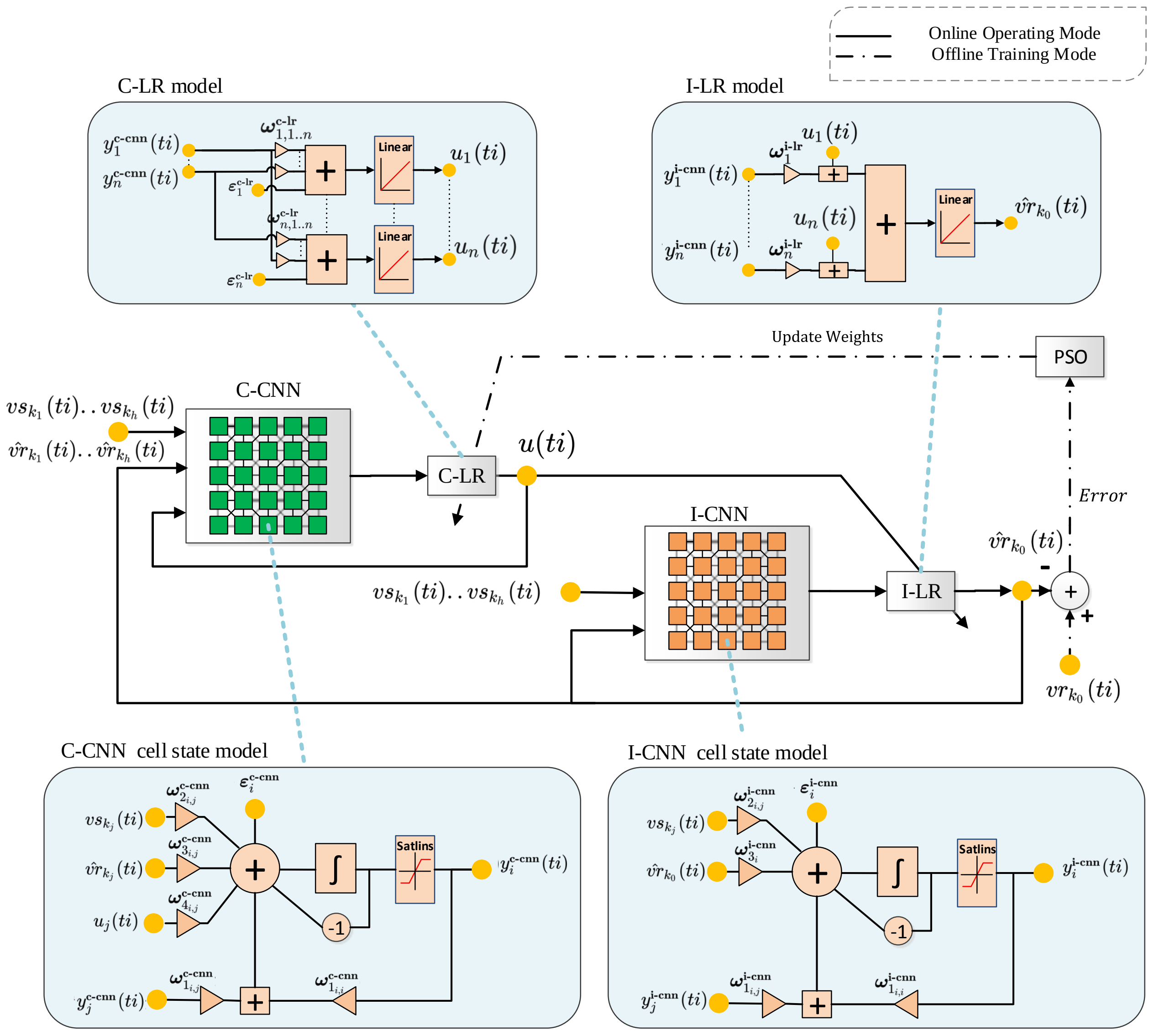


Fig. 5. The OSA-CNN four layers model. Two modes are presented. The online operating (solid line connections) and offline training (doted line) modes. In the operating mode. The first layer is the Controller CNN (C-CNN) connected to the Controller linear layer C-LR. The third layer is the Intuitive CNN (I-CNN) connected to the Intuitive linear layer I-LR. The controller output $u(ti)$ adapts the I-LR biases with respect to the difference between the real historical values $vs_{k_1}(ti)..vs_{k_h}(ti)$ and the corresponding predicted values $\hat{vr}_{k_1}(ti)..\hat{vr}_{k_h}(ti)$. In addition, four blocks visualizing the models of C-CNN cell state, I-CNN cell state, C-LR and I-LR are presented. In the offline training mode, PSO reads the prediction error and adapts accordingly the the C-LR weights

V. OSA-CNN Learning

FOR MORE DETAILS PLEASE CONTACT THE AUTHORS

VI. Evaluation Results

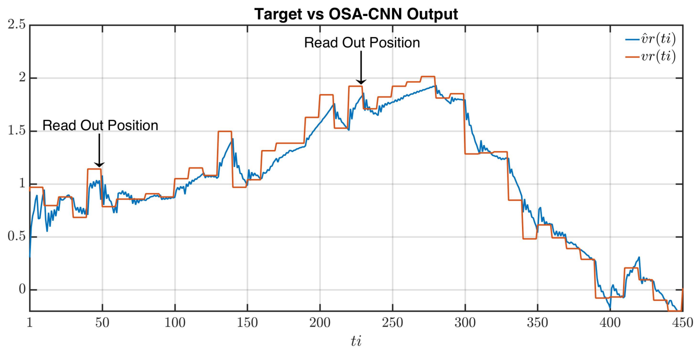
In this section, we evaluate OSA-CNN on two case studies. These two case studies are the traffic flow time series [45] and the NN3 time series competition [46] . All two have been widely studied and utilized as reliable benchmarking time series. Since the related cutting-edge methods with these three popular time series use different performance metrics, we utilize here different performance metrics for each case study. We also address two cases of time series prediction: the first scenario is one future step prediction, and the second one is the multi-step future prediction. For all two scenarios, in the tables, we address the training configuration parameters of both the random initialization phase and the recursive particle swarm optimization.

A. Traffic Flow Time Series

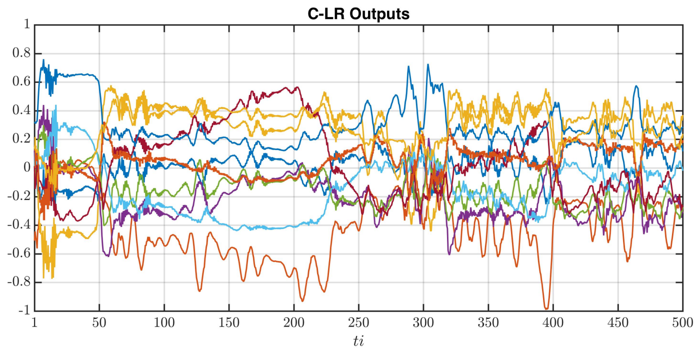
Advanced traffic management systems need timely best-possible accurate traffic flow or rather traffic demand information either for given road section(s) or for fixed sensor(s) or road position(s)/section(s). This information on traffic demand/flow should ideally be known for a future short time

TABLE I
THE RPSO CONFIGURATION PARAMETERS. THESE CONFIGURATIONS ARE USED IN ALL OF THE STUDIED TSF CASES

Parameter	Value
Swarm Size	24
$MAXITER$	100
ϖ_1	0.9
ϖ_2	0.4
$\delta_{1,1}$	2.5
$\delta_{1,2}$	0.5
$\delta_{2,1}$	2.5
$\delta_{2,2}$	0.5
Stopping Criteria 1	MAXITER is exceed
Stopping Criteria 2	$G_{min} = 1e - 10$
Stopping Criteria 3	$\Theta = 10$



(a)



(b)

Fig. 6. (a) An example of OSA-CNN in operation mode. The Signal in red is the ground truth square pulses series, Signal in blue is the OSA-CNN output which includes multiple prediction values. We read out the last predicted values and consider it as the best one. (b) the OSA-CNN corresponding controller output showing the interference of Controller system to improve the prediction performance.

horizon of e.g. 1-min, 5-min, 15-min or 30-min up to some hours, and thus enabling a reliable informational platform for pro-active optimized traffic management. Unfortunately, most existing traffic flow prediction methods are still very unsatisfying for many real-world applications, especially when it comes to forecasting traffic flow/demand over very short time horizons ranging from some seconds up to 15 minutes. This is because traffic counts within shorter time horizons do experience much higher levels of stochasticity and unpredictability. In this paper, we evaluate OSA-CNN on real field-data provided by the Caltrans Performance Measurement System (PeMS) database [45]. This database contains 15000 traffic detector flow data measured every 30 s. These sensors are located in freeways over California. Many researchers have used this dataset to evaluate their work. Generally, in their work, the prediction is made either on a group/all of detectors at once by considering their historical causality [47] or each detector is treated separately and the prediction is made using its own history [48], [6]. In this paper, we will consider the second case where we predict the traffic flow of a single detector based on its own history. In this perspective, we do select the recent related work proposed by [47]. They make a comprehensive comparison between popular traffic flow prediction models using a specific loop detector. The intra-day de-trending (subtracting the traffic flow from its daily trend) model is proposed to remove the influence of the day-time on the prediction performance. Hence the prediction was made on the residual time series. Accordingly, this technique did improve the performance of all utilized techniques. However, since the main objective of OSA-CNN is to deal with dynamic changes and be self adaptive, we do use the original series without removing the daily trend effect. For the same reason, unlike them, we do consider the weekends and holidays.

1) Simulation and Training Setup: As in [47], we use measured traffic data provided by the PeMS dataset of the detector 1006210. Data are collected between October 1, 2009 and November 30, 2009. The raw data has a granularity equals to 30s which is then aggregated into four short-term levels: 3 -min, 5-min, 10-min and 15-min. In addition, two long-term levels are also considered: 30-min and 60-min. All three lanes of the selected detector are considered in the evaluation. As explained in Section IV-A, the original time series is interpolated into pulses and then using Eq.(3) the reference smooth time-series is constructed. The reference model parameters are presented in Table **??**. The resulting data set is then divided into three parts: 40% training, 10% validation and 50% testing set. In addition to one-step prediction, we consider up to five steps ahead. The simulation and the consecutive evaluation process are presented in Algorithm **??** which illustrates the OSA-CNN training and operating phases. The input of this algorithm is the traffic data divided into training, validation and testing sets. Also as inputs, we have the sparsity measure Ψ and the scaling factor f as predefined parameters. The output then is the performance of the test set. In the first three loops, we select the studied time series by selecting the time interval, prediction steps and the target detector. Next, two loops are used to get the best configuration of the OSA-CNN (CNN size + number of lags). Then, the training of OSA-CNN starts following the three training phases. The Best configuration is selected with respect to the best validation performance. The final step is to apply the selected configurations on the testing set and store the corresponding performance.

2) Performance Metrics: In order to make a comprehensive benchmarking, the performance metrics of the traffic flow time series are selected to be same as the ones used in [47]. These metrics are the mean relative error (MRE) [49], the mean squared error (MSE) [50] and the mean absolute error (MAE)

[51]. They are given by:

$$\text{MAE} = \frac{1}{T}\sum_{i=1}^{T} |v_{k_0}(i) - \hat{v}_{k_0}(i)| \tag{10}$$

$$\text{MRE} = \frac{100}{T}\sum_{i=1}^{T} \frac{|v_{k_0}(i) - \hat{v}_{k_0}(i)|}{v_{k_0}(i)} \tag{11}$$

$$\text{MSE} = \frac{1}{T}\sum_{ti=1}^{T} (v_{k_0}(ti) - \hat{v}_{k_0}(i))^2 \tag{12}$$

3) Results: We address the evaluation results in three contexts. These contexts are 1) the OSA-CNN short-term prediction performance (i.e. 3 to 15 min steps), 2) the OSA-CNN long-term prediction performance (i.e. 30 to 60 min steps), 3) the OSA-CNN multiple-steps prediction performance. The results with respect to each of the named contexts are discussed in the following:

a) For short term Traffic flow prediction, Chen (2012)[47] presented a remarkable comparison between six different prediction techniques combined with his proposed intraday trend model. For comparison purposes, we choose the best five methods. These methods are: Auto Regressive Moving Average Model (ARMA) [52], [53], Auto Regressive Integrated Moving Average Model (ARIMA) [4], [5], Support Vector Regression [54] [55], Bayesian Network [56] and Radial Basis Neural Network [57]. The evaluation is done over the aggregation levels 3-min, 5-min, 10-min and 15-min. The related comparison with I-CNN and OSA-CNN is presented in Tables II, III, IV, and V. The general conclusion is that I-CNN does over-perform the related methods by 0.14 to 4.08 % MRE. What is more interesting is that OSA-CNN does improve I-CNN performance by 0.8 to 4 % MRE. The comparison between OSA-CNN and I-CNN is more visible in Fig.8. An example of OSA-CNN control actions can be seen in Fig.6(a),6(b) where OSA-CNN outputs drive I-CNN to a better performance. In addition, Figures 7(a), 7(b), 7(c) and 7(d) show the OSA-CNN predicted values compared with the ground truth traffic flow of the first two days of November 2009 (1st is Sunday (Weekend) and the 2nd is a Monday (Working day)). OSA-CNN shows a capability of reliable prediction in both weekends and working days.

b) In the second evaluation context, both I-CNN and OSA-CNN are tested for long-term (30-min and 60-min) one-step prediction. As in the short-term prediction case, OSA-CNN did improve I-CNN outputs and gave a performance consistent with the short-term forecast one. Table VI presents the related performance values and Figures 7(e) and 7(f) shows the prediction graph of the same time period like for the previous context.

c) In the last context, we test OSA-CNN to do a multi-steps prediction (6 steps). A task which is considered as a challenge for traffic flow data. All presented aggregation levels are reused in this context to investigate the error propagation over time. Table VII and Fig 9 shows the related performance using the MRE metric. As a conclusion we can state that the error propagation trend is acceptable when compared to the one-step performance presented by Chen (2012)[47].

B. NN3 Time Series

The second candidate time-series for evaluation is the NN3 time-series competition dataset [46]. It consists of 111 time series with a one-month time interval and represents different practical business case studies. These case studies are categorized into four groups based on the length and seasonality of each instance. The number of instances in each group is presented in Table VIII. The objective of the related competition is making a multi-step prediction of the next 18 observations. A variety of researchers have investigated their proposed method using NN3. The best performance so far is recorded by [11]. In this paper, we will apply OSA-CNN on the 111 time series and benchmark the outcome with the related cutting-edge techniques.

1) Simulation and Training Setup: All provided time series are transformed into pulses and then using Eq.(3) the related reference series are constructed. Here also we use the reference model parameters presented in Table **??**. Since the objective of the NN3 competition is to predict the last 18 samples. Consider that the length of the studied series is equal to L, then we divide the data into three parts: the training part has the sample range $[1..(L-36)]$, the validation part has the sample range $[(L-35)..(L-18)]$ and the testing part with the sample range $[(L-17)..(L)]$. We consider here only the five steps ahead. The simulation and the related evaluation process are presented in Algorithm **??** which illustrates the OSA-CNN training and operating phases of the NN3 time-series case. The input of this algorithm is the NN3 111 time series divided into training, validation and testing sets. Also as inputs, we have the sparsity measure Ψ and the scaling factor f as predefined parameters. The output then is the performance of the test set. In the first loop, we select the studied time series by selecting its related index. Then, two loops are used to get the best configuration of the OSA-CNN (CNN size + number of lags). Next, the training of OSA-CNN starts following the three training phases. The Best configuration is selected with respect to the best validation performance. The final step is to apply the selected configurations on the testing set and store the corresponding performance value.

2) Performance Metrics: The NN3 competitors have used the Average Symmetric Mean Absolute Percentage Error (SMAPE) [58] as a main metric to be used in the competition. This metric is given by:

$$\text{SMAPE} = \frac{100}{T}\sum_{i=1}^{T} \frac{|v_{k_0}(i) - \hat{v}_{k_0}(i)|}{(v_{k_0}(i) + \hat{v}_{k_0}(i))/2} \tag{13}$$

3) Results: The main objective of the NN3 competition is to make an 18 months multi-steps prediction. This challenge attracted 60 competitors to test their respective methods. We do compare our method with 10 of those techniques including the one with the best performance so far, which is presented by Rahmann (2016) [11]. In this last named method he used a layered bagging of multi-layers neural

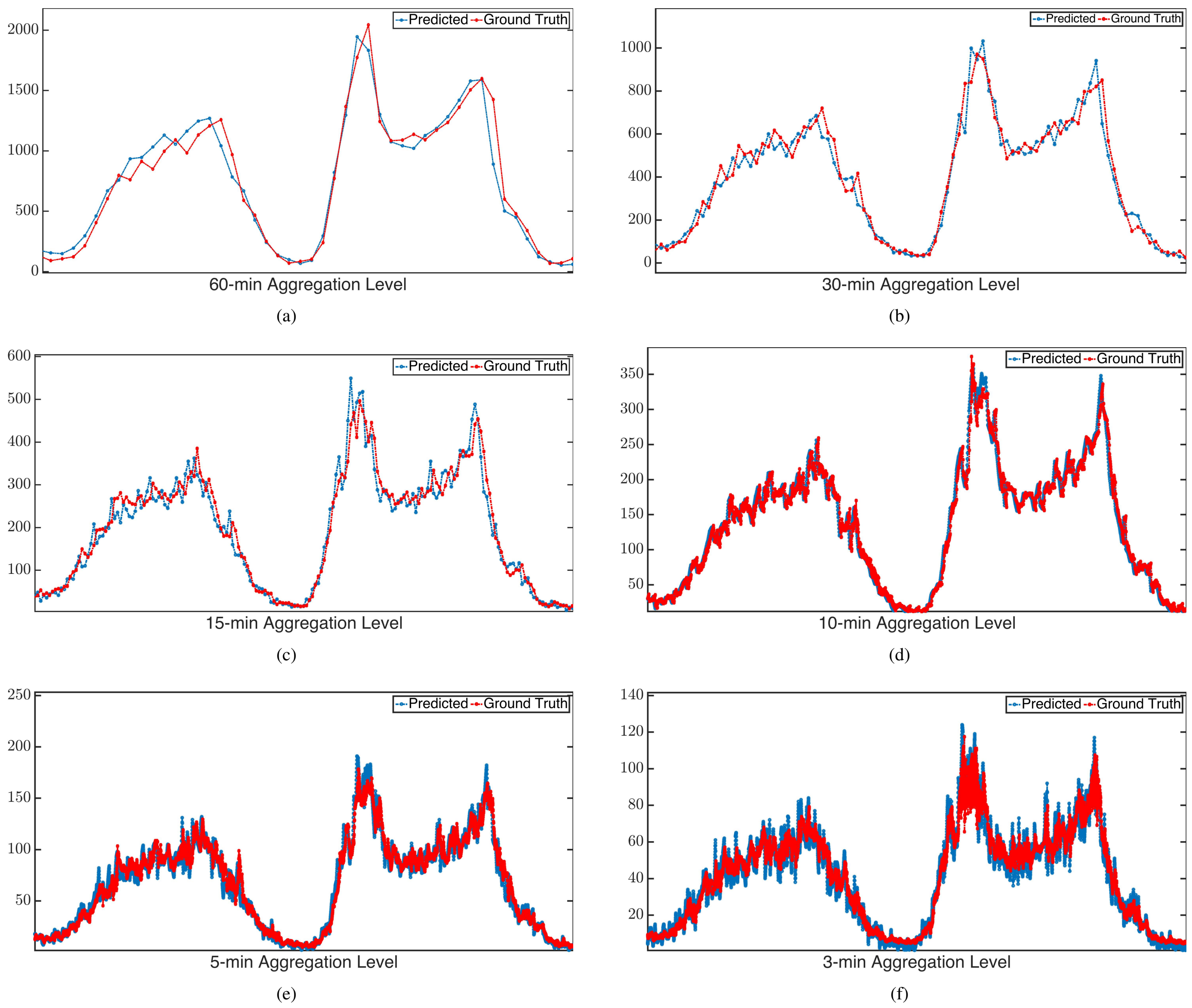


Fig. 7. The OSA-CNN Traffic flow prediction using the detector 1006210 from PEMS data set. It presents OSA-CNN predicted values (in blue) compared with the ground truth traffic flow (in red) of the first two days of November 2009 (1st is Sunday (Weekend) and the 2nd is a Monday (Working day)) using different aggregation levels

TABLE II
THE PERFORMANCE OF OSA-CNN AND I-CNN VERSUS OTHER MODELS, WITH 3-MIN AGGREGATION LEVEL. DATA USED IN THIS EVALUATION IS THE MEASURED TRAFFIC DATA PROVIDED BY THE PEMS DATASET OF THE DETECTOR 1006210. DATA ARE COLLECTED BETWEEN OCTOBER 1, 2009 AND NOVEMBER 30, 2009. THE RESULTS OF OTHER METHOD IS TAKEN FROM [47]

Method	MAD	MRE (%)	MSE	Parameters
ARMA	11.36	11,20	251.81	$p = 5, d = 0, q = 5$
ARIMA	11.38	11,23	252.84	$p = 4, d = 1, q = 5$
Support Vector Regression	11.40	11,30	251.89	
Bayesian Network	11.41	11,33	251.59	2 Gaussian mixtures
Radial Basis ANN	11.42	11.29	252.74	10 neurons
I-CNN	7.74	7.15	57.34	$h = 5, n = 100$
OSA-CNN	**5.35**	**5.34**	**28.53**	$h = 5, n = 100$

TABLE III
THE PERFORMANCE OF OSA-CNN AND I-CNN VERSUS OTHER MODELS, WITH 5-MIN AGGREGATION LEVEL. DATA USED IN THIS EVALUATION IS THE MEASURED TRAFFIC DATA PROVIDED BY THE PEMS DATASET OF THE DETECTOR 1006210. DATA ARE COLLECTED BETWEEN OCTOBER 1, 2009 AND NOVEMBER 30, 2009. THE RESULTS OF OTHER METHOD IS TAKEN FROM [47]

Method	MAD	MRE (%)	MSE	Parameters
ARMA	15.22	8,97	483.65	$p = 3, d = 0, q = 4$
ARIMA	15.31	9.01	489.99	$p = 5, d = 1, q = 5$
Support Vector Regression	15.41	9.15	494.79	
Bayesian Network	15.29	9.05	478.09	2 Gaussian mixtures
Radial Basis ANN	15.38	9.07	487.99	10 neurons
I-CNN	13.32	8.83	239.31	$h = 5, n = 100$
OSA-CNN	**9.8**	**4.83**	**162,009**	$h = 5, n = 100$

TABLE IV
THE PERFORMANCE OF OSA-CNN AND I-CNN VERSUS OTHER MODELS, WITH 10-MIN AGGREGATION LEVEL. DATA USED IN THIS EVALUATION IS THE MEASURED TRAFFIC DATA PROVIDED BY THE PEMS DATASET OF THE DETECTOR 1006210. DATA ARE COLLECTED BETWEEN OCTOBER 1, 2009 AND NOVEMBER 30, 2009. THE RESULTS OF OTHER METHOD IS TAKEN FROM [47]

Method	MAD	MRE (%)	MSE	Parameters
ARMA	23,86	6.99	1352.4	$p = 4, d = 0, q = 3$
ARIMA	23.84	6.97	1351.7	$p = 4, d = 1, q = 4$
Support Vector Regression	23.78	6,94	1338.3	
Bayesian Network	23.72	6.90	1316.2	2 Gaussian mixtures
Radial Basis ANN	24.11	6.98	1393.3	10 neurons
I-CNN	16.83	5.40	490,43	$h = 3, n = 100$
OSA-CNN	**13.74**	**3.52**	**363.64**	$h = 3, n = 100$

TABLE V
THE PERFORMANCE OF OSA-CNN AND I-CNN VERSUS OTHER MODELS, WITH 15-MIN AGGREGATION LEVEL. DATA USED IN THIS EVALUATION IS THE MEASURED TRAFFIC DATA PROVIDED BY THE PEMS DATASET OF THE DETECTOR 1006210. DATA ARE COLLECTED BETWEEN OCTOBER 1, 2009 AND NOVEMBER 30, 2009. THE RESULTS OF OTHER METHOD IS TAKEN FROM [47]

Method	MAD	MRE (%)	MSE	Parameters
ARMA	31.46	6.14	2582.5	$p = 3, d = 0, q = 3$
ARIMA	31.44	6.12	2580.9	$p = 3, d = 1, q = 4$
Support Vector Regression	31.01	5.96	2558.6	
Bayesian Network	30.75	5.90	2518.1	2 Gaussian mixtures
Radial Basis ANN	31.13	5.97	2613.5	10 neurons
I-CNN	26.21	5.46	2365.2	$h = 3, n = 100$
OSA-CNN	**23.55**	**4.60**	**1180,8**	$h = 3, n = 100$

TABLE VI
THE PERFORMANCE OF OSA-CNN AND I-CNN WITH 30-MIN AND 60-MIN AGGREGATION LEVEL. DATA USED IN THIS EVALUATION IS THE MEASURED TRAFFIC DATA PROVIDED BY THE PEMS DATASET OF THE DETECTOR 1006210. DATA ARE COLLECTED BETWEEN OCTOBER 1, 2009 AND NOVEMBER 30, 2009

Method	I-CNN			OSA-CNN			Parameters
Agg	MAE	MRE (%)	MSE	MAE	MRE (%)	MSE	
30 min	60.01	6.15	6167.3	**54.04**	**4.99**	**7837.9**	$h = 3, n = 100$
60 min	115.05	6.13	25153	**99.20**	**4.36**	**18807**	$h = 3, n = 50$

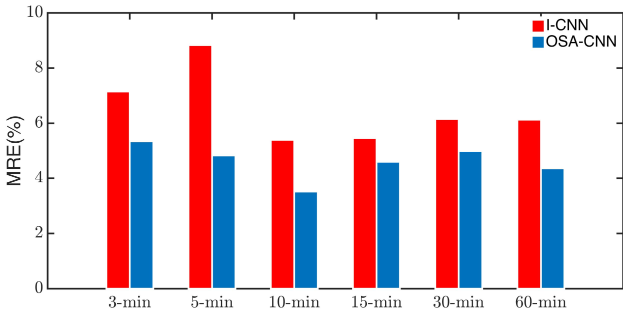


Fig. 8. The performance of OSA-CNN versus I-CNN for different aggregation levels

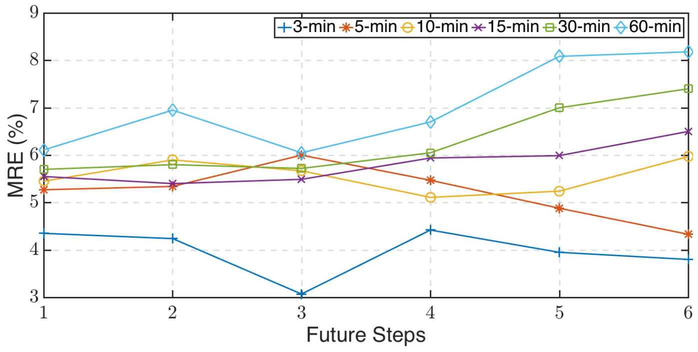
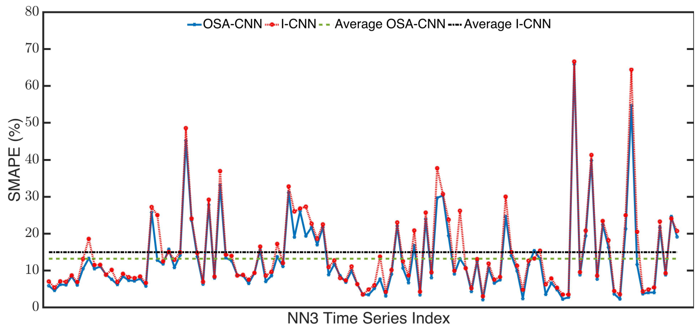
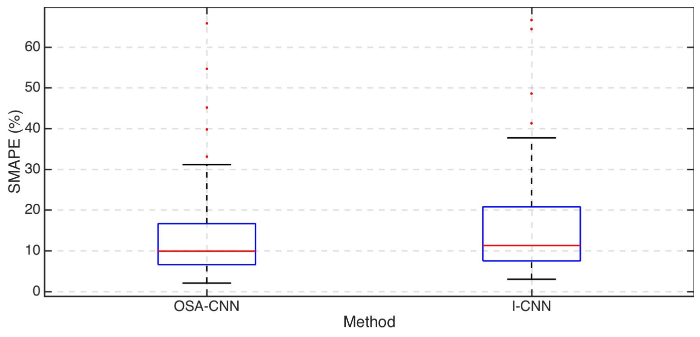


Fig. 9. The performance propagation of OSA-CNN for up to 6 future steps



(a)



(b)

Fig. 10. The OSA-CNN vs I-CNN performance of the 111 NN3 time series for 18 multi steps prediction case. (a) the SMAPE with respect to the time series index. It shows how OSA-CNN improved the reult of the I-CNN over all time series (b) the box distribution of SMAPE over the 111 series

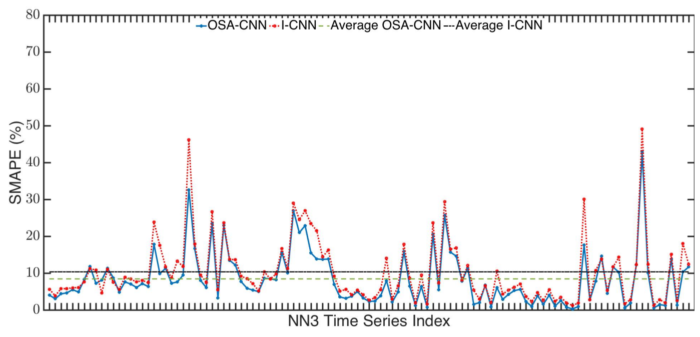
TABLE VII
THE MULTI-STEPS PERFORMANCE OF OSA-CNN OVER THE SHORT AND LONG TERM PREDICTION

Agg	MRE(%)				
	2 Steps	3 Steps	4 Steps	5 Steps	6 Steps
3-min	5.27	5.45	5.55	5.72	6.11
5-min	5.34	5.59	5.34	5.88	6.95
10-min	6.01	5.67	5.49	5.72	6.05
15-min	5.47	5.11	5.95	6.05	6.7
30-min	4.88	5.24	5.99	7.13	8.04
60-min	4.33	5.97	6.55	7.41	8.18

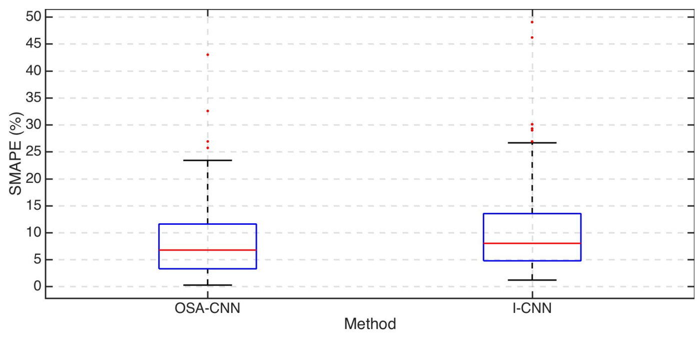
TABLE VIII
THE NUMBER OF NN3 FORECASTING COMPETITION TIME SERIES WITH RESPECT TO THE LENGTH AND SEASONALITY OF EACH TIME SERIES

	short	long	Total
seasonal	25	29	54
non-seasonal	25	32	57
Total	50	61	111

networks. The remaining 9 methods are addressed in [59] and they do combine computational intelligence methods with statical methods. These methods are: A Gaussian Process combined with ANN [60] , K nearest neighbors (KNN) [61], Ensemble Regression Tree [62] Automated Linear Model with Self Adaptive Genetic Algorithm [63], Generalized Regression Neural Network (GRNN) [10], Echo State Network [14] and Adaptive multi-step-ahead out-of-sample forecasting combination approach [64]. The performance of OSA-CNN and I-CNN when compared to these 9 models is presented in Fig. 12. As it is shown, I-CNN records an average SAMPE performacne equal to 14.95 (%) when compared to the best performance by Rhmann(2016) that was of 14.56 (%). However, OSA-CNN showed a remarkable outperformance by recording an average SMAPE equal to **13.8** (%). The OSA-CNN and I-CNN SMAPE values for each of the 111 time series can be seen in Figures 10(a) and 10(b). In these figures, one does clearly see the improvement of performance that OSA-CNN did realize compared to I-CNN starting from 0.0062 (%) at the series number 56 to 12.98 (%) at the series number 73. In addition to the multi-step prediction, we also investigated the performance of our concept for one-step prediction over again all the 111 series. In this case, I-CNN gives an average SMAPE equal to 10.33 (%) decreased by OSA-CNN to 8.43 (%). The related performance of individual series and the related error distributions can be seen in Figures 11(a) and 11(b).



(a)



(b)

Fig. 11. The OSA-CNN vs I-CNN performance of the 111 NN3 time series for one step prediction case. (a) the SMAPE with respect to the time series index. It shows how OSA-CNN improved the reulst of the I-CNN over all time series (b) the box distribution of SMAPE over the 111 series

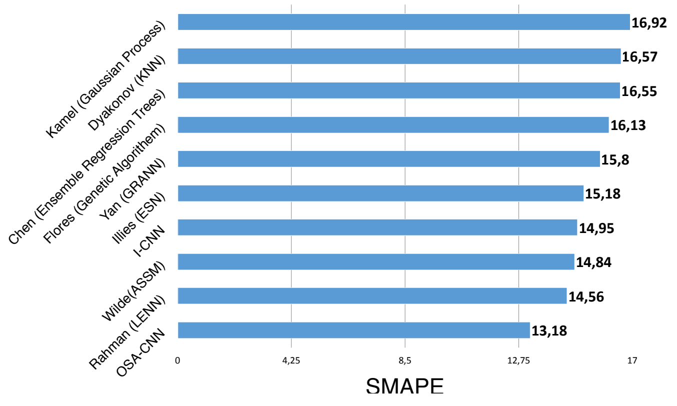


Fig. 12. The comparison between OSA-CNN against the related state of the art methods over the 111 time series of NN3 time series competition. OSA-CNN proved to be an robust TSF algorithm with 13.18 % comparing to other methods

VII. Conclusion

The universality of a time-series prediction model is a critical drawback in most recent research. The low generalization performance is mainly caused by a newly emerged dynamic/behavior which is not observed in the training process. Therefore, a self-adaptive model is needed. In the human mind, such an adaptive model is realized by two thinking systems as the Nobel Prize winner Daniel Kahneman addressed in his book *Thinking, Fast and Slow* thoughts [15]. The first biological system is fast, intuitive, and operates when an experienced task is required. In contrast, the second system is a slow, calculation-based, decision-making system and operates when a new emerging task is presented. The second system controls the actions of the first system by either modifying or trusting them. This fact inspired us to build an artificial neuron model that consists of two neuron systems acting similarly to the biological one.

Echo State Network (ESN) has been presented recently as a very promising neurocomputing model for time-series modeling. Similarly, Cellular Neural Network (CNN) can be considered a continuous model of ESN (if it's trained in the same way). Accordingly, the two connected systems are realized by two connected CNNs; we call it OSA-CNN. System 1 of OSA-CNN predicts the future value of the time-series. Also, System 2 of OSA-CNN controls System 1 and drives it to deliver a better performance. The control action is based on the recent prediction error and modeled as a reference control model. The modeling, training, and validating of OSA-CNN is presented extensively. OSA-CNN is benchmarked with the best state-of-the-art methods. Multiple examples of time-series are considered: real field traffic flow data obtained from the "PeMs traffic database" and 111 time series examples offered by the "NN3 competitions." The novel OSA-CNN concept highly outperforms the state-of-the-art competing methods.

References

[1] G. Zhang, B. E. Patuwo, and M. Y. Hu, "Forecasting with artificial neural networks:: The state of the art," *International journal of forecasting*, vol. 14, no. 1, pp. 35–62, 1998.

[2] S. F. Crone and P. C. Graffeille, "An evaluation framework for publications on artificial neural networks in sales forecasting." in *IC-AI*, 2004, pp. 221–227.

[3] G. P. Zhang and D. M. Kline, "Quarterly time-series forecasting with neural networks," *Neural Networks, IEEE Transactions on*, vol. 18, no. 6, pp. 1800–1814, 2007.

[4] M. S. Ahmed and A. R. Cook, *Analysis of freeway traffic time-series data by using Box-Jenkins techniques*, 1979, no. 722.

[5] T. Mills, "Time series techniques for economistscambridge university press," *Cambridge et al*, 1990.

[6] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: A deep learning approach," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 16, no. 2, pp. 865–873, 2015.

[7] B. C. Pijanowski, D. G. Brown, B. A. Shellito, and G. A. Manik, "Using neural networks and gis to forecast land use changes: a land transformation model," *Computers, environment and urban systems*, vol. 26, no. 6, pp. 553–575, 2002.

[8] D. S. Pereira Salazar, P. J. Leitao Adeodato, and A. Lucena Arnaud, "Continuous dynamical combination of short and long-term forecasts for nonstationary time series," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 25, no. 1, pp. 241–246, 2014.

[9] X. Li and A. G.-O. Yeh, "Neural-network-based cellular automata for simulating multiple land use changes using gis," *International Journal of Geographical Information Science*, vol. 16, no. 4, pp. 323–343, 2002.

[10] W. Yan, "Toward automatic time-series forecasting using neural networks," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 23, no. 7, pp. 1028–1039, 2012.

[11] M. Rahman, M. Islam, K. Murase, X. Yao *et al.*, "Layered ensemble architecture for time series forecasting," 2016.

[12] H. Jaeger, "The ?echo state? approach to analysing and training recurrent neural networks-with an erratum note," *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, vol. 148, p. 34, 2001.

[13] M. Lukoševičius, "A practical guide to applying echo state networks," in *Neural Networks: Tricks of the Trade*. Springer, 2012, pp. 659–686.

[14] I. Ilies, H. Jaeger, O. Kosuchinas, M. Rincon, V. Sakenas, and N. Vaskevicius, "Stepping forward through echoes of the past: forecasting with echo state networks," *Short report on the winning entry to the NN3 financial forecasting competition, available online at http://www. neural-forecasting-competition. com/downloads/NN3/methods/27-NN3 Herbert Jaeger report. pdf*, 2007.

[15] D. Kahneman, *Thinking, fast and slow*. Macmillan, 2011.

[16] L. Chua and L. Yang, "Cellular neural networks: theory," *Circuits and Systems, IEEE Transactions on*, vol. 35, no. 10, pp. 1257–1272, 1988.

[17] L. O. Chua and L. Yang, "Cellular neural networks: Applications," *Circuits and Systems, IEEE Transactions on*, vol. 35, no. 10, pp. 1273–1290, 1988.

[18] T. Roska and L. O. Chua, "The cnn universal machine: an analogic array computer," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol. 40, no. 3, pp. 163–173, 1993.

[19] P. Kinget and M. S. Steyaert, "A programmable analog cellular neural network cmos chip for high speed image processing," *Solid-State Circuits, IEEE Journal of*, vol. 30, no. 3, pp. 235–243, 1995.

[20] Z. Nagy and P. Szolgay, "Configurable multilayer cnn-um emulator on fpga," *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, vol. 50, no. 6, pp. 774–778, 2003.

[21] T.-Y. Ho, P.-M. Lam, and C.-S. Leung, "Parallelization of cellular neural networks on gpu," *Pattern Recognition*, vol. 41, no. 8, pp. 2684–2692, 2008.

[22] P. Dudek and P. J. Hicks, "A cmos general-purpose sampled-data analog processing element," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol. 47, no. 5, pp. 467–473, 2000.

[23] H. Harrer and J. A. Nossek, "Discrete-time cellular neural networks," *International Journal of Circuit Theory and Applications*, vol. 20, no. 5, pp. 453–467, 1992. [Online]. Available: http://dx.doi.org/10.1002/cta.4490200503

[24] G. Destri, "Discrete-time cellular neural network construction through evolution programs," in *Cellular Neural Networks and their Applications, 1996. CNNA-96. Proceedings., 1996 Fourth IEEE International Workshop on*, Jun 1996, pp. 473–478.

[25] V. Kasparian and C. Batur, "Model reference based neural network adaptive controller," *ISA transactions*, vol. 37, no. 1, pp. 21–39, 1998.

[26] V. Kasparian, C. Batur, H. Zhang, and J. Padovan, "Davidon least squares-based learning algorithm for feedforward neural networks," *Neural networks*, vol. 7, no. 4, pp. 661–670, 1994.

[27] W. C. Davidon, "New least-square algorithms," *Journal of Optimization Theory and Applications*, vol. 18, no. 2, pp. 187–197, 1976.

[28] F. Rossomando, C. Soria, D. Patiño, and R. Carelli, "Model reference adaptive control for mobile robots in trajectory tracking using radial basis function neural networks," *Latin American applied research*, vol. 41, no. 2, pp. 177–182, 2011.

[29] S. Wei, "A model reference-based adaptive pid controller for robot motion control of not explicitly known systems," *International Journal of Intelligent Control and Systems*, vol. 12, no. 3, pp. 237–244, 2007.

[30] S. G. Shuzhi, T. H. Lee, and C. J. Harris, *Adaptive neural network control of robotic manipulators*. World Scientific, 1998, vol. 19.

[31] H. A. Kingravi, G. Chowdhary, P. A. Vela, and E. N. Johnson, "Reproducing kernel hilbert space approach for the online update of radial bases in neuro-adaptive control," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 23, no. 7, pp. 1130–1141, 2012.

[32] MATLAB, *version 7.14.0 (R2013b)*. Natick, Massachusetts: The MathWorks Inc., 2013.

[33] E. Hairer, S. NORSETT, and G. Wanner, *Solving Ordinary, Differential Equations I, Nonstiff problems/E. Hairer, SP Norsett, G. Wanner, with 135 Figures, Vol.: 1*. 2Ed. Springer-Verlag, 2000, 2000.

[34] B. Mirzai, Z. Cheng, and G. S. Moschytz, "Learning algorithms for cellular neural networks," in *IEEE International Symposium On Circuits And Systems*. INSTITUTE OF ELECTRICAL ENGINEERS INC (IEEE), 1998, pp. 159–162.

[35] A. Fasih, J. C. Chedjou, and K. Kyamakya, "Cellular neural network trainer and template optimization for advanced robot locomotion, based on genetic algorithm," in *Mechatronics and Machine Vision in Practice, 2008. M2VIP 2008. 15th International Conference on*. IEEE, 2008, pp. 317–322.

[36] S. SanthoshKumar, J. Vignesh, L. Rangarajan, V. Narayanan, K. Rangarajan, and A. Venkatkrishna, "A fast time scale genetic algorithm based image segmentation using cellular neural networks (cnn)," in *Signal Processing and Communications, 2007. ICSPC 2007. IEEE International Conference on*. IEEE, 2007, pp. 908–911.

[37] H.-L. Wei and S. A. Billings, "Generalized cellular neural networks (gcnns) constructed using particle swarm optimization for spatio-temporal evolutionary pattern identification," *International journal of Bifurcation and Chaos*, vol. 18, no. 12, pp. 3611–3624, 2008.

[38] T.-J. Su, J.-C. Cheng, and Y.-D. Sun, "Particle swarm optimization with time-varying acceleration coefficients based on cellular neural network for color image noise cancellation," in *ICDT 2011, The Sixth International Conference on Digital Telecommunications*, 2011, pp. 109–115.

[39] P. A. Mastorocostas, "Resilient back propagation learning algorithm for recurrent fuzzy neural networks," *Electronics Letters*, vol. 40, no. 1, pp. 57–58, Jan 2004.

[40] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov 1997.

[41] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*. IEEE, 1998, pp. 69–73.

[42] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 3. IEEE, 1999.

[43] A. Ratnaweera, S. Halgamuge, and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *Evolutionary Computation, IEEE Transactions on*, vol. 8, no. 3, pp. 240–255, 2004.

[44] G. Fornarelli and A. Giaquinto, "Adaptive particle swarm optimization for {CNN} associative memories design," *Neurocomputing*, vol. 72, no. 16?18, pp. 3851 – 3862, 2009, financial Engineering Computational and Ambient Intelligence (IWANN 2007). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231209001684

[45] Caltrans, "Performance measurement system (pems)." [Online]. Available: http://pems.dot.ca.gov

[46] "Time series forecasting competition for neural network and computational intelligence." [Online]. Available: http://www.neural-forecasting-competition.com/NN3/

[47] C. Chen, Y. Wang, L. Li, J. Hu, and Z. Zhang, "The retrieval of intra-day trend and its influence on traffic prediction," *Transportation research part C: emerging technologies*, vol. 22, pp. 103–118, 2012.

[48] L. Li, X. Su, Y. Wang, Y. Lin, Z. Li, and Y. Li, "Robust causal dependence mining in big data network and its application to traffic flow predictions," *Transportation Research Part C: Emerging Technologies*, vol. 58, pp. 292–307, 2015.

[49] C. Tofallis, "A better measure of relative prediction accuracy for model selection and model estimation," *Journal of the Operational Research Society*, vol. 66, no. 8, pp. 1352–1362, 2014.

[50] E. L. Lehmann and G. Casella, "Theory of point estimation (springer texts in statistics)," 1998.

[51] R. Geary, "The ratio of the mean deviation to the standard deviation as a test of normality," *Biometrika*, vol. 27, no. 3/4, pp. 310–332, 1935.

[52] W. A. Fuller, *Introduction to statistical time series*. John Wiley & Sons, 2009, vol. 428.

[53] P. Whitle, *Hypothesis testing in time series analysis*. Almqvist & Wiksells, 1951, vol. 4.

[54] W. K. Härdle, M. Müller, S. Sperlich, and A. Werwatz, *Nonparametric and semiparametric models*. Springer Science & Business Media, 2012.

[55] K. Takezawa, *Introduction to nonparametric regression*. John Wiley & Sons, 2005, vol. 606.

[56] A. Darwiche, *Modeling and reasoning with Bayesian networks*. Cambridge University Press, 2009.

[57] W. Yang, D. Yang, Y. Zhao, and J. Gong, "Traffic flow prediction based on wavelet transform and radial basis function network," in *Logistics Systems and Intelligent Management, 2010 International Conference on*, vol. 2. IEEE, 2010, pp. 969–972.

[58] S. Makridakis, C. Chatfield, M. Hibon, M. Lawrence, T. Mills, K. Ord, and L. F. Simmons, "The m2-competition: A real-time judgmentally based forecasting study," *International Journal of Forecasting*, vol. 9, no. 1, pp. 5–22, 1993.

[59] S. F. Crone, M. Hibon, and K. Nikolopoulos, "Advances in forecasting with neural networks? empirical evidence from the nn3 competition on time series prediction," *International Journal of Forecasting*, vol. 27, no. 3, pp. 635–660, 2011.

[60] N. K. Ahmed, A. F. Atiya, N. El Gayar, and H. El-Shishiny, "A combined neural network/gaussian process regression time series forecasting system for the nn3 competition."

[61] G. D?yakonov Alexander, "Simple knn-method for times series prediction."

[62] H. Chen and X. Yao, "Ensemble regression trees for time series predicitions," *methods*, vol. 11, p. 6, 2007.

[63] P. Flores, C. Anaya, H. M. Ramírez, and L. B. Morales, "Automated linear modeling of time series with self adaptive genetic algorithms," in *Neural Networks, 2007. IJCNN 2007. International Joint Conference on*. IEEE, 2007, pp. 1389–1396.

[64] S. B. Taieb, G. Bontempi, A. F. Atiya, and A. Sorjamaa, "A review and comparison of strategies for multi-step ahead time series forecasting based on the nn5 forecasting competition," *Expert systems with applications*, vol. 39, no. 8, pp. 7067–7083, 2012.

Ahmad Haj Moas received his B.S. degree in mechatronics engineering from Tishreen University, Lattakia, Syria, in 2008. He received his M.S. degree in health care IT from from Carinthia University of Applied Science, Klagenfurt, Austria, in 2012 and is currently a Ph.D. candidate, a university assistant, and a lecturer at the Institute for Smart System Technology (IST), Alpen-Adria University of Klagenfurt in Austria. His research interests include machine vision, machine learning, applied mathematics, and neurocomputing. He developed a variety of methods in the scope of human-machine interaction and pattern recognition. He has participated in several projects the field of transportation informatics, and the outcomes were patented

Kyandoghere Kyamakya obtained the Ingnieur Civil degree in Electrical Engineering in 1990 at the University of Kinshasa, DR Congo. In 1999 he received his Doctorate in Electrical Engineering at the University of Hagen in Germany. He then worked three years as post-doctorate researcher at the Leibniz University of Hannover in the field of Mobility Management in Wireless Networks. From 2002 to 2005 he was junior professor for Positioning and Location Based Services at Leibniz University of Hannover. Since 2005 he is full Professor for Transportation Informatics and Director of the Institute for Smart Systems Technologies at the University of Klagenfurt in Austria. He is actively conducting research involving modeling, simulation, and test-bed evaluations for a series of concepts in the frame of the application of information and communication technology in transportation. Three main transportation systems are addressed: (a) Intelligent Transportation Systems; (b) Intelligent Vehicles and Mobile Robotics Systems; and (c) Intelligent Logistics and Supply Chains. In the research addressing these systems, a series of fundamental and theoretical tools from the fields of applied mathematics, electronics, and computer science are either extensively exploited or are source of inspiration for innovative solutions and concepts: nonlinear dynamics and synchronization, cellular neural networks, nonlinear image processing, cellular-neural-networks-based analog computing, systems science, and computational intelligence

Mouhannad Ali is currently a PhD Student and Research Assistant in the Institute for Smart System Technology (IST), Alpen-Adria University of Klagenfurt in Austria. He received his BSc in information technology (2008) from Virtual Syrian University in Syria, and diploma in Health Care IT (2012) from Carinthia University of Applied Science of Klagenfurt in Austrian. His research interests include Computer Vision, Machine Learning and Data Mining

Fadi Al Machot studied computer science at the University of Potsdam, Germany between 2005 and 2010. He received his degree Diploma in the field of artificial intelligence. He also focused on the field of data mining and computer vision. In November 2013 he finished his PhD at Klagenfurt University. He was a research member in different European projects. In 2011 his SRSnet project was awarded as one of the best practice projects in the Interreg IV program. As a researcher, he developed different algorithms in the areas of complex event detection in video surveillance systems, advanced driver assistance systems and human cognitive reasoning. His work was published and patented in different international conferences and Journals. He is currently a lecturer and a senior researcher at Klagenfurt University in the application engineering research group.