2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Deep Process Neural Network for Temporal Deep Learning

Wenhao Huang, Haikun Hong, Guojie Song, Kunqing Xie

Abstract— **Process neural network is widely used in modeling temporal process inputs in neural networks. Traditional process neural network is usually limited in structure of single hidden layer due to the unfavorable training strategies of neural network with multiple hidden layers and complex temporal weights in process neural network. Deep learning has emerged as an effective pre-training method for neural network with multiple hidden layers. Though deep learning is usually limited in static inputs, it provided us a good solution for training neural network with multiple hidden layers. In this paper, we extended process neural network to deep process neural network. Two basic structures of deep process neural network are discussed. One is the accumulation first deep process neural network and the other is accumulation last deep process neural network. We could build any architecture of deep process neural network based on those two structures. Temporal process inputs are represented as sequences in this work for the purpose of unsupervised feature learning with less prior knowledge. Based on this, we proposed learning algorithms for two basic structures inspired by the numerical learning approach for process neural network and the auto-encoder in deep learning. Finally, extensive experiments demonstrated that deep process neural network is effective in tasks with temporal process inputs. Accuracy of deep process neural network is higher than traditional process neural network while time complexity is near in the task of traffic flow prediction in highway system.**

I. Introduction

Research of neural networks has been lasting for several decades. Many different structures of neural networks such as perceptron[17], feed-forward neural networks[1] and back-propagation neural networks[16] have been proposed and discussed in extensive researches. However, inputs for these neural networks are independent of time. Outputs are usually not only related with inputs in a static time but also related with accumulation of inputs for a period of time. For example, traffic flow (number of vehicles) on a road in a specific time is related with traffic flow on nearby roads in previous several time intervals. Inspired by biological observations that a neuron is activated by the accumulation of related input neurons for a time period, He et al. proposed process neural network (PNN) to deal with temporal process inputs in neural networks[5][6]. Inputs and weights between two layers are represented as functions of time or time-varying sequences in PNN. PNN has been applied in many areas such as simulation of oil reservoir exploitation[5], worm harm prediction[14] and churn prediction in mobile communications[14].

As same as many other neural networks, traditional process neural network is usually limited in the structure of single hidden layer. On the one hand, it is because of the unfavorable training strategies of neural network with multiple hidden layers as discussed in many researches[9]. Error back-propagation would stalk in the top layer in training so that weights of other layer are nearly the same as weights of random initialization[4]. Neural network with multiple hidden layers was examined not as effective as neural network with single hidden layer in many applications. On the other hand, since weight is represented as a time-varying function or a sequence, the structure of process neural network would be very complex when extended to multiple hidden layers. However, for complex systems with temporal process inputs, one single hidden layer usually would be not enough in describing complicated relations between inputs and outputs.

To overcome two difficulties mentioned above, in this paper, we proposed deep process neural network (DPNN) with multiple hidden layers. Two basic structures of deep process neural network: accumulation first DPNN and accumulation last DPNN are presented and discussed. With the two basic structures, we could build any architecture of deep process neural networks. For training strategy of DPNN, emergence of deep learning approaches provided a promising solution of training multi-layer neural networks, though it is limited in static inputs[8]. We proposed learning algorithms for two basic structures based on numerical learning for process neural network[20] and auto-encoder in deep learning[2]. In this paper, temporal process inputs are represented as time-varying sequences. In most of applications based on process neural network, inputs are also represented as sequences since it is very difficult to derive a time-varying function of inputs with limited prior knowledge. From limited temporal observations, we could only represent inputs as sequences. In addition, deep learning advocates unsupervised feature learning with less prior knowledge. We have to make inputs as sequences for this purpose. Finally, we conducted abundant experiments on real highway data. Transportation system is a typical system with temporal process inputs. Vehicles arrived in an exit station are accumulations of cars from other entrance stations in highway during several time periods. Experimental results show that DPNN is more effective than process neural network with single hidden layer (PNN-S) and traditional multi-layer process neural network (MPNN) in accuracy of traffic flow prediction. In addition, time complexity of DPNN is nearly the same as PNN-S and MPNN.

The rest of this paper was organized as follows. Section II introduces backgrounds of process neural network, deep learning and traffic flow prediction. Section III presents two structures and learning method of deep process neural

Wenhao Huang, Haikun Hong, Guojie Song, Kunqing Xie are with the Department of Electronic Engineering and Computer Science, Peking University, China.

Guojie Song is the corresponding author of this paper (corresponding email:gjsong@pku.edu.cn).

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

network. Experiments are reported in section IV. Conclusion and future prospects are given in Section V.

II. Backgrounds

Before introducing the deep process neural network, we first review backgrounds of process neural network and deep learning. The background of traffic flow prediction which is the application situation in this study is also introduced in this section.

A. Process Neural Network

Process neural network, also referred as procedure neural network, is proposed by He and Liang[5][6]. PNN is a generalized neural network in which the inputs are a serials of values varying with a procedure while the output is a static vector. A process neuron has the similar structure with a neuron of traditional artificial neural network. It has same operators such as weights multiplication, addition and inspiration. A single process neuron is shown in Figure 1 as an example.

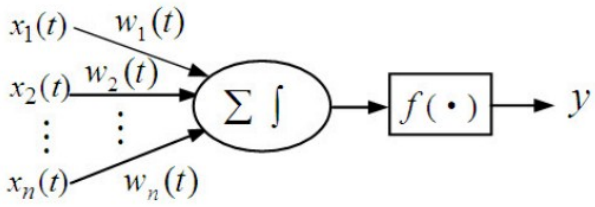

Fig. 1. Illustration of a process neuron.

The process neuron is different with a traditional neuron that inputs and weights in a process neuron could vary with time which enables process neural network dealing with process inputs. In addition, a process neuron has both time accumulation and space aggregation operators while traditional artificial neuron has only the space aggregation operator. Relation between inputs and outputs in process neural network could be formulated as follows:

$$y = f(\sum(\int(W(t), X(t)) + B)) \tag{1}$$

where $X(t)$ is the temporal process inputs and $W(t)$ is the temporal process weights; y is the output; B is the bias in neural network; $f(\cdot)$ is the excitation function such as sigmoid function; $\int$ is a time accumulation operator such as integral; $\sum$ is a space aggregation operator such as sum. Consider the most simple and familiar form where $\int$ is integral on time and $\sum$ is weight sum. It constructs the most classical form of process neural network. A hidden process neuron could be computed as follows:

$$h_j = f(\sum_{i=1}^{n}(\int_{t\in T} w_{ij}(t)x_i(t)dt + b)) \tag{2}$$

where $x_i(t)$ and $w_{ij}(t)$ are the temporal process vectors. They could be represented as time-varying function or temporal sequences.

Base function learning (BFL) is the most general learning approach for process neural network. Since $w_{ij}(t)$ could be any kind of time-varying functions, it is almost impossible to obtain their analytic forms. According to BFL, $w_{ij}(t)$ is assumed to be composed of a series of base functions, i.e. $w_{ij}(t) = \sum_{l=1}^{L} w_{ij}^{(l)} b_l(t)$ where $b_l(t)$ is a base function and L is the number of base functions. Then (2) could be rewritten as:

$$h_j = f(\sum_{i=1}^{n}\sum_{l=1}^{L} w_{ij}^{(l)} \int_{t\in T} b_l(t)x_i(t)dt + b) \tag{3}$$

With concrete form of base function, $\int_{t\in T} b_l(t)x_i(t)dt$ could be computed and the network could be trained by error back-propagation algorithm. However, it is difficult to choose appropriate base functions and their parameters before knowing concrete form of weight function in real application.

Numerical learning is proposed by Wu et.al based on the assumption that temporal inputs could be modeled as discrete numerical values[20]. In other words, temporal inputs are represented as sequences of observation values instead of time-varying functions. Weights are represented as temporal sequences too. It is much easier to collet temporal sequence of observations in real-world scenarios. Under this assumption, $\int_{t\in T} w_{ij}(t)x_i(t)dt$ could be computed by numerical integration in training, which can simplify the training process to a large degree. In this study, we follow the assumption that inputs are represented as temporal sequences used in the numerical learning approach.

B. Deep Learning

Recent works on deep learning have demonstrated that deep sigmoidal networks could be trained layer-wise to produce good results for many tasks such as image and audio classification[11][8][13]. Idea of deep learning is first using large amount of unlabeled data to learn feature by pre-training a multi-layer neural network in an unsupervised way and then using labeled data for supervised fine-tuning to adjust learned features slightly for better prediction.

Deep Belief Network is the most common and effective approach among all deep learning models. It is a stack of Restricted Boltzmann Machines each having only one hidden layer. The learned units activations of one RBM are used as the "data" for the next RBM in the stack. Hinton et al. proposed a way to perform fast greedy learning of the deep network in [8].

Another frequently used deep neural network is deep auto-encoder[2][19][12]. Stacked auto-encoders uses an auto-encoder instead of the RBM as a layer building block. An auto-encoder is composed of two parts: *encoder* and *decoder*.

Encoder: The encoder transforms an input vector $\boldsymbol{x}$ into hidden representation $\boldsymbol{x}$.

$$\boldsymbol{h} = f_\theta(\boldsymbol{x}) = sigm(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) \tag{4}$$

where $\theta = \{\boldsymbol{W}, \boldsymbol{b}\}$ is the parameter set of the encoder.

Decoder: The decoder maps the hidden representation $\boldsymbol{h}$ back to a reconstructed input vector $\boldsymbol{x'}$.

$$\boldsymbol{x'} = g_{\theta'}(\boldsymbol{h}) = sigm(\boldsymbol{W'}\boldsymbol{h} + \boldsymbol{b'}) \tag{5}$$

where $W' = W^T$ and b' is the bias for the decoder. The objective function of auto-encoder could be formulated as loss function between input vector $\boldsymbol{x}$ and reconstruction vector $\boldsymbol{z}$, i.e., $L(\boldsymbol{x}, \boldsymbol{x'}) = |\boldsymbol{x} - \boldsymbol{x'}|^2$. For greedy layer-wise training, hidden representation $\boldsymbol{h}$ could be computed by optimizing the objective function of one layer. Then $\boldsymbol{h}$ is used as input vector for the next layer of stacked deep auto-encoders.

In this study, we uses auto-encoder in each layer when constructing deep process neural network since auto-encoder is effective in modeling real-valued inputs.

C. Traffic Flow Prediction

Transportation system is a typical system with temporal process inputs[20]. Vehicles arriving at a specific location are from nearby locations. Therefore, traffic flow (number of vehicles) of a location is the accumulation of traffic flow on related locations in several previous time intervals. Transportation data is used in the experiment section to examine the effectiveness of proposed deep process neural network. The task in the experiment is traffic flow prediction. Here, we give a brief introduction of traffic flow prediction.

Traffic flow prediction, which aims at estimating number of vehicles in specific areas such as roads and stations in several future time intervals, is an important work in transportation management. Traffic flow prediction approaches differ largely from one another in many different aspects including data source (inductive loops, toll stations, surveillance cameras), prediction methods, scale of prediction (a fixed location or a whole network) and types of transportation system (urban network, freeway or highway)[10]. In very general terms, it can be formulated as:

$$y_j^t = G(x_i^{t'}, \theta), t - k \leq t' \leq t - 1, i \in O \tag{6}$$

where y_j^t output traffic flow of location j at time t, $x_i^{t'}$ is the traffic flow of all locations in the set of observation points O from time period $t - k$ to $t - 1$, k is the time step, G is a prediction model and θ is the parameter set of the model.

Toll data of highway system is used in our study. One record is produced when a car entering or leaving the highway from a toll station. The observation set O consists all the in and out stations in the highway system. Task we focusing on is predicting volume of vehicles in a station j at time period t.

A second preliminary is the measurement for traffic flow prediction. Two most commonly used measurements are mean absolute percentage error (MAPE) and root mean square error (RMSE)[10]. MAPE is used in our study as reported in section of experiments. It is computed as follows:

$$MAPE(y, y') = \frac{1}{n} \sum_{i=1}^{n} \frac{|y_i - y_i'|}{y_i} \tag{7}$$

III. Deep Process Neural Network

In this section, we first propose two basic types of deep process neural network. Then we introduce how to construct a general type of DPNN via the basic types of DPNN.

We would have two choices when extending process neural network to deep process neural network. The neurons in the extended deep layers could be traditional neurons or process neurons. Therefore, we could build two basic structures according to the position of process neurons. Accumulation first DPNN is in the structure that only the hidden neurons in the first layer are process neurons. Neurons in other layers are all static neurons as same as the neurons in traditional artificial neural networks. The first layer is referred as accumulation layer according to definition of process neurons. With that, it is easy to define the accumulation last DPNN. All the hidden neurons are process neurons in accumulation last DPNN. The general type of DPNN could be defined as a DPNN in which the hidden units in first m layers are process neurons and the hidden units in other n layers are static neurons.

A. Accumulation First DPNN

With the definition of accumulation first DPNN, it is easy to build the architecture of the network as shown in Figure 2. Oval nodes in the figure are process neurons as defined in traditional process neural network while circle nodes are static neurons as in traditional neural networks. From the perspective of feature learning, we can view the first layer (accumulation layer) as the temporal process feature representation which learns accumulation characteristics of temporal process inputs and other layers as feature transformation layers as in deep learning.

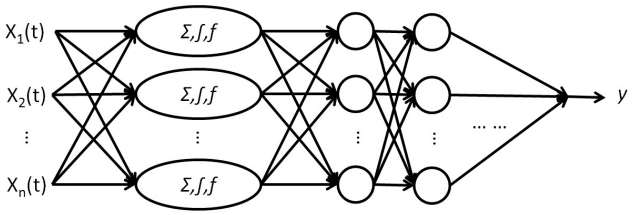

Fig. 2. The accumulation first deep process neural network.

Learning algorithm of the accumulation first DPNN is mainly focusing on the accumulation layer. The challenge is how to learn weights in the first layer in an unsupervised way without knowing hidden units in other layers. Inspired by the auto-encoder, we could build a process auto-encoder as shown in Figure 3.

From the process neural network , we can derive that hidden unit could be written as equation (2). Since we assume inputs and weights are represented as temporal sequences, for each temporal process input $x_i(t)$, the reconstruction of the input can be computed by:

$$x_i'(t) = g(\sum_{j=1}^{m} w_{ij}'(t) h_j + b_i') \tag{8}$$

For convenience, denote $u_j = \sum_{i=1}^{n} \int w_{ij}(t) x_i(t) dt + b$, $z(t) = \sum_{j=1}^{m} w_{ij}'(t) f(u_j) + b'$. Then the objective function

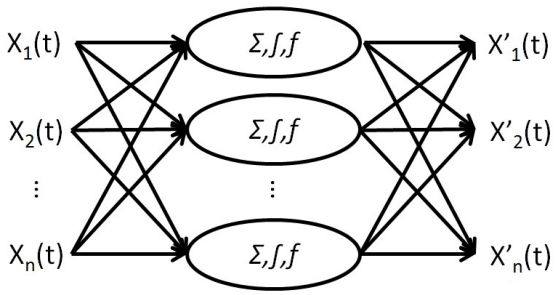

Fig. 3. Illustration of the process auto-encoder.

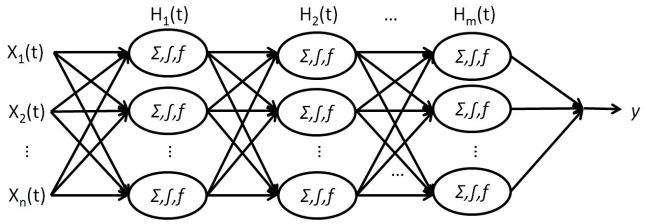

Fig. 4. Illustration of accumulation last deep process neural network.

could be written as:

$$\begin{aligned} & L(\boldsymbol{x(t)}, \boldsymbol{x'(t)}) \\ &= \sum_{i=1}^{n}(g(\sum_{j=1}^{m} w'_{ij}(t)h_j + b'_i) - x_i(t)))^2 \\ &= \sum_{i=1}^{n}(g(\sum_{j=1}^{m} w'_{ij}(t)f(u_j) + b'_i) - x_i(t)))^2 \\ &= \sum_{i=1}^{n}(g(z(t)) - x_i(t))^2 \end{aligned} \tag{9}$$

We could apply gradient descent in learning of parameter set $\{w_{ij}(t), b_j, b'_i\}$. Gradient of each parameter could be computed as follows:

$$\begin{aligned} & \Delta w_{ij}(t) = -\frac{\partial L}{\partial w_{ij}(t)} \\ & = -2\sum_{i=1}^{n}(g(z(t) - x_i(t))g'(z(t))(f(u_j) + w_{ij}(t)f'(u_j)\int x_i(t)dt)) \\ & \Delta b = -\frac{\partial L}{\partial b} = -2\sum_{i=1}^{n}(g(z(t) - x_i(t))f'(u_j)) \\ & \Delta b' = -\frac{\partial L}{\partial b'} = -2\sum_{i=1}^{n}(g(z(t) - x_i(t))) \end{aligned} \tag{10}$$

where $\int x_i(t)dt$ could be computed by numerical integration. If excitation function f and g are the most widely used sigmoid functions, $f(x) = (1 + e^{-x})^{-1}$ then $f'(x) = f(x)(1 - f(x))$. Weight $w_{ij}(t)$ and bias b, b' could be computed by above equations in unsupervised training.

Since other hidden nodes are all static neurons, learning algorithm is as same as learning algorithm of traditional deep auto-encoders as in [2].

B. Accumulation Last DPNN

Since one layer of process neurons may be insufficient to learn temporal process feature representation, we propose accumulation last DPNN here. Architecture of an accumulation last DPNN is shown in Figure 4. All the hidden layers are composed of process neurons. We use $H_i(t)$ to denote process hidden layer i.

Learning algorithm of accumulation last DPNN would be more complex than accumulation first DPNN since all the hidden layers are temporal process layers. However, under the assumption that temporal inputs are represented as sequences, we could simplify learning process to a large degree. The learning algorithm is proposed based on the assumption. Some structure rearrangements are required in the learning process. We give an illustration of learning structure for accumulation last DPNN in Figure 5.

One temporal process input $x_i(t)$ is a serial of inputs $\{x_i^{t-k+1}, x_i^{t-k+2}, \ldots, x_i^{t-1}, x_i^t\}$ where superscript denotes a specific time interval and k is the length of the sequence. The first step is rearranging these inputs according to time intervals instead of features. For a specific time interval t, we could get a sequence of inputs $X^t = \{x_1^t, x_2^t, \ldots, x_n^t\}$ where n is the number of features. The hidden nodes are also rearranged according to time. Then we could use the sequence of inputs in time interval t to produce the first hidden layer $H_1^t = \{h_{1,1}^t, h_{1,2}^t, \ldots, h_{1,n_1}^t\}$ in time interval t where n_1 is the number of hidden neurons in first hidden layer. After hidden nodes for all time intervals are established, we could further arrange hidden neurons according to nodes as $H_1(t) = \{h_{1,1}(t), h_{1,2}(t), \ldots, h_{1,n_1}(t)\}$. Or we can just leave it alone since we have to rearrange it according to time intervals when building the next hidden layer. Suppose there are $m + 1$ hidden layers in the whole structure. The first m layers could be built by the same way as the process of building first hidden layer $H_1(t)$.

We can use a traditional auto-encoder to describe relations between input X^t and hidden layer H_1^t at a specific time interval t. Similarly, it could be viewed as a traditional auto-encoder between two consecutive layers $H_i(t)$ and $H_{i+1}(t)$. For a specific time interval t_i, it formulates a deep auto-encoder. On the whole, there are k independent deep auto-encoders for each time interval t_k. Each auto-encoder could be learned as same as learning algorithm of traditional auto-encoder separately. To now, we have given unsupervised learning algorithm for the first m hidden layers.

It is different between last two hidden layers. Hidden layer m is produced as described above. Then we rearrange these hidden neurons in different time intervals according to features, i.e., $H_m(t) = \{h_{m,1}(t), h_{m,2}(t), \ldots, h_{m,n_m}(t)\}$. These nodes are arranged as process neurons which are represented as black nodes in Figure 5. Consider the structure of hidden layer m, hidden layer $m+1$ and output layer y. It is a classical process neural network with one hidden layer if we view hidden layer m as input layer. Learning algorithm

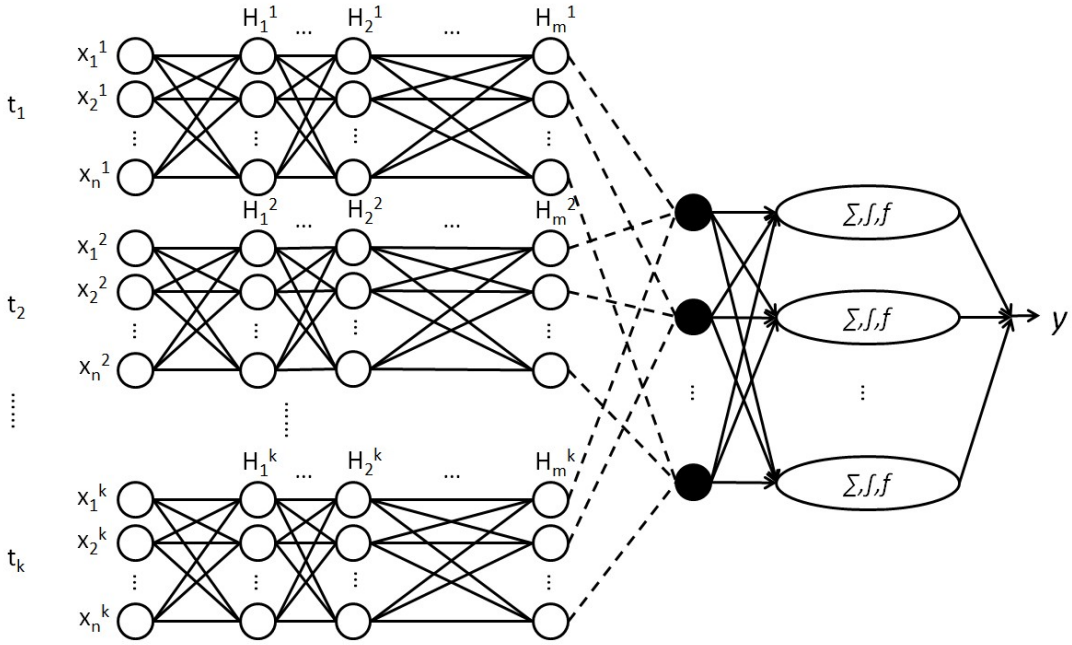

Fig. 5. Learning structure for accumulation last deep process neural network.

is the same as learning algorithm of the accumulation layer of accumulation first DPNN which has been introduced in previous subsection.

In conclusion, the whole learning procedure consists two parts. The first part is learning t separate deep auto-encoders for first m hidden layers. The second part is learning one hidden layer process neural network by process auto-encoder in an unsupervised fashion.

C. General Type of DPNN

After introducing two basic structures of the deep process neural network, we give an illustration of how to build a general type of DPNN based on that.

Suppose a general DPNN has m layers where the first m_1 layers are temporal process layers and the last m_2 layers are static layers where $m = m_1 + m_2$. Figure 6 is an example of a general type of DPNN. The first m_1 layers are actually an accumulation last DPNN. The m_1 layer and the last m_2 hidden layers form an accumulation first DPNN. Therefore, any type of deep process neural network could be built by two basic DPNNs.

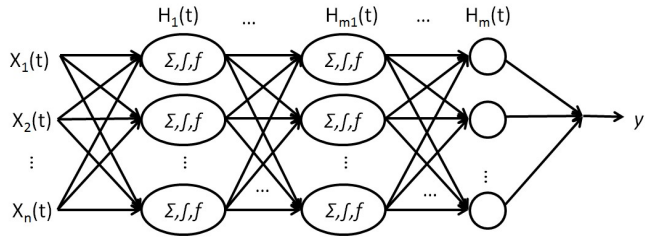

Fig. 6. Illustration of a general type of deep process neural network.

Since a general DPNN is composed of one accumulation last DPNN and one accumulation first DPNN. The learning algorithm of a general DPNN is learning an accumulation last DPNN first and then learning an accumulation first DPNN.

IV. Experiments

A. Experimental Settings

Dataset. One dataset we employed is from highway system of a province in China. There are total 284 entrance and exit stations in the province. In each entrance and exit of highway, there is a toll station for charging and recording related information. Data are collected in each station for 24 hours each day and aggregated into 15 minutes periods. Our task is predicting number of vehicles arriving at each exit toll station in next time interval (15 minutes) from traffic flow of entrance stations in previous several time intervals. It is a typical example of a complex system with temporal process inputs. Vehicles arriving at an exit toll station are all from other entrance stations in previous several time intervals. One time interval is setting at 15 minutes as suggested by Highway Capacity Manual.

Comparison Methods. In the experiments, we implemented three models for comparison: process neural network with single hidden layer (PNN-S), process neural network with multiple hidden layers but training with traditional learning methods (MPNN), deep neural network with full inputs (DNN). DNN expands MPNN by k times where k is the number of time intervals in a temporal process input, i.e. length of the sequence. All the nodes in hidden layers are static nodes and they are fully connected. For deep process neural network, we designed three structures: DPNN-1, DPNN-2 and DPNN-3. They are all with three hidden

layers. Only the first hidden layer is temporal process layer in DPNN-1 which is an example of accumulation first DPNN. First two hidden layers are temporal process layers in DPNN-2 and the last layer is the same as hidden layer in traditional artificial neural network. DPNN-2 is a general type of DPNN. All the three hidden layers are temporal process layers in DPNN-3 which is an example of accumulation last DPNN. MPNN is also setting at three hidden layers. Actually, the structures of MPNN and DPNN-3 are exactly the same. The only difference is the training algorithm.

Inputs and Outputs. Input (feature vector) in the model is the traffic flow of entrance stations in previous k time intervals. Traffic flow of one station i is viewed as one input x_i. It is represented as a sequence of k time intervals $x_i(k) = \{x_i(t_1), x_i(t_2), \ldots, x_i(t_k)\}$. Length of time intervals k is fixed at 8 from experimental results as reported in later section. Outputs are traffic flow of all exit toll stations in next time interval, i.e. $Y = \{y_1, y_2, \ldots, y_n\}$ where n is the number stations in the highway system (142 in our experiments). Reported results are the average of these stations.

Training and Testing Data. Highway dataset contains data of totally 12 months in 2011 while we use data of first 10 months as training set and later 2 months as testing set.

Though deep learning and process neural network cost a lot in storage and computation, most of the experiments could be finished in less than 1 hour in a single machine with Core i3 CPU, 4G memory and 512MB GPU memory.

B. Architecture of neural networks

One important issue in neural network is selecting appropriate number of hidden nodes in each hidden layer. In the experiments, only PNN-S is designed with only one single hidden layer. We tested number of hidden nodes from $\{100, 200, 300, 400, 500\}$. Accuracy on training set and training time are reported in Table 1.

TABLE I
EFFECT OF NUMBER OF NODES IN HIDDEN LAYER.

Number of nodes	MAPE	Training time
100	0.172	33s
200	0.161	65s
300	0.155	102s
400	0.154	135s
500	0.153	175s

From the results, we could see that training time increases linearly with the increase of hidden nodes. However, the MAPE dose not improve a lot when size of hidden nodes is over 300. The PNN-S is setting at 300 hidden neurons in later experiments.

For MPNN and DPNN, we fixed the number of neurons in each hidden layer to be equal for the convenience of computing. The number of hidden neurons is selected from $\{50, 100, 150, 200, 300\}$. The results of MAPE and training time are reported in Table 2.

From MAPE on training set, we could see that more nodes in hidden layers does not provide improvements on performance when number of nodes exceeds 100. However, training time is continuously increasing with the growth of size of the network. Hence, MPNN and DPNN are designed with 100 hidden neurons in each hidden layers. Notice that PNN-S and MPNN are training in supervised way while DPNN is training in unsupervised way. Training time of DPNN 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. We would make detail comparisons on training time in later section.

TABLE II
EFFECT OF NUMBER OF NODES IN ONE HIDDEN LAYER FOR NEURAL NETWORKS WITH MULTIPLE HIDDEN LAYERS.

	MPNN		DPNN-3	
Number of nodes	MAPE	Training time	MAPE	Training time
50	0.163	153s	0.149	199s
100	0.152	311s	0.143	383s
150	0.152	464s	0.143	567s
200	0.151	620s	0.143	748s
250	0.151	770s	0.143	936s
300	0.151	923s	0.143	1112s

C. Experimental Results

In this section, we report experimental results and comparisons of several process neural network related models.

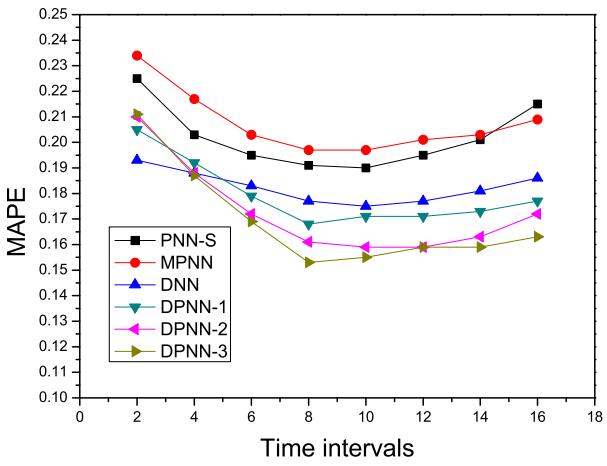

Fig. 7. MAPE on testing set of all comparison methods under different length of input sequence.

Prediction accuracy is the most important evaluation metric for traffic flow prediction. MAPE is used for measurement of accuracy. It is related with length of temporal process inputs, i.e. number of time intervals k. We give results of MAPE on testing set of all comparison methods under different length of input sequence in Figure 6. From the result, all the methods obtain best performance when k is at 8 and 10. Average traveling time of cars on the highway is about two hours (8 time intervals) from the data. Length of temporal accumulation should be similar with average traveling time to ensure best performance of temporal process neural networks. DPNNs perform better than other methods

on MAPE when length of time intervals (k) is over 4. DPNN could improve accuracy over 3% than traditional PNN. MPNN is worse than PNN-S due to the unfavorable training algorithm as introduced in previous sections. This also explains why process neural network with only single hidden layer is preferred in previous studies. Another interesting finding is that testing error (Figure 6) is bigger than training error (Table 1 and Table 2, $k = 8$ in those experiments). All the models are over fitted or under fitted on training set to a certain extent. Difference of performance on training set and testing set is smaller in DPNN (about 1%) than PNN-S (about 4%) and MPNN (about 5%). It demonstrates that DPNN is more robust than other models. DPNN-3 is generally better than DPNN-1 and DPNN-2 when k is over 8. With the increasing of time intervals, temporal feature representation would be more important. Therefore, more temporal process layers would lead better performance. This also supports the fact that performance of DPNN is over PNN.

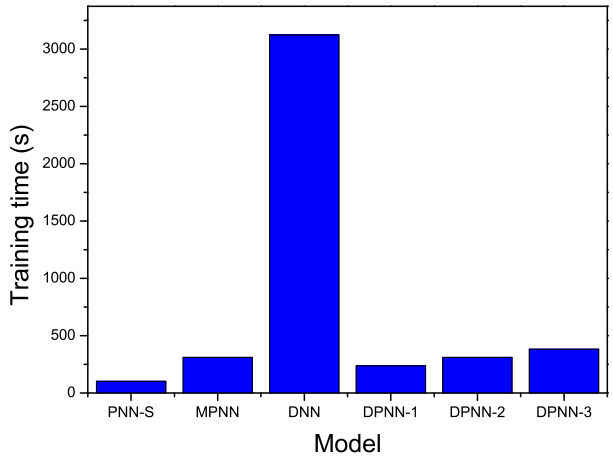

Fig. 8. Comparison of time complexity.

Another factor we concern about is time complexity of model training. Though the neural network models would not be updated very frequently, training time should still be acceptable. Training time of different models is reported in Figure 8. PNN is the fastest since it is with the simplest structure of single hidden layer. For deep networks, training time of DPNN and MPNN is similar. It implies that training temporal process neural network and deep auto-encoders are with similar time complexity. DNN is expanded from DPNN. It connects several separate anto-encoders together which leads to high computational complexity. Actually, from Figure 5, we could see that spatial and time complexity of DPNN is $O(nmk)$ and complexity of DNN is $O(nmk^2)$ where n is number of inputs, m is number hidden neurons and k is length of time intervals. Though DNN is with a more complex structure, co of DPNN is better than DNN. This implies that many unrelated connections in DNN would cause negative effect on final results.

Finally, we make comparison on converging speed and effect. Gradient descent would be performed several iterations during training of each model. We recorded the model after each iteration and tested it on the testing set. The result is demonstrated in Figure 9. DPNN is faster on converging speed than other methods. It could reach convergence in less than 10 iterations while PNN needs about 30 iterations and MPNN requires about 15 iterations. It is interesting that more iterations after convergence would make performance of PNN-S and MPNN on testing set worse. This is a typical phenomenon of local minima in training. The result shows that deep learning method could avoid sticking in local minima effectively. Or deep learning approaches could find a better local minima than traditional error back-propagation as presented in [3].

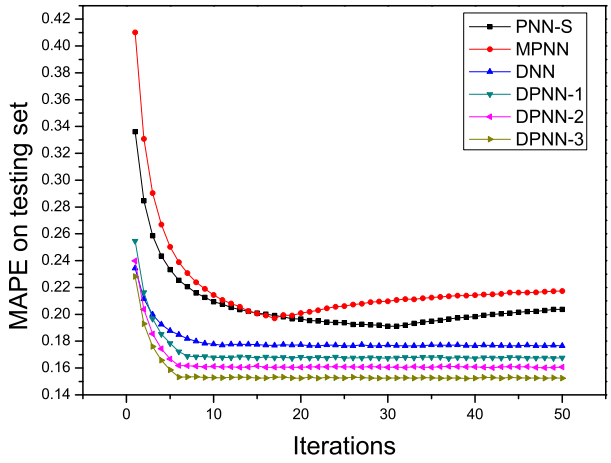

Fig. 9. Comparison on converging speed and effect.

From these experimental results, we could find that DPNN is effective in traffic flow prediction. More layers in process neural network could capture better temporal feature representation. Training algorithm based on deep learning is also much more effective than traditional error back-propagation.

V. Conclusion

In this paper, we extended traditional process neural network into deep process neural network. We proposed two basic structures to build a deep architecture of process neural network with multiple hidden temporal process layers. Accumulation first DPNN advocates learning temporal feature representation in first process layer and then performing static feature transformation in other hidden layers. Accumulation last DPNN prefers more hidden process layers for temporal feature representation. Any general type of DPNN could be built with these two basic structures. Then we proposed learning algorithm for accumulation first DPNN and accumulation last DPNN. The learning algorithm is inspired by deep auto-encoders for effective learning of deep static neural networks and numerical learning for process neural network. Finally, we conducted extensive experiments on the scenario of traffic flow prediction, which is a typical system with temporal process inputs. The results demonstrated that DPNN is better in comparison of prediction accuracy. Learning algorithm we proposed could also reduce

computational complexity of DPNN to make it close to other comparison methods.

There are still many potential works to do along this direction. One is using time-varying functions instead of sequences of temporal process inputs. The key may be how to bring the idea of unsupervised deep learning into base function learning for process neural network. Another work we have to do is examining DPNN on more scenarios such as climate prediction and worm harm prediction. Finally, it is interesting to investigate temporal accumulation effect between different features. The process itself may contain useful information. Inputs in different time intervals are actually related with each other. Finding these connection explicitly would be a great help for building the architecture of deep process neural network.

REFERENCES

[1] George Bebis and Michael Georgiopoulos. Feed-forward neural networks. *Potentials, IEEE*, 13(4):27–31, 1994.

[2] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007.

[3] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, 11:625–660, 2010.

[4] Dumitru Erhan, Pierre-Antoine Manzagol, Yoshua Bengio, Samy Bengio, and Pascal Vincent. The difficulty of training deep architectures and the effect of unsupervised pre-training. In *International Conference on Artificial Intelligence and Statistics*, pages 153–160, 2009.

[5] Xin-Gui He and Shao-Hua Xu. Process neural network with time-varied input and output functions and its applications. *Journal of software*, 14(4):764–769, 2003.

[6] Xingui He and Shaohua Xu. Process neural networks. *Process Neural Networks: Theory and Applications, Advanced Topics in Science and Technology in China, ISBN 978-3-540-73761-2. Springer-Verlag Berlin Heidelberg, 2010*, 1, 2010.

[7] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[8] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[9] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[10] SHAN HUANG and ADEL W SADEK. Artificial intelligence and microscopic traffic simulation models. *Artificial Intelligence Applications to Critical Transportation Issues*, page 65, 2012.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1106–1114, 2012.

[12] Sascha Lange and Martin Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–8. IEEE, 2010.

[13] Hugo Larochelle, Yoshua Bengio, Jérôme Louradour, and Pascal Lamblin. Exploring strategies for training deep neural networks. *The Journal of Machine Learning Research*, 10:1–40, 2009.

[14] Jiuzhen Liang and Xiaohong Wu. Worm harm prediction based on segment procedure neural networks. In *Rough Sets and Knowledge Technology*, pages 383–388. Springer, 2006.

[15] Abdel-rahman Mohamed, George Dahl, and Geoffrey Hinton. Deep belief networks for phone recognition. In *NIPS Workshop on Deep Learning for Speech Recognition and Related Applications*, 2009.

[16] Fernando J Pineda. Generalization of back-propagation to recurrent neural networks. *Physical review letters*, 59(19):2229–2232, 1987.

[17] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[18] Yee Whye Teh and Geoffrey E Hinton. Rate-coded restricted boltzmann machines for face recognition. *Advances in neural information processing systems*, pages 908–914, 2001.

[19] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 9999:3371–3408, 2010.

[20] Tianshu Wu, Kunqing Xie, Guojie Song, and Xingui He. Numerical learning method for process neural network. In *Advances in Neural Networks–ISNN 2009*, pages 670–678. Springer, 2009.