Supervised Deep Learning Based for Traffic Flow Prediction

Hendrik Tampubolon
Department of Information Systems
Krida Wacana Christian University
Jakarta, Indonesia
hendrik.tampubolon@ukrida.ac.id

Pao-Ann Hsiung
Dept. of Computer Science and Information Eng.
National Chung Cheng University
Chiayi, Taiwan
pahsiung@csie.io

Abstract— **In metropolitan areas, common traffic issues include traffic congestion, traffic accidents, air pollution, and energy consumption occur. To resolve this issues, Intelligent Transportation Systems (ITS) have been evolved by many researchers. One of the important sub-systems in the development of ITS is a Traffic Management System (TMS) which attempts to reduce a traffic congestion. In fact, TMS itself relies on the estimation of traffic flow, therefore providing such an accurate traffic flow prediction is needed. For this reason, we aim to provide an accurate traffic flow prediction to facilitate this system. In this works, a Supervised Deep Learning Based Traffic Flow Prediction (SDLTFP) was proposed which is a type of fully-connected deep neural network (FC-DNN). Timely prediction is also a major issue in guaranteeing reliable traffic flow prediction. However, training a deep network could be time-consuming, and overfitting is might be happening, especially when feeding small data into the deep architecture. The network is learned perfectly during the training, but in testing with the new data, it could fail to generalize the model. We adopt the Batch Normalization (BN) and Dropout techniques to help the network training. SGD and momentum are carried out to update the weight. We then take advantage of open data as historical traffic data which are then used to predict future traffic flow with the proposed method and model above. Experiments show that the Mean Absolute Percentage Error (MAPE) for our traffic flow prediction is within 5 % using sample data and between 15% to 20% using out of the sample data. Training a deep network faster with BN and Dropout reduces the overfitting.**

Keywords— Deep Learning, Traffic Flow Prediction, Dropout, Batch Normalization, Intelligent Transportation Systems

I. INTRODUCTION

In the metropolitan areas, common traffic issues include traffic congestion, traffic accident, energy consumption occurs. One of the reason is due to the use of road demand is increasing as the number of vehicles is also continued increasing [1]. One of the solutions is the development of Intelligent Transportation Systems (ITS) application which estimates the traffic stream. As such, traffic flow prediction is a strong need. Therefore, providing an accurate and timely predictive model is a serious concern. On the other hand, open data allow us to exploit the traffic information without having our own vehicle detector or sensor in the road such as Data Taipei [2].

Many researchers have been developing the model for traffic flow prediction in the past years. For example, a time series based approaches such ARIMA was applied [3][4][5], SVR based approaches such in the work of [6][7][8] [9] that perform well for the short-term predictive model, Backpropagation Neural Network (BPNN) [6] and also Neural Network(NN) approaches were also implemented [10]. However, the traditional NN is still shallow. As traffic flow is affected by many complex factors such as event, weather, accident, and etc. Thus, the deep network architecture may be needed. Deep Learning (DL) is now possible after Hinton proposed a breakthrough fast learning for the deep network [11][12]. There are some DL approaches have been used for traffic flow prediction, for instance, Stacked Autoencoders (SAEs) was proposed to learn the features of the traffic flow that consider the correlation between spatial and temporal data as in [13]. A greedy layerwise technique that learns one layer at a time is performed for learning representation of features. In [14] DBN and Stacked Restricted Boltzmann Machines (RBMs) were used at the bottom layer that learns features with limited prior knowledge. The multitask regression layer at the top, subsequently, a grouping method that applies a simple k-means based on weight sharing in order to make multitask learning (MTL) more effective. J. Lemieux and Y. Ma [15] also study DL approach to predict specific driver speed profile. Stacked Autoencoders (SAEs) were implemented. The features of freeway road are learned and then used as the input to Neural Network(NN), which can learn a particular driver's behavior. the work can predict the velocity accurately at each point of the drive route. Huang et. Al [16] proposed Deep Process Neural Network (DPNN) for temporal DL for traffic flow prediction in the highway system. All of above works mentioned do pre-training unsupervised then followed by supervised learning to tune the model. The pre-training step is less important nowadays, for instance, Nicholas G. Polson [17] proposed Feed Forward Neural Network(FFNN) with advance Dropout regularization technique, Vector Auto-Regressive(VAR) for predictor selection and Rectifier linear unit (Relu) activation function to capture nonlinear spatio-temporal effect in both recurrent and non-recurrent traffic congestion. In a deep network, there are several layers, the input layer, followed by several hidden layers and then the output layer. The input layer when provided to the first hidden layer, would be processed upon and a new output would be generated by the hidden layer. Then the output of the hidden layer becomes the input to the next hidden layer and goes on and on. When this process continues, the learning rates and other parameters such as weight initialization etc, would have to be reviewed over and over, in order to prevent the training process from becoming too slow due to the requirement of low learning rates, Batch normalization(BN) approach is introduced[18]. BN will help to accelerate the deep network learning. In addition, overfitting may occur when the dataset is small and the network is deep. The Dropout can solve this issue as in [19][20][21] explained. Moreover, the traditional activation such sigmoid may suffer saturation. Thus, Relu was used which can tackle this problem[22].

In this work, we focus on supervised DL which is a type of Fully-Connected Deep Neural Network (FC-DNN). We add the Dropout layer and BN layer into our network. We train our predictive model over Stochastic Gradient Descent (SGD) and Back Propagation learning using traffic data and weather data as our variables predictors. Instead of saturated non-linearity, Relu was applied to the network.

II. Related Background

A. Neural Network and Deep Learning

In the past years, training a deep network was very hard and time consuming until [23]. Hinton invented a fast learning for deep belief networks with a greedy layerwise training that lead to revival of deep network, now referred as Deep Learning (DL). As a time goes by, many DL variants have been proposed, for instance Deep Boltzmann Machine (DBM), Denoising Autoencoder (DAE), Deep Neural Network (DNN).

Consider a supervised learning problem where we have access to labeled training examples (s^t, y^i). Neural networks give a way of defining a complex, non-linear form of hypotheses $h_{W,b(s)}$, with parameters *W, b* that we can fit to our data. The neuron is a computational unit that takes as input s^t, s^{t-1}, s^{t-m} (and b +1 intercept term), and outputs $h_{W,b(s)} = f(W^T s) = f(\sum_{t=1}^{3} W_t\, s_t + b)$ where $f\colon \Re \to \Re$ is called the activation function.

The activation function can be defined as follows:

- *Sigm:* Sigmoid function, formulated by $f(a) = \frac{1}{1+e^{-a}}$
- *Tanh*: Tangent hyperbolic, formulated by $f(a) = \frac{e^{a} - e^{-a}}{e^{a} + e^{-a}}$
- *Relu*: Rectifier linear unit, formulated by *f(a) = max(0, a)*.

  Max: Maxout, formulated by *f(a1, a2) = max(a1, a2)*

B. Traffic Flow Prediction

The problem of traffic flow prediction can be stated as follows. Let F_t denote the observed traffic flow quantity during the *t-th* time interval at the particular observation road segment in a transportation network. Given a sequence $\{F_t\}$ of observed traffic flow data, *t = 1, 2, ..., T,* the problem is to predict the traffic flow at time interval (*t + k*) for *k-th* prediction horizon, k is positive integer, *1, 2, .., K* meaning that *k = 1* is one ahead prediction namely short term prediction and *k-th* ahead prediction is long term prediction.

C. Stochastic Gradient Descent (SGD) and Momentum

We use well-known learning algorithm backpropagation to calculate the gradient and update the weight parameters using SGD. Suppose given training set pairs $(F_{in(t)}, F_{out(t)})$ for *t = 1, ..., T*, our goal is to find parameter Θ that minimize the distance between $F_{in(t)}$and $F_{out(t)}$ with respect to Ł(Θ). The loss function Ł(Θ) can be denoted as follows:

$$\mathcal{L}(\theta) = MSE = \tfrac{1}{2}\textstyle\sum_{t=1}^{T}(F_{out(t)} - \hat{F}_{out(t)})^2 \qquad (1)$$

where $F_{out(t)}$is observed traffic flow or the target and $\hat{F}_{out(t)}$ is predicted traffic flow and *Θ = {W, B}*

$$\theta_t = \theta_t - \alpha \frac{\partial}{\partial \theta_t} \mathcal{L}(\theta; F_{out(t)}, \hat{F}_{out(t)}) \qquad (2)$$

where α is the learning rate and $\frac{\partial}{\partial \theta_t}\mathcal{L}(\theta)$ is the derivative term with a pair $(F_{in(t)}, F_{out(t)})$ from the training set. If α is too small, gradient descent will be slow to reach the region of the minimum, if α is too large, gradient descent will go past the region of minimum unintentionally. An inappropriate α may lead to failure in converge. Therefore, we add velocity ϑ component to parameter SGD update as follows:

$$\vartheta_t = \eta\vartheta_{t-1} + \alpha \frac{\partial}{\partial \theta_t} \mathcal{L}(\theta_{t-1}) \qquad (3)$$

$$\theta_t = \theta_{t-1} - \vartheta_t) \qquad (4)$$

Where $\eta \in (0,1)$ is momentum rate, *t* is the number of iteration, however, η is set to be 0.5 until the initial learning stabilizes and then is increased to 0.9 as recommended in general practice in the most recent work.

In backward propagate, we first need to calculate the error denoted as $\delta F_{out(t)}$ in **Eq. 5** which is basically the partial derivative. Then the chain rule of Backpropagation(BP) can be written in **Eq. 6.**

$$\frac{\delta F_{out(t)}}{\delta \hat{F}_{out(t)}} = \hat{F}_{out(t)} - F_{out(t)} \qquad (5)$$

$$\frac{\Delta\delta F_{out(t)}}{\Delta F_{in}} = \frac{\Delta\delta F_{out(t)}}{\Delta F_{hid}} \frac{\Delta\delta F_{hid}}{\Delta F_{in}} \qquad (6)$$

D. Dropout Regularization

Consider a network with L_{hid} as the number of hidden layers, where $l \in \{1, \dots, L_{hid}\}$ is the index of the hidden layers of the network. $F_{in}^{(l)}$ denotes the traffic data input vector at layer l where $F_{in}^{(l)} \in \{F_{train}, F_{valid}\}$ and $F_{out}^{(l)}$ denotes the traffic data output vectors from layer l, $F_{out}^{0} = F_{in}$, w^l and b^l are the weights and biases at layer l, respectively. Thus, a feed forward network operation for $l \in \{1, \dots, L_{hid-1}\}$ at any hidden units t can expressed as follows:

$$F_{in(t)}^{(l)} = w_t^{(l+1)} F_{out}^{l} + b_t^{(l+1)} \qquad (7)$$

$$F_{out(t)}^{(l+1)} = Relu(F_{in(t)}^{(l+1)}) \qquad (8)$$

Algorithm 1: Stochastic Gradient Descent (SGD)

```
Input:
$F_{train}$: Data of traffic flow used for training ;
$F_{valid}$: Data of traffic flow used for validation;
$F_{in} \in \{F_{train}, F_{valid}\}$ : Input data set
Output:
$\theta$: updated weights;
Variable:
$W$: Weights ;
b: Biases;
α: The learning rate;
Ł($\theta$): The loss function;
1 compute cost function Ł with Eq.1;
2 initialize parameter $\theta$ and Ł;
3 randomly shuffle data set $F_{in}$;
4 repeat
// For each example in $F_{in}$
5 for t = 1, .., T do
6 update $\theta_t$ by Eq. 2 and Eq.4 (for t =0,..,T)
```

Algorithm 2: Batch Normalization (BN)

```
Input:
$B = \{F_{in(1)}, \dots, s\}$: value of $F_{in}$ over s mini-batch B ;
$\beta$: BN shift parameter to be learned ;
γ: BN scale parameter to be learned ;
$F_{train}$: Data of traffic flow used for training ;
$F_{valid}$: Data of traffic flow used for validation;
$F_{in} \in \{F_{train}, F_{valid}\}$ : Input data set
Output:
$F_{out(t)}$ = { BNγ,β($F_{in}$ )}: Batch normalizing transform over a mini-batch;
Variable:
s: size of mini-batch B;
1 begin
// Batch normalization step
2 Calculate mini-batch mean $\mu B$ using Eq.13 ;
3 Calculate mini-batch variance $\sigma 2B$ using Eq. 14 ;
4 Normalizing using Eq.15;
5 Scale and shift using Eq.16;
6 Return $F_{out}$;
```

These feed forward operation can be rewritten after applying the Dropout in the following equation

$$dropout_{mask}{}^{(l)}_{(t)} = dropout_{rate}{}^{(l)}_{(t)} \sim bernoulli(p) \quad (9)$$

$$\tilde{F}^{(l)}_{out(t)} = dropout_{rate}{}^{(l)}_{(t)} * F^{(l)}_{(out)} \quad (10)$$

$$F^{(l+1)}_{out(t)} = w^{(l+1)}_t \tilde{F}^{(l)}_{out(t)} + b^{(l+1)}_t \quad (11)$$

$$F^{(l+1)}_{out(t)} = Relu\left\{F^{(l+1)}_{in(t)}\right\} \quad (12)$$

E. Batch Normalization

Suppose that the layer we want to normalize has T dimensions of the input set $F_{in} = (F_{in(1)}, F_{in(2)}, \dots, F_{in(T)})$. Let the mean over a mini-batch size B be given by Equation 4.18 and the variance $\sigma 2$ is calculated by Equation 4.19 where as s is the number of examples over mini-batch size B.

$$\mu B = \frac{1}{S}\sum_{t=1}^{S} F_{in(s)} \quad (13)$$

$$\sigma^2 B = \frac{1}{S}\sum_{t=1}^{S}\left(F_{in(s)} - \mu B\right)^2 \quad (14)$$

$$\hat{F}_{in(t)} = \frac{F_{in(t)} - \mu B}{\sqrt{\sigma^2 B + \varepsilon}} \quad (15)$$

$$F_{out(t)} = \gamma \hat{F}_{in(t)} + \beta \equiv BN_{\gamma\beta}(F_{in(t)}) \quad (16)$$

III. Supervised Deep Learning Based Traffic Flow Prediction (SDLTFP)

As we aim to provide an accurate traffic flow predictive model with supervised deep learning based without pre-training process, thus, we then proposed SDLTFP framework design as you can see in Fig.1. There are several steps we would like to attain. Firstly, we adopt Dropout technique to mask the input training set both in the input layer and hidden layers. Secondly, we perform batch normalization which standardizes the input distribution before passing to non-linearity. Thirdly, A stochastic gradient descent (SGD) is used to update the weights during the training with back propagation learning. and using less saturated activation function. With this technique, we aim to train our network faster and provide an accurate predictive model.

A. SDLTFP Network Architecture

Usually, there are two types of DNN, Convolution Neural Network(CNN) and Fully Connected (FC) network. SDLTFP Network is type of fully connected deep neural network (FC-DNN). The difference between our network and traditional ANN with a standard feed forward and Back-Propagation Neural Network (BPNN) is that it has Dropout to mask neurons layer and BN layer to normalize the distribution of activation layer.

B. Training the SDLTFP Network

The SDLTFP Network training is divided into two process, forward propagation and backward propagation. At first, feed forward the training input to the network after initialize the network configuration and its weight, then make the batch data by given s. The Dropout layer will mask some of neurons which is determined by given Dropout rate, then calculate the input the remain neurons with the weights and biases, before passing to activation unit, normalizing the result calculation first by mini batch mean μB and variance $\sigma 2B$ followed by scale and shift. Repeating this process until the desire layers. In the backward step, calculate first the error with the loss function, then updating the weight and the gamma factor by SGD. BN layer is also need to back propagate as it has parameter γ and β to be learned together with δF_{out}. Fig. 3. depict the procces of training and also can be seen in the Algorithm 3.

C. Traffic Flow Predictive Model

The traffic flow data is divided into *t(day)* intervals as our goal to predict oneday ahead. We use the historical data of the traffic flow from November to December 2015. If we want to predict on 9 November use *t-3*, it means three days historical used for training, 2 November to 4 November. The input data has *NxT* dimension with mini-batch B {1, 2, ..s}. In the case of using weather data, it has different dimension which is hourly weather data, we repeat hourly data 12 times to have the same dimension. k-fold cross validation (k − CV) is used and the size of mini-batch B is set based on F_{valid}. After training process, the model of SDLTFP Network is stored in SDLTFP object which has a learned weight W, γ, and β of BN. We can predict $\hat{F}_{out}$ by given new inputs data F_{test} and its target as shown in Algorithm 4.

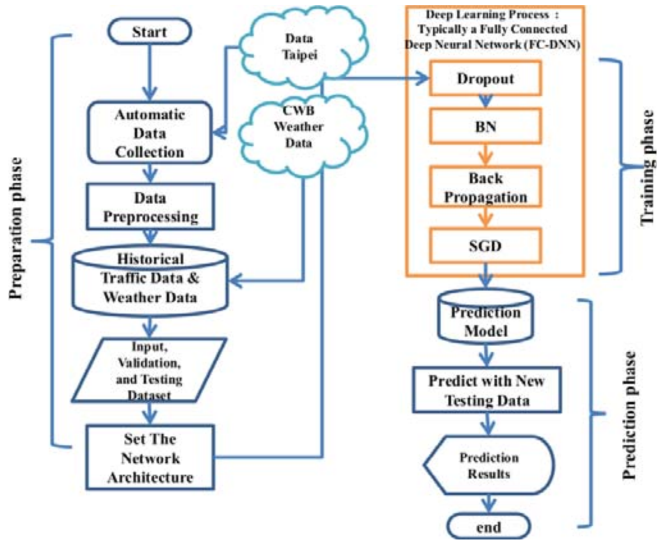
Preparation phase
Start
Data Taipei
CWB Weather Data
Automatic Data Collection
Data Preprocessing
Historical Traffic Data & Weather Data
Input, Validation, and Testing Dataset
Set The Network Architecture
Deep Learning Process : Typically a Fully Connected Deep Neural Network (FC-DNN)
Dropout
BN
Back Propagation
SGD
Training phase
Prediction Model
Predict with New Testing Data
Prediction Results
end
Prediction phase

Fig. 1. SDLTFP Basic Framework.

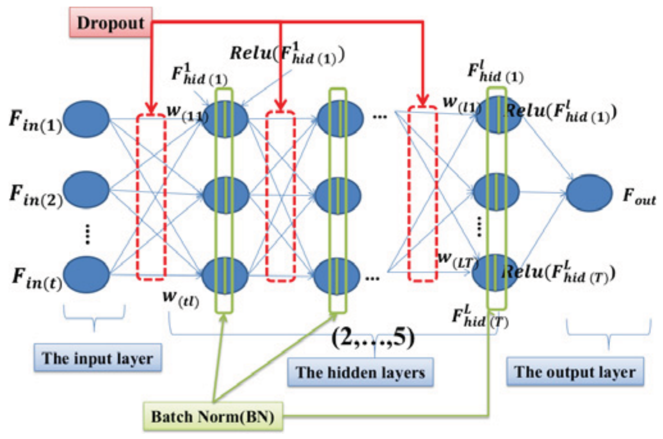

Fig. 2. SDLTFP Network Achitecture

Algorithm 3: Training SDLTFP Network

```
Input:
$SDLTFP \in \{N_{in}, N_{hid}, N_{out}\}$: neural network config;
$F_{in} \in \{F_{train}, F_{valid}\}$: The Input data training;
$F_{out} \in \{FtrainTarget, FvalidTarget\}$: The target traffic flow;
$\alpha$: Learning rate in SGD where $\alpha \in \{\alpha_{weight}, \alpha_{bias}\}$;
$pdropout_{Fin}$: Dropout rate in the input layer;
$pdropout_{Fhid}$: Dropout rate in the hidden layer;
$s$: Size of mini-batch $B$ ;
γ: The learning rate for the gamma factor in BN layer;
Nepoch: The number of iteration during the training time ;

Output:
SDLTFP: A trained neural network $SDLTFP \in \{W, \beta, \gamma\}$;

Variable:
flagBN: To denote the BN layer is used;
lossfunction: MSE ;
L: The number of layer ;
begin
      // Initialize Network Config
1:    SDLTFP ← newSDLTFP;
     // Initial variable
2:    Sepoch ← 0 ; i ← 1 ; s ← s;
3:    L ← length(SDLTFP.layershape); l ← 1;
      Lhid ← L − 1 ;
4:    for Sepoch ← 1 to Nepoch do
5:       Forward propagate ; Make batch ;
6:       for i ← 1 to s do
             // Dropout steps
7:           Generate Dropout mask ;
8:           Mask neuron with given pDropoutFin; //Eq.9,Eq.10
9:           for l ← 2 to Lhid do
10:              Mask neuron with pDropoutFhid; //Eq.9,Eq.10
11:              apply Dropout ; s + 1; //Eq.11, Eq.12
12:               if flagBN then
                    // Algorithm 2 BN;
13:               Calculate error signal δFout // Eq. 5;
14:               Calculate error BN δβ, δγ of BN ;
15:               Update the weights // Algorithm 1
16:               Back propagate;
17:               Nepoch + 1 ;
18:   save updated W, γ, β with minimum MSE ;
19:   return SDLTFP;
```

Algorithm 4: Traffic Flow Prediction

```
Input:
$F_{in} \in \{F_{test}\}$: Prediction inputs data frame;
W: Final weights of DNN;
γ: Final γ factor of BN ;
β: Final β BN ;
SDLTFP ∈ {W, γ, β}: Neural network config ;
Output:
$\hat{F}_{out}$: Prediction outputs;
Variable:
$Ftest_{Target}$: Target prediction ;
begin
// After training process
$\hat{F}_{out}$ ← SDLTFP (Ftest, FtestTarget)
return $\hat{F}_{out}$;
```

IV. Experiments

A. Datasets

The traffic data and the weather data used in our experiment are come from Open Data Taipei City[2] and CWB[24] respectively. We focus on the ZKAHN20 road segment (Zhongxiao Rd) which a heavy traffic flow as shown in Fig. 4. The heavy traffic flow is selected for the reason that we want the high density as the deep network is able to feed the high dimensional.

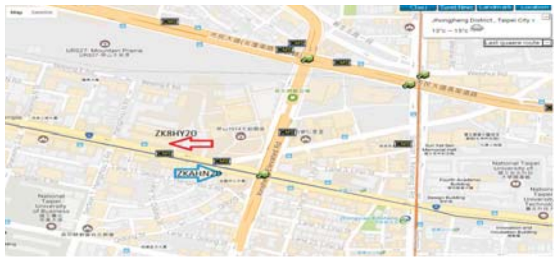

Fig. 4. Section 1, Zhongxiao E Rd, used in our experiment

TABLE I. VARIABLE PREDICTORS

Variables Predictor	Description	Unit
AvgVol	Average traffic volume (aggregate every 5 minutes)	Vehicles
AvgSpeed	Average traffic speed (aggregate every 5 minutes)	Km/h
Occ	Occupancy	%
MOELevel	Congestion degree	Number (0,1,2)
Temp	Weather temperature	℃
DewPoint	Dew point	℃
WS	Wind speed	m/s
WSGust	Wind speed gust	m/s
Precp	Precipitation	Mm
PrecpHour	Precipitation hour	Hour
Visb	Visibility	Km

B. Experimental Setting

There are three main scenarios in our experiment. Firstly, we do not use Dropout in the input layer since we only use the historical traffic flow of Wednesday to predict next Wednesday. Secondly, we use Monday to Wednesday traffic data as a variables predictors. The $dropout_{rate}$ was set up bigger in the input layer. Thirdly, we then add the weather data of the target day for the reason to investigate the effect of BN and Dropout when the network was trained with a deeper network and more data. The parameter setting can be seen in TABLE II in which we set the value of each parameter by several experiments.

TABLE II. PARAMETER SETTINGS

Parameters	Values		
	Experiment I	*Experiment II*	*Experiment III*
N_{in}	2 - 7	12	21
N_{hid}	(50,50)	(50,50,25)	(100,100,100)
N_{out}	1	1	1
α	$1e-8$	$1e-6$	$1e-6$
γ	$1e-8$	$1e-6$	$1e-6$
s	28	28	28
$pdropout_{Fin}$	0.0	0.2	0.1
$pdropout_{Fhid}$	0.5	0.5	0.5
mom_{init}	0.6	0.6	0.6
mom_{final}	0.9	0.9	0.9
mom_{switch}	100	100	100
N_{epoch}	100-1000	100-1000	100-1000

C. Performance Evaluation

To evaluate the accuracy of our prediction model, we use Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE) as follows:

$$\text{RMSE} = \sqrt{\frac{1}{M}\sum_{m=1}^{M}\left(F_{out(m)} - \hat{F}_{out(m)}\right)^2} \tag{17}$$

$$\text{MAE} = \frac{1}{M}\sum_{m=1}^{M}\left|F_{out(m)} - \hat{F}_{out(m)}\right| \tag{18}$$

$$\text{MAPE} = \frac{100}{M}\sum_{m=1}^{M}\left|\frac{F_{out(m)} - \hat{F}_{out(m)}}{F_{out(m)}}\right| \tag{19}$$

Where $F_{out(m)}$: Actual traffic flow value of sample *m*, $\hat{F}_{out(m)}$: Predictive traffic flow value of sample *m*, *M* : Number of training samples.

D. Experimental Results

In the experiment I. aims to test the influence of the number of inputs and the usage of Dropout in the input layer. The historical traffic flow of Wednesday is used to predict the next Wednesday by given the parameter settings as it shown in TABLE II. when N_{in} that used is equal to four, the SDLTFP network is convergence at around 40 of the epoch and predict well using the seen dataset, however, at the testing with unseen data (out of the samples), the result is bad, meaning that it fails to generalize. See the Fig. 5. In fact, the training data set is just a small data. The deep architecture will train perfectly by memorizing the data already seen, but it falls down for generalization. It seems likely an over-fitting. Therefore, adding Dropout regularization into our networks to reduce this occasion.

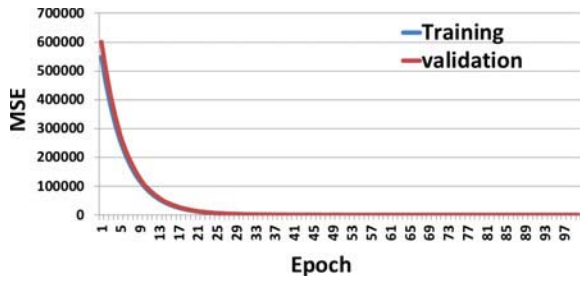

Fig. 5. Training error in Experiment I

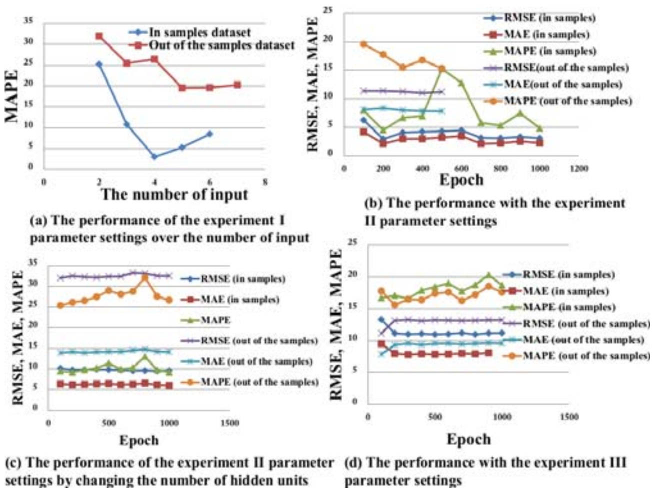

Fig. 6. The performance of the SDLTFP Network with various parameter settings and the number of input data, either using insamples dataset or out of the samples dataset

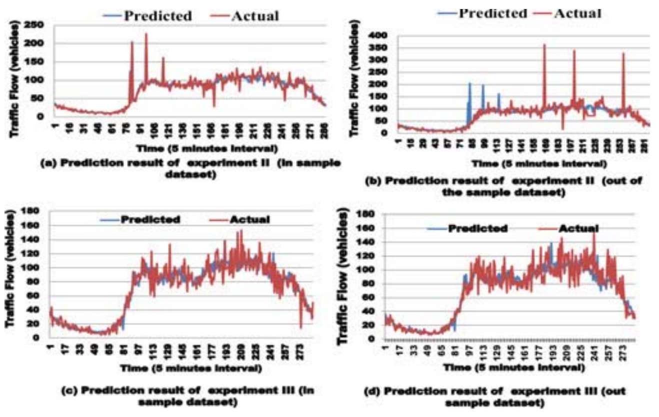

Fig. 7. Prediction results

Fig.6. (a) shows that using four or five previous days of traffic data bringing a lower MAPE. In the experiment II. aim to investigate changing the the structure of the *Nhid* to (100, 100, 100) as the same as the *Nhid* in the experiment III. The results can be seen in Fig.6. (b) and Fig. 6.(c). The MAPE is getting bigger, it implies that selecting architecture of the network is also important. We then use this architecture to the experiment III. and add the weather data of the target day into the our training set. While the experiment III. examines the effect of BN and Dropout when we train a deep network and more data. The results can be seen in Fig. 6. (d). As we can see the epoch of 200 gain better for generalization which is 15.610% MAPE. Comparing to the experiment II. when changing the number of hidden units *Nhid* to (100, 100, 100) the generalization is better, but the testing to the in samples data is not better, meaning that bigger data influence to the generalization and help to adapt to the new situation. In order to gain the advantages of Dropout and BN, we need more data set or large dataset, however our data still a small data. We also test without BN and with BN with respect to the training. For simplicity, we only test for 100 of epochs. Training with BN and training without BN are resulted 29.05 seconds and 83.09 seconds respectively.

V. Conclusions

In this work, we proposed a Supervised Deep Learning-based for Traffic Flow Prediction (SDLTFP) to provide an accurate traffic flow predictive model. We applied the Dropout and BN with Relu activation make network relax to the data without so much pre-processing. Experimental results show that training with BN and Dropout will give better result to the generalization and reduce over-fitting. In addition, with BN and Dropout is also giving the training of FC-DNN faster. In experiment I, it is easy to fit the training perfectly by memorizing the training data, however, it will be less accurate in generalization which out of the samples. The experiment shows us the prediction is around 15 % - 19% out of the samples and 5% to 10 % using in samples data.

VI. Future Work

In the future, we can add *early stopping* technique to our network for the training as we can see in Fig. 5., the training error does not improve that much after iteration 50-*th*. We can also consider Spatio-temporal by adding all traffic from each road segment, then examine the prediction with network perspective. Instead of BN, it is worth to extend the normalization into layer normalization incorporating with the LSTM recurrent neural network to exploit Spatio-temporal correlation. Furthermore, training a network parallely over GPU or cluster machine to speed up the training time. We believe with that approaches, we can gain better prediction result.

References

[1] M. Directorate General of Highways, “Taiwan registered vehicle amount,” 05-Dec-2014. [Online]. Available: https://www.thb.gov.tw/. [Accessed: 19-Apr-2018].

[2] Taipei City Government, “data.taipei.” [Online]. Available: http://data.taipei/. [Accessed: 09-Apr-2018].

[3] C. Chen, J. Hu, Q. Meng, and Y. Zhang, “Short-time traffic flow prediction with ARIMA-GARCH model,” *IEEE Intell. Veh. Symp. Proc.*, vol. 100084, no. Iv, pp. 607–612, 2011.

[4] M. Lippi, M. Bertini, and P. Frasconi, “Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning,” *Intell. Transp. Syst. IEEE Trans.*, vol. 14, no. 2, pp. 871–882, 2013.

[5] S. V. Kumar and L. Vanajakshi, “Short-term traffic flow prediction using seasonal ARIMA model with limited input data,” *Eur. Transp. Res. Rev.*, vol. 7, no. 3, pp. 1–9, 2015.

[6] D. Zeng, J. Xu, J. Gu, L. Liu, and G. Xu, “Short Term Traffic Flow Prediction Based on Online Learning SVR,” *8th Work. Power Electron. Intell. Transp. Syst.*, pp. 616–620, 2016.

[7] J. Y. Ahn, E. Ko, and E. Kim, “Predicting Spatiotemporal Traffic Flow Based on Support Vector Regression and Bayesian Classifier,” *2015 IEEE Fifth Int. Conf. Big Data Cloud Comput.*, pp. 125–130, 2015.

[8] Y. Yao, “Traffic flow forecasting with Particle Swarm Optimization and Support Vector Regression,” *2014 IEEE 17th Int. Conf. Intell. Transp. Syst.*, pp. 2267–2268, 2014.

[9] “Vessel Traffic flow forecasting with the combined model based on Support vector machine,” pp. 695–698, 2015.

[10] C. Bao-ping and M. Zeng-qiang, “Short-Term Traffic Flow Prediction Based on ANFIS,” *2009 Int. Conf. Commun. Softw. Networks*, pp. 791–793, 2009.

[11] G. E. Hinton, S. Osindero, and Y. W. Teh, “A fast learning algorithm for deep belief nets.,” *Neural Comput.*, vol. 18, no. 7, pp. 1527–54, 2006.

[12] G. E. Hinton and R. R. Salakhutdinov, “Science,” vol. 313, no. July, pp. 504–507, 2006.

[13] L. Yisheng, Y. Duan, and W. Kang, “Traffic Flow Prediction With Big Data : A Deep Learning Approach,” *Intell. Transp. Syst. IEEE Trans.*, vol. 16, no. 2, pp. 865–873, 2015.

[14] W. Huang, G. Song, H. Hong, and K. Xie, “Deep Architecture for Traffic Flow Prediction: Deep Belief Networks With Multitask Learning,” *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 5, pp. 2191–2201, 2014.

[15] J. Lemieux and Y. Ma, “Vehicle Speed Prediction Using Deep Learning,” *2015 IEEE Veh. Power Propuls. Conf.*, pp. 1–5, 2015.

[16] W. Huang, H. Hong, G. Song, and K. Xie, “Deep process neural network for temporal deep learning,” *Proc. Int. Jt. Conf. Neural Networks*, pp. 465–472, 2014.

[17] N. Polson and V. Sokolov, “Deep Learning for Short-Term Traffic Flow Prediction,” pp. 1–19, 2016.

[18] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” 2015.

[19] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.

[20] D. R. Tobergte and S. Curtis, “Improving Neural Networks with Dropout,” *J. Chem. Inf. Model.*, vol. 53, no. 9, pp. 1689–1699, 2013.

[21] P. Baldi and P. J. Sadowski, “Understanding Dropout,” *Nips*, no. 1, pp. 2814–2822, 2013.

[22] R. Arora, A. Basu, P. Mianjy, and A. Mukherjee, “Understanding Deep Neural Networks with Rectified Linear Units,” pp. 1–17, 2016.

[23] G. Hinton, Y. Bengio, and Y. Lecun, “Breakthrough Deep Learning : machine learning algorithms based on,” 2015.

[24] Central Weather Bureau of Taiwan, “CWB Observation Data Inquire System.” [Online]. Available: https://e-service.cwb.gov.tw/HistoryDataQuery/index.jsp. [Accessed: 20-Apr-2018].