Available online at www.sciencedirect.com

ScienceDirect

Cognitive Systems Research 50 (2018) 206–213

Cognitive Systems RESEARCH

www.elsevier.com/locate/cogsys

Fuzzy deep learning based urban traffic incident detection

Chaimae El Hatri [*], Jaouad Boumhidi

Computer Science Department, Sidi Mohamed Ben AbdEllah University, LIIAN Laboratory, Faculty of Sciences Dhar-Mahraz, Fez 30000, Morocco

Received 13 April 2017; received in revised form 7 December 2017; accepted 11 December 2017
Available online 15 December 2017

Abstract

Traffic incident detection (TID) is an important part of any modern traffic control because it offers an opportunity to maximise road system performance. For the complexity and the nonlinear characteristics of traffic incidents, this paper proposes a novel fuzzy deep learning based TID method which considers the spatial and temporal correlations of traffic flow inherently. Parameters of the deep network are initialized using a Stacked Auto-Encoder (SAE) model following a layer by layer pre-training procedure. To conduct the fine tuning step, the back-propagation algorithm is used to precisely adjust the parameters in the deep network. Fuzzy logic is employed to control the learning parameters where the objective is to reduce the possibility of overshooting during the learning process, increase the convergence speed and minimize the error. To find the best architecture of the deep network, we used a separate validation set to evaluate different architectures generated randomly based on the Mean Squared Error (MSE). Simulation results show that the proposed incident detection method has many advantages such as higher detection rate and lower false alarm rate.
© 2017 Published by Elsevier B.V.

Keywords: Automatic traffic incident detection; Fuzzy deep learning; Deep artificial neural network; Stacked auto-encoder model; Back-propagation algorithm; Fuzzy logic controller

1. Introduction

Traffic congestion is a key problem to cause driver frustration and air pollution in today's urban area (El Hatri & Boumhidi, 2017). One of the important parts of intelligent transportation systems is incident detection. Traffic incidents are the main cause of congestion, which subsequently increases travel delay and fuel consumption in major cities. Traffic incidents usually cannot be predicted which poses great challenge to traffic management centers. Therefore, early detection of incidents reduces the delay experienced by road users, fuel consumptions, gas emissions, and the probability of other collisions, as well as improves road safety and real-time traffic control (El Hatri, Tahifa, & Boumhidi, 2016). This paper focuses on developing an intelligent system able to determine the presence of an incident by using real-time traffic data. The objective is to help the officer manage and take action to clear the roadway as early as possible and to ensure that traffic return to normal safety conditions.

For the aforementioned reasons, automatic traffic incident detection techniques have been investigated quite a lot in the recent years. A number of incident detection approaches based on traffic behavior or mathematical models have been proposed for this task. However, earlier incident detection methods are limited in distinguishing recurrent and non-recurrent congestions, and the complexity of current approaches makes them insufficient to handle the real time task. This paper presents a new approach for detecting traffic incident, and compares its performance with established techniques. Different from traditional incident detection methods, both spatial and temporal traffic

* Corresponding author.
E-mail addresses: chaimae.elhatri@usmba.ac.ma (C. El Hatri), jaouad.boumhidi@usmba.ac.ma (J. Boumhidi).

https://doi.org/10.1016/j.cogsys.2017.12.002
1389-0417/© 2017 Published by Elsevier B.V.

information are considered to find the potential incidents. Meanwhile, adaptive learning ability and short detection response time are achieved in the proposed method. Traffic incident detection can be viewed as a pattern recognition problem. The proposed method is based on fuzzy deep neural network learning. We selected this technique because it performs well to a wide range of classification problems and it is fairly robust to irrelevant features. The rest of the paper is organized as follows: the related works are discussed in Section 2. Fuzzy deep learning based approach is presented in Section 3. Network configuration and measures to evaluate detection performance are discussed in Section Section 4. Simulation results and analysis are given in Section Section 5. Finally, the paper is concluded in Section Section 6.

2. Related works

The problem of TID can be regarded as a task of classification, to determine whether or not an incident happens according to data gathered from traffic flow. Various automatic incident detection techniques have been released to address this problem, such as Support Vector Machine (SVM) (Yuan & Cheu, 2003), Fuzzy Logic (FL) (Rossi, Gastaldi, Gecchele, & Barbaro, 2015), and Back-Propagation Neural Network (BPNN) (Lu, Chen, Wang, & van Zuylen, 2012). All of the works cited suffer from drawbacks. The limitation of SVM comes from the choice of kernel function, the determination and tuning of several parameters. FL is based on human knowledge by determining fuzzy rules which are often set manually by experts. Thus, the detection performance is affected by subjective decisions. The Back-Propagation (BP) algorithm suffers from slow convergence and the possibility of getting caught in the local minimum. In any event, it is difficult to say that one method is clearly superior ever other methods in any situation. One reason for this is that the proposed models are developed with a small amount of specific data. The accuracy of TID methods is dependent of the traffic flow features embedded in the collected spatiotemporal traffic data. Thus we propose a novel TID method which considers the spatial and temporal correlations of traffic flow inherently. In general, the use of intelligent classification offers helpful techniques to deal with this kind of problem. Artificial intelligence refers to a set of procedures that apply reasoning and uncertainty in complex decision making and data analysis processes (Rossi, Gastaldi, Gecchele, & Barbaro, 2015). Among artificial intelligence methods, Artificial Neural Networks (ANNs) have been widely used in various research areas to solve problems including classification, traffic incident detection, function approximation, optimization, prediction, and so on (Chakraborty, 2012; de Oliveira & de Almeida Neto, 2014). ANN possesses a variety of alternative features such as massive parallelism, distributed representation and computation, generalization ability, adaptability and inherent contextual information processing (Azar, 2013). Several studies confirmed that neural network models can provide fast and reliable incident detection for several reasons. First, automatic incident detection can be cast as a pattern recognition problem. Neural networks are known to solve pattern recognition problems effectively (Kumar Basu, Bhattacharyya, & Kim, 2010). Second, neural networks are suitable for solving the problem when there is no mathematical model or explicit rules.

There are a variety of types of ANNs. Traditional process neural network is usually limited in the structure of single hidden layer. ANN with multiple layers was examined not as effective as ANN with single hidden layer in many applications. It is because the structure of process neural network would be very complex when extended to multiple hidden layers. However, for complex detection systems with rich amount of data, one single hidden layer usually would be not enough in describing complicated relations between inputs and outputs. Complexity theory of circuits strongly suggests that deep architectures can be much more efficient than shallow architectures, in terms of computational elements and parameters required to represent some functions. Deep learning is a type of machine learning method. It has been introduced with the objective of moving machine learning closer to one of its original goals: artificial intelligence. The concept of deep learning is derived from neural network research, therefore deep learning regarded as a new generation of neural networks (Chakraborty, 2012). It use multiple-layer architectures or deep architecture to extract inherent features in data from the lowest level to the highest level, and they can discover huge amounts of structure in the data. Recently, deep learning had attracted a lot of attention from both academic and industrial communities. It has been applied with success in classification tasks, natural language processing, object detection, traffic data prediction and so on (Deng, Ren, Kong, Bao, & Dai, 2016; Deng & Yu, 2013; Lv, Duan, Kang, Li, & Wang, 2015; Shin, Orton, Collins, Doran, & Leach, 2013). As traffic incident is complicated in nature, deep learning algorithms can represent traffic features without prior knowledge, which may have good performance for traffic incident detection.

The main contributions of this paper are the following. First, we present a novel fuzzy deep learning based incident detection method. Second, we propose the use of a stacked auto-encoder model following a layer by layer pre-training procedure to represent traffic flow features for incident detection. Third, to conduct the fine tuning step and adjust the parameters of the deep network, we propose the use of the back-propagation algorithm. Fourth, since fuzzy logic systems have demonstrated their ability in many applications, especially for the control of complex nonlinear systems that are difficult to model analytically (Azar, 2012), we propose the use of the fuzzy logic approach to adaptively vary the learning parameters where the objective is to reduce the possibility of overshooting during the learning process and help the deep network get out of a local minimum. The simulation results show that the proposed fuzzy deep learning based TID method has the advantages of high detection and classification rates and low false alarm rate.

3. Methodology

3.1. The structure of stacked auto-encoders

An Auto-Encoder (AE) is a neural network with only one hidden layer and with the same number of nodes in the input and output layers. Given a set of training samples $\{x^{(1)}, x^{(2)}, x^{(3)}, \ldots\}$, where $x^{(i)} \in R^d$, an AE first encodes an input $x^{(i)}$ to a hidden representation $y(x^{(i)})$ computed by Eq. (1), and then decodes representation $y(x^{(i)})$ back into a reconstruction $z(x^{(i)})$ computed by Eq. (2) (Huand, Hong, Song, & Xie, 2014).

$$y(x) = f(W_1 \cdot x + b_1) \tag{1}$$

$$z(x) = g(W_2 \cdot y(x) + b_2) \tag{2}$$

where W_1 is a weight matrix, b_1 is an encoding bias vector, W_2 is a decoding matrix, and b_2 is a decoding bias vector. $f(x)$ and $g(x)$ are the activation functions. The auto-encoder can be trained using the conventional back-propagation algorithm. Training of auto-encoder takes each input sample itself as its label. The target is to minimize the reconstruction error $E(X,Z)$ defined by Eq. (3).

$$E(X,Z) = \frac{1}{2}\sum_{i=1}^{N} ||x^i - z^i||^2 \tag{3}$$

where X is the set of N input samples $\{x^{(1)}, x^{(2)}, \ldots, x^{(N)}\}$, and Z is the set of corresponding reconstructed output $\{z^{(1)}, z^{(2)}, \ldots, z^{(N)}\}$. And the notation $\|\cdot\|$ represents the 2-norm of a vector.

A SAE model is created by stacking auto-encoders to form a deep neural network. As shown in Fig. 1, considering a SAE with three layers and given a set of input samples, the training of a SAE is straightforward. The first layer is trained as an auto-encoder, with the training set as inputs such that W_1^1 and W_2^1 are obtained. Then for each of others auto-encoders, the encoding of the first AE is taken as the input to train the next one such that W_1^i and W_2^i are obtained. In this way, the training of multiple auto-encoders is completed.

3.2. Back-propagation algorithm

Fine tuning is a strategy that is commonly found in deep learning. It can be used to greatly improve the performance of a stacked auto-encoders. From a high level perspective, fine tuning treats all layers of SAE as a single model, so that in one iteration, we are improving upon all the weights. As the back-propagation algorithm can be extended to apply for an arbitrary number of layers, we can actually use this algorithm on stacked auto-encoders of arbitrary depth. In this work, to adapt the connections

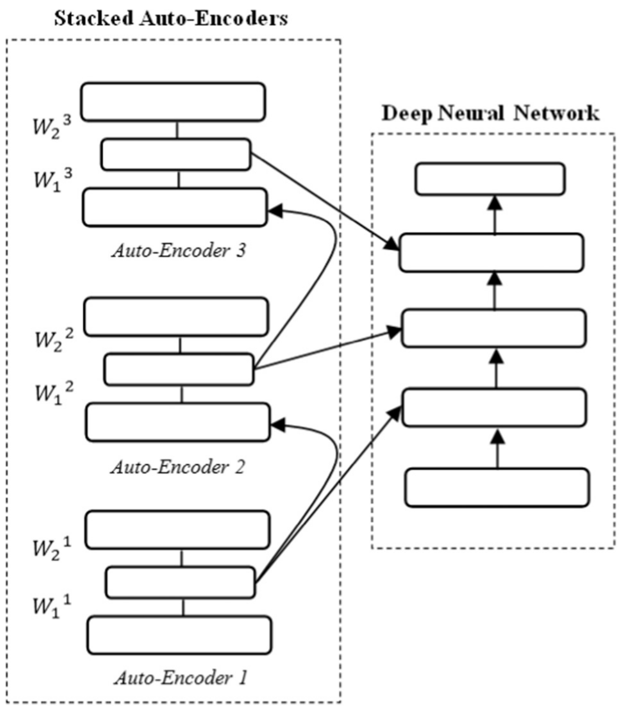

Fig. 1. The structure of a stacked auto-encoders.

weights in order to obtain minimal difference between the network output and the desired output, we used the back-propagation algorithm. It was developed by Paul Werbos in 1974. BP algorithm is quite simple; output of NN is evaluated against desired output. If results are not satisfactory, connection between layers are modified and process is repeated again until error is small enough. The objective of BP method is adapting synaptic weights in order to minimize an error function. The approach most commonly used for the minimization of the error function is based on the gradient method. The error function is defined by Eq. (4).

$$E(t) = \frac{1}{2}|e(t)|^2 \tag{4}$$

where $e(t)$ is the error value, i.e. the difference between the output and the estimated output. This difference is used to change the connection weights between neuron in the network according to Eq. (5).

$$\Delta w_j(t+1) = \eta\left(\frac{\partial E(t)}{\partial w_j}\right) + \alpha \Delta w_j(t) \tag{5}$$

where η is the learning rate parameter and the positive constant α is a momentum factor. Neural networks are sensitive to the selection of the learning rate and the momentum value. The learning rate can limit or expand the extent of weight adjustment in a learning cycle. A higher learning rate can make the network unstable and can cripple the network prediction ability. In the contrary if the learning rate is low, the training time of the network is increased. In the other hand, low momentum causes weight oscillations and instability and thus preventing the network from learning. High momentum can cripple the network adaptability. During the middle of training, when steep slope occurs, a small momentum value is recommended. However, during the end of training, large momentum value is desirable. Therefore, for the best results, it is necessary that these two parameters should be adjusted adaptively during the learning process, since no single value is optimal for all dimensions. In the next part, the implementation of fuzzy control system to adaptively determine the learning rate and momentum value is discussed.

3.3. Fuzzy logic controlled deep neural network

A fuzzy logic control is a technique useful in representing human knowledge in a specific domain of application and in reasoning with that knowledge to make useful inferences or actions (Eze, Emmanuel, & Stephen, 2014). A fuzzy logic control system consists of four components. A fuzzifier converts data into fuzzy data or Membership Functions (MFs). The fuzzy rule base contains the relations between the input and output. The fuzzy inference process combines MFs with the control rules to derive the fuzzy output, and the defuzzifier converts the fuzzy numbers back to a crisp value. There are two reasons that fuzzy logic systems are preferred: fuzzy systems are suitable for uncertain or approximate reasoning and they allow decision making with estimated values under incomplete or uncertain information. By employing a fuzzy control system to adaptively adjust the learning parameters of the neural network according to the MSE error, we can reduce the possibility of overshooting during the learning process and help the network get out of a local minimum. There are four parameters used to create the rules for the fuzzy logic control system; the relative error (RE), change in relative error (CRE), sign change in error (SC) and cumulative sum of sign change in error (CSC). The four parameters are described as follows:

$$\begin{cases} RE(t) = E(t) - E(t-1) \\ CRE = RE(t) - RE(t-1) \\ SC(t) = 1 - \left\| \frac{1}{2}\left[sign(RE(t-1) + sign(RE(t))\right\| \right. \\ CSC = \sum_{m=t-4}^{t} SC(m) \end{cases} \tag{6}$$

For simplicity, we assume that the fuzzy logic system contains two inputs RE and CRE, and two output; the change in the learning parameter $\Delta\eta$ and change in the momentum value $\Delta\alpha$. The range of RE and CRE is 0–1. The linguistic values of RE, CRE and $\Delta\eta$ are NL, NS, ZE, PS and PL. NL represents a "Negative Large" value, NS is "Negative Small", ZE is "Zero", PS is "Positive Small" and PL is "Positive Large". The change in the momentum value is small enough to avoid overcorrection. Tables 1 and 2 describe the matrix representations of the rule bases for $\Delta\eta$ and $\Delta\alpha$, respectively.

Table 1
Decision table for $\Delta\eta$ when CSC ≤ 2.

CRE	RE				
	NL	NS	ZE	PS	PL
NL	NS	NS	NS	NS	NS
NS	NS	ZE	PS	ZE	NS
ZE	ZE	PS	ZE	NS	ZE
PS	NS	ZE	PS	ZE	NS
PL	NS	NS	NS	NS	NS

3.4. Fuzzy deep neural network training

Deep multi-layer neural networks have many levels of non-linearity allowing them to compactly represent highly non-linear and highly varying functions. The training phase of deep neural network contains two major steps of parameter initialization and fine tuning. The initialization step is critical in deep learning because the whole learning system is not convex. A better initialization strategy may help the neural network to converge to a good local minimum more efficiently. In this paper, a SAE model is introduced. The fine tuning step allows to precisely adjusting the parameters in the neural network in a supervised way to enhance the discriminate ability of the final feature presentation. To conduct this step, we propose to use the back-propagation method with gradient based optimization technique and an on-line fuzzy logic controller in the role of supervised to adapt the learning parameters based on the mean squared error generated by the deep neural network. The training procedure can be summarized by Algorithm 1.

Algorithm 1 (*Fuzzy Deep Network Training Algorithm*).

```
Given the training sample and the structure of the deep
  neural network.
Step 1: Parameters Initialization
1. Train the first layer as an auto-encoder with the
  training sets as the input using the back-propagation
  algorithm.
2. Train the second layer as an auto-encoder taking the
  first layer's output as input.
3. Iterate as in 2 for the desired number of layers.
Step 2: Fine-Tuning
4. Use the output of the last layer as the input for the
  classification layer, and initialize its parameters by
  supervised training.
5. Fine-tune the parameters of all layers with the Back-
  Propagation method in a supervised way using fuzzy
  logic controller to adaptively adjust the learning
  parameters.
```

4. Network configuration and performance measures

Traffic incidents are the main cause of traffic congestion, which subsequently increases travel delay and fuel

Table 2
Decision table for $\Delta\alpha$ when CSC ≤ 2.

CRE	RE				
	NL	NS	ZE	PS	PL
NL	−0.01	−0.01	0	0	0
NS	−0.01	0	0	0	0
ZE	0	0.01	0.01	0.01	0
PS	0	0	0	0	−0.01
PL	0	0	0	−0.01	−0.01

consumption in major cities. TID became very important topic in traffic management systems. Detection of incident will avoid future traffic incidents and will help authorities to make road segment available for traffic again. This article focuses on the application of a fuzzy deep learning method for TID. To improve the efficiency of the proposed method in detecting traffic incidents, we used the microscopic traffic simulator SUMO (Simulator for Urban Mobility). Our traffic network is shown in Fig. 2. It contains twenty-five four way intersections equipped with traffic signals and more than 100 edges. Each edge contains three lanes in each direction. Each lane has two detectors, one at the entry to collect upstream traffic data and the other at the exit to collect downstream traffic data. During the simulation, more than 1000 vehicles are generated by uniform distribution over 20 input sections.

To train and validate our proposed model, sufficient data that includes different incident patterns under a variety of flow conditions has to be acquired. There are a few options in SUMO to simulate traffic incident such as stopping a car for a fixed period of time, by defining a point along its route when it should halt and for how long. We decided to employ this option since it is the easiest to implement. We introduced 30 incidents and we aggregated the data in 100-s intervals. Our dataset contained traffic information including the mean speed of vehicles traveling on a lane, the lane occupancy rate, the current traffic flow F^t and the flow rate at previous time intervals, i.e., $F^{t-1}, F^{t-2}, \ldots, F^{t-r}$. Therefore, the proposed model accounts for the temporal correlations of traffic flow. The model can be build from the perspective of a transportation network considering also the spatial correlations of traffic flow. Thus, we collected the data from all edges of our network. The dataset was captured by 360 pairs of inductive loop detectors. It contains 15,000 samples divided into three distinct sets called training, testing and validation sets. The training set with 60% sample is used to train the deep network model to get optimal parameters. The testing set with 20% samples is used to predict future performance of the network. A final check of the performance of the trained network is made using validation set with 20% samples. The sets are selected randomly from the full dataset. It is the simplest method for dividing the data which usually produces good results.

The performance of a deep neural network model is very sensitive to the proper selection of network architecture. To find the best deep neural network for a given dataset, we need to decide such specific details as how many neurons and layers we want to use and how many outputs the network should have. Different combination of number of layers and number of nodes in each layer makes different architectures of a deep network. The basic architecture selection process proposed in this paper is described in Algorithm 2. It is inspired from Wright and Manic (2010). We randomly generate a set of architectures. Each architecture is trained and the MSE is computed for the training set and the validation set. The goal is to find a neural network architecture with least mean of $MSE_{training}$ and $MSE_{validation}$. The MSE of the training set $MSE_{training}$, the MSE of the testing set $MSE_{testing}$ and the MSE of the validation set $MSE_{validation}$ are defined by the following equations (Azar & El-Said, 2013):

$$MSE_{training} = \frac{1}{N_{tr}} \sum_{i=1}^{N_{tr}} (d_i - o_i)^2 \tag{7}$$

$$MSE_{testing} = \frac{1}{N_{te}} \sum_{i=1}^{N_{te}} (d_i - o_i)^2 \tag{8}$$

$$MSE_{validation} = \frac{1}{N_{va}} \sum_{i=1}^{N_{va}} (d_i - o_i)^2 \tag{9}$$

where the MSE is the mean $\left(\frac{1}{N}\sum_{i=1}^{N}\right)$ of the square of the errors $(d_i - o_i)^2$. N_{tr} is the number of training data, N_{te} is the number of testing data and N_{va} is the number of validating data. d_i is the desired output and o_i is the actual output produced by the neural network.

Algorithm 2 (*Architecture Selection Process*).

```
1. Gather Data
2. Divide data into three sets
3. **For all** Candidate architectures **Do**
4.    Train network with training set
5.    $MSE_{training} \leftarrow$ MSE of training set
6.    $MSE_{validation} \leftarrow$ MSE of validation set
7. **End For**
8. Choose network with minimum mean of $MSE_{training}$
   and $MSE_{validation}$
```

There are numerous measures to evaluate the detection performance such as Detection Rate (DR), False Alarm Rate (FAR), Mean Time to Detection (MTTD) and Classification Rate (CR). DR is defined as the percentage of incident cases detected correctly by the system. FAR is one of the main parameters for evaluating incident detection systems. It is an index to represent the rate at which the non-incident cases are falsely classified as incident cases, thus raising a false alarm. MTTD is computed as the average length of time between the start of the incident and the time the alarm is initiated. The first correct alarm declared for a single incident is used for computing MTTD.

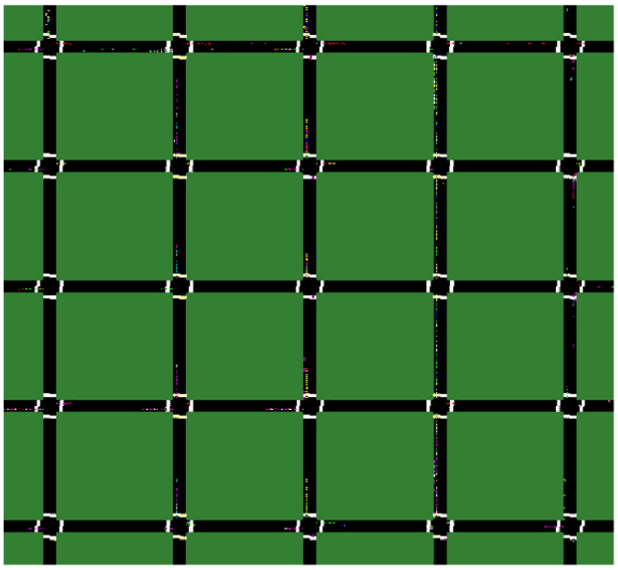
Fig. 2. Network configuration composed of twenty-five intersections.

CR is computed as the percentage of classified objects including both incident and non-incident objects, by the model of the total number of testing objects. The measures mentioned above are proposed as follows:

$$DR = \frac{number\ of\ incident\ detected}{total\ number\ of\ incident\ cases} * 100 \tag{10}$$

$$FAR = \frac{number\ of\ false\ alarm\ cases}{total\ number\ of\ non\text{-}incident\ instances} * 100 \tag{11}$$

$$MTTD = \frac{t_1 + t_2 + \ldots + t_m}{m} \tag{12}$$

$$CR = \frac{TP + TN}{P + N} * 100 \tag{13}$$

where t_i is the time delay between the occurrence of the incident and its detection. m is the number of the incidents detected. P is the number of positive objects indicating an incident state and N is the number of negative objects indicating an incident free state. TP and TN are numbers of true positive and true negative respectively.

5. Simulation and results

Our simulation is divided basically into three parts. We determine the optimal structure of the deep network in the first part. In the second part, we compare four models: process neural network with single hidden layer (SLNN), process neural network with multiple hidden layers (MLNN), deep process neural network (DNN) and fuzzy deep process neural network (FDNN). Notice that SLNN and MLNN are training in supervised way while DNN and FDNN are training in unsupervised way. The structure of MLNN, DNN and FDNN are exactly the same. The only difference is the training algorithm. MLNN is training with traditional learning method. In the last part, the comparison is made between the DNN and FDNN in term of speed of convergence.

One important issue in neural network is selecting appropriate structure. Different combination of number of layers and number of neurons in each layer makes different structures of a deep network. We generated 5 structures randomly. We limited the number of hidden layers between 3 and 7, and the number of neurons in each layer between 3 and 20. Simulation results are reported in Table 3. The structure with minimum mean of $MSE_{training}$ and $MSE_{validation}$, is determined as the optimal structure. The minimum mean of $MSE_{training}$ and $MSE_{validation}$ is $(0.16 + 0.14)/2 = 0.15$. Thus, the optimal structure is [11, 12, 10, 14], which means the deep network has four AEs stacked whose number of neurons are 11, 12, 10 and 14 respectively.

To evaluate the TID method, we used the performance measures. DR and FAR quantify the effectiveness of an algorithm while the MTTD and CR reflect the efficiency of the algorithm. The MSE assesses the quality of our estimator. Another term of comparison is training time (TT). It can calculate the time consumed by the method to reach an optimal classifier. Tables 4 and 5 summarize the testing performance of SLNN, MLNN, DNN and FDNN. It is evident that FDNN showing high detection rate, low false alarm rate, high classification rate, least $\text{MSE}_{\text{training}}$ and least $\text{MSE}_{\text{testing}}$, performs better than SLNN, MLNN and DNN. SLNN is the fastest since it is with the simplest structure of single hidden layer. Training time of DNN compared to MLNN does not increase a lot. One reason maybe that time complexity of training a process neural network and training a deep auto-encoder is similar. DNN uses fixed learning rate and fixed momentum term. 0.5 is set as the value of η and α. We can conclude that FDNN is far way better that DNN; the previous can converge in 275 s better than DNN which takes 383 s. FDNN showed a 28.19% average improvement in training time by employing a fuzzy logic controller to adaptively vary the learning parameters.

Finally, we make comparison on converting speed and effect. We have provided training details in Fig. 3. FDNN is faster on converging than DNN, MLNN and SLNN; FDNN could reach convergence in less than 7 iterations and DNN in 12 iterations, while MLNN requires about 15 iterations and SLNN needs about 20 iterations. From the results, we could find that DNN is effective in traffic incident detection. More layers in process neural network could capture better feature representation. Training algorithm based on fuzzy deep learning is also much more

Table 3
Comparaison between different structures.

# of AEs	Structure	$\text{MSE}_{\text{training}}$	$\text{MSE}_{\text{validation}}$
3	[8, 10, 15]	0.54	0.85
4	[11, 12, 10, 14]	**0.16**	**0.14**
5	[19, 5, 20, 9, 11]	0.28	0.23
6	[9, 8, 15, 13, 18, 12]	0.33	0.45
7	[16, 11, 17, 8, 5, 13, 9]	0.37	0.57

Table 4
Comparaison between SLNN, MLNN, DNN and FDNN in terms of indices of performance.

	Detection rate	False alarm rate	Mean time to detection	Classification rate
SLNN	80.24%	0.33%	161.55 s	77.54%
MLNN	87.58%	0.32%	**155.02 s**	83.02%
DNN	92.89%	0.23%	188.52 s	89.79%
FDNN	**98.23%**	**0.24%**	192.44 s	**91.47%**

Table 5
Comparaison between SLNN, MLNN, DNN and FDNN in terms of mean squared error and training time.

	Training time	$MSE_{training}$	$MSE_{testing}$
SLNN	**122 s**	0.34	0.35
MLNN	321 s	0.31	0.32
DNN	383 s	0.19	0.18
FDNN	275 s	**0.16**	**0.13**

effective than traditional deep learning. By employing a fuzzy logic system to adjust the learning parameters, the fuzzy deep learning method could avoid sticking in local minimum effectively, increase the convergence speed and minimize the error.

6. Conclusions

Traffic incident detection is critical for increasing road safety. For the complexity and the nonlinear characteristics of traffic incidents, this paper proposes a detection method based on a novel fuzzy deep learning approach. Parameters of the deep network are initialized using a Stacked Auto-Encoder (SAE) model, while the back-propagation algorithm is used to precisely adjust the parameters in the deep network. The BP algorithm suffers from some drawbacks. First, slow convergence rate and second the possibility of settling into a local minimum. To avoid these issues, fuzzy logic system is employed to control the learning parameters. The performance of the proposed method was evaluated in terms of detection rate, false alarm rate, mean time to detection, classification rate, training time and mean squared error. The obtained simulation results show that FDNN model performs better than others models when provided with the same data source. FDNN surpasses SLNN and MLNN in terms of indices of performance and DNN in term of speed of convergence. FDNN minimizes the $MSE_{training}$ and the training time by 15.78% and 28.19% comparing to DNN, respectively. The presented analysis results confirm the efficiency and accuracy of the proposed approach.

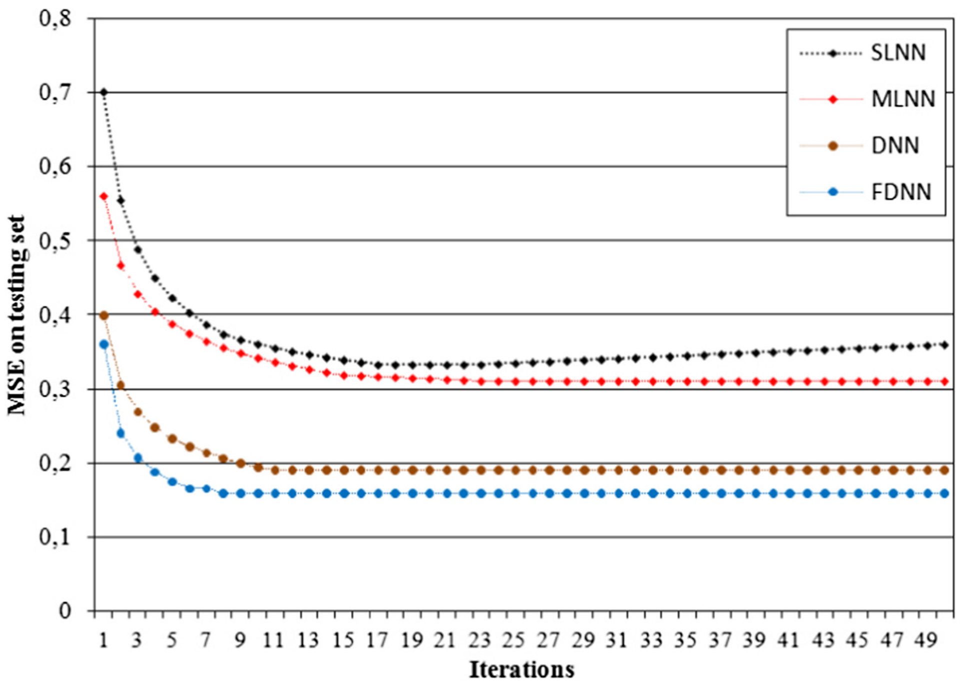

Fig. 3. Comparison on converting speed and effect.

References

Azar, A. T. (2012). Overview of type-2 fuzzy logic systems. *International Journal of Fuzzy System Applications, 2*, 1–28.

Azar, A. T. (2013). Fast neural network learning algorithms for medical applications. *Neural Computing and Applications, 23*, 1019–1034. https://doi.org/10.1007/s00521-012-1026-y.

Azar, A. T., & El-Said, S. A. (2013). Probabilistic neural network for breast cancer classification. *Neural Computing and Applications, 23*, 1737–1751. https://doi.org/10.1007/s00521-012-1134-8.

Chakraborty, M. (2012). Artificial neural network for performance modeling and optimization of CMOS analog circuits. *International Journal of Computer Applications, 58*(18), 6–12.

de Oliveira, M. B. W., & de Almeida Neto, A. (2014). Optimization of traffic lights timing based on artificial neural networks. In *2014 IEEE 17th international conference on Intelligent Transportations Systems (ITSC), October 8–11, 2014*. .

Deng, Y., Ren, Z., Kong, Y., Bao, F., & Dai, Q. (2016). A hierarchical fused fuzzy deep neural network for data classification. *IEEE Transactions on Neural Networks and Learning Systems, 6706*, 1–8.

Deng, L., & Yu, D. (2013). Deep learning: Methods and applications. *Foundations and Trends in Signal Processing, 7*, 197–387.

El Hatri, C., & Boumhidi, J. (2017). Traffic management model for vehicle re-routing and traffic light control based on multi-objective particle swarm optimization. *Journal of Intelligent Decision Technologies, 11*(2), 199–208. https://doi.org/10.3233/IDT-170288.

El Hatri, C., Tahifa, M., & Boumhidi, J. (2016). Extreme learning machine-based traffic incidents detection with domain adaptation transfer learning. *Journal of Intelligent Systems*. https://doi.org/10.1515/jisys-2016-0028.

Eze, U. F., Emmanuel, I., & Stephen, E. (2014). Fuzzy logic model for traffic congestion. *Journal of Mobile Computing and Application*, 15–20.

Huand, W., Hong, H., Song, G., & Xie, K. (2014). Deep process neural network for temporal deep learning. In *International joint conference on neural networks*. .

Kumar Basu, J., Bhattacharyya, D., & Kim, T. (2010). Use of artificial neural network in pattern recognition. *International Journal of Software Engineering and its Applications, 4*(2).

Lu, J., Chen, S., Wang, W., & van Zuylen, H. (2012). A hybrid model of partial least squares and neural network for traffic incident detection. *Expert Systems with Applications, 39*, 4775–4784.

Lv, Y., Duan, Y., Kang, W., Li, Z., & Wang, F. (2015). Traffic flow prediction with big data: A deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 865–873.

Rossi, R., Gastaldi, M., Gecchele, G., & Barbaro, V. (2015). Fuzzy logic-based incident detection system using loop detectors data. *Transportation Research Procedia, 10*, 266–275.

Rossi, R., Gastaldi, M., Gecchele, G., & Barbaro, V. (2015). Fuzzy logic-based incident detection system using loop detectors data. *Transportation Research Procedia, 10*, 266–275.

Shin, H.-C., Orton, M. R., Collins, D. J., Doran, S. J., & Leach, M. O. (2013). Stacked autoencoders for unsupervised feature learning and multiple organ detection in a pilot study using 4D patient data. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 35*(8), 1930–1943.

Wright, J. L., & Manic, M. (2010). Neural network architecture selection analysis with application to cryptography location. In *WCCI IEEE world congress on computational intelligence*. .

Yuan, F., & Cheu, R. L. (2003). Incident detection using support vector machines. *Transportation Research Part C: Emerging Technologies, 1*, 309–328.