**Deep-learning architectures to forecast bus ridership at the stop and stop-to-stop levels for dense and crowded bus networks**

**Junghan Baek**

Graduate student

Department of Urban Engineering, Chung‑Ang University, Seoul, Korea

221, Heukseok-dong, Dongjak-gu, Seoul 156-756, Korea

sweetieone2@naver.com

**Keemin Sohn\*** Corresponding author

Associate Professor

Department of Urban Engineering, Chung‑Ang University, Seoul, Korea

221, Heukseok-dong, Dongjak-gu, Seoul 156-756, Korea

[kmsohn@cau.ac.kr](mailto:kmsohn@cau.ac.kr)

**Abstract**

The conventional transit assignment models that depend on either probabilistic or deterministic theory have failed to accurately estimate rider demand for dense and crowded bus transit networks. It is well known that the existing models are so blunt that they cannot accommodate the impact on bus demand of miscellaneous changes in activity and transportation systems. Recently, artificial neural networks (ANNs) have been refocused after two monumental breakthroughs: Big-data and a novel pre-training method. A deep-learning model, which simply represents an ANN with multiple hidden layers, has had great success in recognizing images, human voices, and handwritten texts. The present study adopted a deep-learning model to forecast bus ridership at the stop and stop-to-stop levels. While the stop-level model, which had insufficient training data, suffered from an over-fitting of the data, the stop-to-stop-level model showed good performance both in training and testing. The success of the latter model is owed to a larger sample size compared with the former model. This represents the first meaningful attempt to apply a data-driven approach to forecasting transportation demand.

**Keywords**: Deep-learning; Big-data; Transit assignment; Bus ridership; Restricted Boltzmann machine (RBM); Deep neural network (DNN)

**1. Introduction**

The traditional four-step model for forecasting travel demand reveals limitations when applied to solving a real transport problem. In particular, it cannot estimate bus demand for either stop or stop-to-stop levels on a dense and crowded transit network. There are 611 bus lines and 14,287 bus stops within the 25 boroughs of the city of Seoul. Given the dense conditions, the existing models are incapable of accommodating the impact of a miscellaneous change in the activity and supply systems. In other words, if a single bus line or, more specifically, a single bus stop in the area is adjusted for some reason, it is nearly impossible to exactly measure how this change will affect the nearby bus-demand pattern. Nonetheless, available transit networks that a transportation expert can utilize have become more precise, as commercial transit navigation services have prospered. In fact, the existing transit assignment models that have been developed thus far in the transportation research field are too blunt to work in parallel with the networks that the commercial agencies provide. Although some transport experts may think that TRNASIMS-like simulation tools could accommodate them, it is apparent that the existing tools, which have been developed for cities in North America or in Europe, cannot properly allocate bus passengers for transit networks in the Seoul metropolitan area, wherein there are many road segments along which more than 150 different bus lines pass in parallel. This discrepancy discourages many transportation researchers in the area.

Traditionally, transportation experts have an idea that a transit network represents the transportation system, and, thus, it should be the base from which all theory-based trip assignment models are implemented. The present study applied a data-driven approach to forecasting bus demand, with a transit network regarded as inputs along with other features representing the activity system. The main intention of the present study was to verify whether data itself could explain everything that generates bus ridership without using any theory when much data are available.

The conventional transit assignment model depends on either a probabilistic choice theory (Lam and Xie 2007; Hoogendoorn-Lanser et al. 2005) or a deterministic algorithm for an optimal strategy (No¨kel and Wekeck 2009; Friedrich et al. 2007). Models based on a probabilistic choice cannot avert the following two problems: enumerating alternative paths is not trivial, and overlapping between alternative paths hinders a rational allocation of passengers. The latter problem could be solved by adopting a Probit model with fully correlated path utilities (Yai et al. 1997), but that measure entails a computational burden. A model based on the optimal strategy algorithm also does not properly split passengers across competitive transit lines, particularly when the transit lines are densely distributed and some of them are severely crowded. The algorithm can accommodate neither a capacity restraint on a transit line nor crowding in a bus.

The present study presents a deep-learning approach as a novel alternative to forecasting bus demand based only on feature data. Of course, the existing models also largely depend upon data, too. Thus, a theory-based model must be calibrated on observed data. However, the model performance is not likely to be proportional to the amount of data mobilized. No matter how much extra data are used for calibration, the fitting performance of the model cannot be ameliorated, since the effect of the additional data is trapped within a predefined model specification that is usually simple and shallow. On the other hand, a data-driven approach is sufficiently flexible to accommodate almost all the effects that data can provide. The goal of the present study was to investigate this advantage when a data-driven approach is applied to forecasting bus demand.

From the viewpoint of a transportation researcher, a deep-learning model can be seen as a direct travel-demand model. Regression analysis has been the most sought-after method to forecast railway ridership at the station and station-to-station levels (Ramos-Santiago and Brown, 2015; Zhao et al., 2015; Zhao et al., 2013; Choi et al., 2012; Sohn and Shim, 2010; Yao, 2007; Cervero, 2006; Boyle, 2006; Kuby et al., 2004; Parsons Brinckerhoff, 1996). Various direct-demand models also have been employed to forecast bus demand (Dill et al., 2013; Pulugurtha and Agurla 2012; Estupinan and Rodriguez, 2008; Chu, 2004; Dargay and Hanly, 2002). These methods, however, have focused mainly on the statistical association between the target demand and explanatory variables than on the accuracy of prediction. Furthermore, these methods cannot be applied to a case where strong collinearity between explanatory variables resides. The main difference between the present data-driven approach and the previous direct-demand models is that the present approach can account for every linear and nonlinear correlation between variables. Details of this advantage will be described in the following section for a modeling framework.

A deep neural network (DNN) is simply an artificial neural network (ANN) with multiple hidden layers. Stacking multiple hidden layers made it possible to more closely mimic the function of the human brain. Various DNN models have reaped great success in recognizing images, human voices, and handwritten text. ANNs with a single hidden layer were developed in the 40s and continued to prosper into the 80s and 90s. During the same period, there were many ANN applications to transportation studies (Park et al., 1998; Dougherty and Cobbett, 1997; Ledoux, 1997; Hashemi et al. 1995; Smith and Demetsky, 1994; Ritchie and Cheu, 1993). Unfortunately, ANNs with a shallow structure were nearly abandoned after the new millennium, since they were very susceptible to the over-fitting of training data. The term “deep-learning” was born against the failure of the previous shallow-learning structure with a single hidden layer. A DNN was created simply by employing multiple hidden layers for an ANN.

Recently, two breakthroughs have led to the success of a DNN: employing a new pre-training method and utilizing Big-data for training. While the former plays a crucial role in averting the fatal over-fitting problem (Hinton et al., 2006), the latter is more amenable to enhancing the performance of a DNN along with improvements in processing speed (Dean, 2012). Deng and Yu (2014) stated that some carefully designed random initial parameters can resolve the over-fitting problem without pre-training if a large amount of training data are available. The present study also investigated whether the pre-training process could enhance the training performance of a DNN.

A deep-learning model, the performance of which has already been verified in the field of artificial intelligence, has yet to be fully validated for solving transportation problems. Instead, adopting neural networks with a single hidden layer has been a main stream in the transportation research field (Moniruzzaman et al., 2016; Ma et al., 2015; Zhu et al. 2015). Recently, only a few researchers have tried to apply deep-learning technology to forecasting traffic parameters such as traffic volume, speed, and density (Lemieux and Ma, 2015; Lv et al., 2015; Huang et al., 2013). They commonly argued that a deep-learning model outperforms the existing models that are based on shallow-learning. As far as we could ascertain, the present study is the first application of a deep-learning technology to forecasting travel demand. While adopting a deep-learning model, many potential features that affect bus demand were chosen in the present study, and how to rearrange them into an input feature vector was then devised to feed them into the DNN model.

The present study is organized as follows. The next section will introduce the basic structure of a DNN, followed by describing how to train a DNN in the third section. Details of acquiring data to train and test a model will be described in the fourth section. Training and testing results will be shown and discussed in the fifth section. The last section will draw conclusions and suggest further possible enhancements.

**2. Modeling framework**

**2.1 The structure of a deep neural network (DNN)**

A DNN is composed of three types of layers: a single input layer, multiple hidden layers, and a single output layer. Generally, most researchers have adopted an ANN with a single hidden layer, since multiple hidden layers cause gradient-vanishing problems and entail computational burdens with regard to training. The former problem often encounters derivatives of cost function that diminish to zero, and, thus, updating weight parameters fails when implementing a back-propagation algorithm. This handicap has made researchers part with adopting ANNs with multiple hidden layers. The availability of Big-data has shed light on resolving this problem along with a new pre-training method that provides a good initial solution for weight parameters to be calibrated. Nonetheless, as mentioned earlier, the availability of Big-data is a more powerful breakthrough that has led to more success for deep-learning approaches than adopting a pre-training method. Regarding the computation burden, parallel computing skill is a key solution while implementing a back-propagation algorithm. The use of a graphics-processing unit (GPU) makes it possible to carry out operations with large-dimensional matrices within a practical amount of computation time.



Fig. 1. The structure of a DNN

Fig. 1 shows the structure of a DNN. The number of nodes in the input layer is identical to the dimensions of the input feature space (= the number of explanatory variables in terms of traditional demand forecasting models). The ability to have a large number of input nodes is also a key contribution to the success of deep-learning models. Each node in the input layer is linked to every node in the first hidden layer, and each node in the first hidden layer connects to every node in the second hidden layer, and so on. Each connection between nodes has its own weight. A node in the first hidden layer takes on the activation function value for the sum of weighted input values [see Eq. (1)]. A node in the second to the last of the hidden layers has an activation function value for the sum of the weighted node values in the previous hidden layer [see Eq. (2)]. A Sigmoid function is the most popular activation function in DNNs.

for (1)

for and (2)

where, represents the input variable, is the activation value of the node in the layer, is the weight for the connection between the node in the layer

and the node in the layer, is the number of hidden nodes, is the number of layers, and is the number of input variables.

Each output node in the output layer takes on an activation value of from 0 to 1, which represents the probability that the corresponding class is chosen. Thus, the number of output nodes should be consistent with the number of classes. The present study adopted 5 classes to represent the bus-demand level, which realistically reflects the actual profile of bus demand. A merit of dividing bus demand into a predefined number of classes is in the ability to measure the model performance more straightforwardly. Performance measures for a single target value, such as a root mean square error (RMSE) and a mean absolute percent error (MAPE), are less intuitive than measuring a matching rate directly for discrete levels. Eq. (3) denotes the estimated probability for each node in the output layer. The label estimate for an input feature vector is determined from the output probabilities, which will be compared with the observed label while training a DNN based on a back-propagation algorithm. That is, 1 is imposed to a node that has the greatest output probability, and 0s are given to the other nodes.

for (3)

where represents the estimated output probability for the output node, and is the number of possible output classes.

**2.2 The structure of a deep belief network (DBN)**

Most researchers who have used a DNN have the common experience of encountering a situation whereby an incumbent solution is stuck in a local minimum while implementing a back-propagation algorithm. To avoid this situation, robust initial solutions are necessary for weight parameters. Together with the use of Big-data, adopting a deep belief network (DBN) is known to be another key to secure a robust initial solution that can avert the situation. A DBN belongs to the category of unsupervised machine learning tools that requires no labeled data for training. It can be regarded as a tool for reducing the large dimension of input features to a tractable number of output nodes with the characteristics of the original features preserved. Another interpretation is that a DBN clusters input features (input nodes) into several homogeneous groups (output nodes). Consequently, weight parameters obtained after training a DBN could be a good initial solution for training a DNN with the same structure, prior to implementing a back-propagation algorithm.

A DBN is a stacked version of multiple restricted Boltzmann machines (RBMs). An RBM has only two layers with the nodes connected between them, which is the simplified version of a general Boltzmann machine that connects every pair of nodes both within and between the layers. Fig. 2 shows the structure of an RBM. In this section, the structure of a DBN is briefly introduced according to the interpretations of Fischer and Igel (2012).

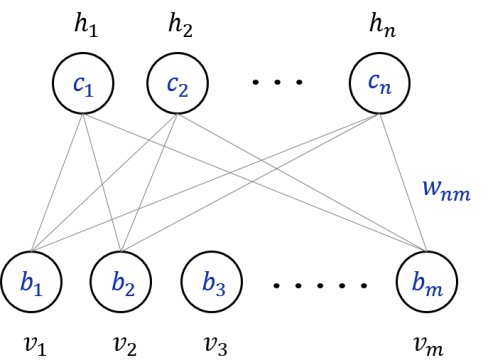


Fig. 2. An illustration of a restricted Boltzmann machine

As shown in Fig. 2, an RBM is composed of m visible nodes and n hidden nodes. Hidden nodes play the role of capturing dependencies between observed variables. represents an observed variable, and represents a hidden variable. is a weight parameter connecting these two variables. and are bias parameters for and , respectively. Parameters to be estimated through training are represented by . Random variables are assumed to be jointly distributed by a Boltzmann distribution as follows.

(4)

where, .

Energy function can be chosen between the following two options according to variable types. An RBM is a unit component to form a DBN. A DBN is created by stacking multiple RBMs. Fig. 3 shows the multi-layer architecture of a DBN. Eq. (5) represents the energy function of an RBM with real observed variables and binary hidden variables. A Gaussian-Bernoulli RBM usually composes the bottom two layers of a DBN, since input features are likely to be real values. All upper RBMS in a DBN take the Bernoulli-Bernoulli form [Eq. (6)], since the hidden layer of a lower RBM becomes the observed layer of its upper RBM.

●Gaussian (observed variables)-Bernoulli (hidden variables)

(5)

●Bernoulli (observed variables)-Bernoulli (hidden variables)

(6)

The likelihood function of an RBM can be established by summing the joint probability across all possible values of hidden variables [see Eq. (7)]. When training an RBM, parameters are estimated so that the likelihood function can be maximized. Details of the training algorithm will be introduced in the next section.

(7)

As shown in Eqs. (8) to (11), conditional probabilities which are necessary when training an RBM can be expressed in a simple form, since an RMB does not connect between nodes within a layer.

●Bernoulli (observed variables)-Bernoulli (hidden variables)

(8)

(9)

●Gaussian (observed variables)-Bernoulli (hidden variables)

(10)

(11)

where, sig() stands for sigmoid function [ ] and N() denotes Gaussian density function.

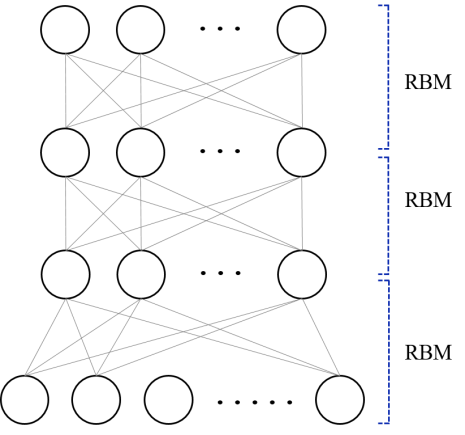


Fig. 3. An example of a Deep Belief Network with three layers

The training procedure of a DBN (= stacked RBMs) is as follows. Once a Gaussian-Bernoulli RBM or Bernoulli-Bernoulli RBM are learned, activation probabilities for hidden units are used as input visible observations for training the Bernoulli-Bernoulli RBM one layer up. This process repeats until the top layer is encountered. After completing the calibration up to the topmost RBM, calibrated weight parameters of all RBMs from bottom to top are used as initial weights for a DNN with the same structure except for an output layer. Prior to implementing a back-propagation training algorithm for the DNN, initial weights between the output layer and the topmost hidden layer of a DBN are chosen randomly. Another possible option to determine the weights is to implement a multinomial regression model using nodes within the output layer as dependent variables and nodes within the topmost hidden layer as independent variables.

**3. Training method for a DNN**

Obtaining a large amount of labeled data from the Internet is not easy when recognizing images, voices, and text, unless a data uploader deliberately attaches a relevant nametag for it. Most researchers in the field of machine learning studies have a limited amount of labeled data for training their own models. For this reason, a pre-training methodology was developed based only on unlabeled data. The parameters derived from an unsupervised pre-training are then used as initial parameters for a DNN model based on a limited number of labeled data. On the other hand, Deng and Yu (2014) pointed out that a DNN with a random initialization of parameters could resolve the over-fitting problem once a large amount of labeled data is available, even without pre-training. Securing sufficient labeled data is relatively easy for the prediction of bus demand owing to smart-card data. The present study tested back-propagation algorithms with both random initialization and pre-training. This section briefly introduces how to implement a back-propagation algorithm and a DBN as pre-training tools.

**3.1 Back-propagation algorithm**

The back-propagation is not a new concept, but has been widely used to train ANNs. The algorithm is known to find a local minimum of the cost function of a DNN with respect to weight parameters. The mathematical programming for training a DNN is set up such that the cost function can be minimized. The cost function generally represents the discrepancy between the observed and estimated outputs. If each output node value is assumed to follow a binomial distribution, an appropriate cost function is the so-called cross entropy [see Eq. (12)]. More specifically, the expression within the bracket of the equation is the log-likelihood proportional to the probability that the observed level of bus demand matches the estimated level. Maximizing the sum of the log-likelihoods across a training sample yields the optimal weight parameters. The second term of the cost function is added to avoid over-fitting, which penalizes large parameters and is modulated by a regularization factor.

(12)

where characters in bold font represent a vector, the superscript represents the example of a training set, is sample size, is the regularization factor, is the number of nodes within the layer, and is an indicator for the observed class (i.e., 1 if the example’s observed class is , and 0 otherwise).

The core of the back-propagation algorithm is the ease of deriving the derivative of the cost function with respect to a given parameter set. However, it cannot be implemented in a practical amount of computation time, as the network size gets larger. This is also a reason why DNNs were rarely been adopted in the 80s and 90s. Recently, as the computing environment has improved, the back-propagation algorithm has becomes the main tool for training a DNN. In particular, the use of a graphics-processing unit (GPU) leads to the popularity of deep-learning models, which facilitates a back-propagation algorithm by computing the activation value of all nodes within a layer in parallel.

The resultant computing procedure for a back-propagation algorithm is very simple to code, as shown below. On the other hand, the derivation of the algorithm is not straightforward. Readers who are interested in it can refer to the literature (Ng, 2011).

|  |
| --- |
| **Training set**: (  **Initialize**: ,  **for do**  Compute forwardly for  Compute  Sequentially compute using the following recursive formula   This is only for a sigmoid activation function.  where, ‘.\*’ represents the element-wise multiplication.    if  if  for |

**3.2 Pre-training a DNN**

How to train a DBN model is presented in this section as a pre-training method for a DNN. A DBN is trained in a greedy manner such that multiple RBMs are sequentially learned from bottom to top, accepting the output probabilities of the lower RBM as the input to the next RBM one layer up. Therefore, a DBN is automatically trained after the training of all the RBMs that belong to it are completed.

A RBM is an unsupervised machine learning tool that requires no labeled data. The training algorithm that is used to estimate the parameters of a RBM adopts a gradient ascent methodology on the log-likelihood (). However, as shown in Eq. (7), the likelihood function of a RBM is expressed as a marginal probability of observed variables, which sums the joint probability of observed and hidden variables across all possible hidden variables. This summing operation makes it difficult to evaluate the function and then to compute its gradient. Computing the gradient of log-likelihood can be approximated by the k-step contrastive divergence (CD) algorithm. This algorithm depends upon a Gibbs sampler that easily samples random draws from a joint probability density function using a Markov chain Monte Carlo simulation. The resultant algorithm below is short and easy to code. For brevity, the details of how the algorithm was derived are skipped in the present study, but readers who are interested in them can refer to Fischer and Igel (2012).

|  |
| --- |
| **Input**: RBM (  **Output**: gradient approximation  **Determine training rate**  **Initialize**  **for all thedo**  **for****do**  **for** **do** sample  **for** **do** sample  **for** **do**      ) |

**4. Data preparation**

Data for both training and testing a DNN were obtained from several sources. Data for output were derived from smart-card data, considering the profiles of bus demand. The bus ridership was aggregated for two different spatial units for each bus line: at the stop level or the stop-to-stop level. Smart-card data for five consecutive weekdays (2015/10/13-2015/10/17) were used to obtain an average daily bus ridership. Data for the input layer were extracted from multiple sources such as smart-card data, a bus management system (BMS), and land-use data from the national geographic information system (GIS).

**4.1 Preparing output data**

The present study used two spatial units to aggregate the bus demand of a specific bus line. The bus ridership was summed up at either the stop level or the stop-to-stop level for each bus line. The observed label for an output layer denoted the level of daily bus demand at the stop level or the stop-to-stop level for a specific bus line. To avoid confusion, a bus stop that belongs to a specific bus line will hereafter be defined as a line-stop. The line-stop, as a spatial unit of the present study, should be distinguished from a general bus stop that accommodates multiple bus lines. Accordingly, the term “stop” within both “the stop level” and “the stop-to-stop level” should be regarded as a line-stop.

Distributions of bus ridership at both spatial levels are depicted in Fig. 4. Bus ridership proved to be distributed by a negative exponential density. All ridership data were divided into 5 bins according to the magnitude, as shown in Fig. 4. The real scale was used to divide bus demand at the stop level, so that the width of each bin would be identical. On the other hand, the cut-off values to divide the bus demand at the station-to-station level were determined on a log-scale, since the bus demand profile exhibited a more extreme tail. By doing so, the bin width increased as the bus-demand level increased.



|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Level 1 | Level 2 | Level 3 | Level 4 | Level 5 |  |  | Level 1 | Level 2 | Level 3 | Level 4 | Level 5 |
| No. of line-stops | 13,643 | 6,052 | 3,262 | 1,623 | 1,971 | No.. of line-stop to line-stop pairs | 319,119 | 91,884 | 39,151 | 10,298 | 2,073 |
| Proportion | 51.4% | 22.8% | 12.3% | 6.1% | 7.4% | Proportion | 68.9% | 19.9% | 8.5% | 2.2% | 0.5% |
| Daily bus ridership | 0100 | 100200 | 200300 | 300400 | >400 | Daily bus ridership | 0  (07) | (720) | (2055) | (55150) | (>150) |

1. Stop-level bus demand (b) Stop-to-stop-level bus demand

Fig. 4. Bus-demand profile

**4.2 Preparing input data**

Input features for the present model were categorized into two groups: activity-related variables and supply-related variables. Floor areas for each building were stored and maintained for taxation purposes by the Korean government. We obtained the floor data for our testbed, which covered the Seoul metropolitan area. This data were expected to address hidden motivations behind bus ridership demand. The floor area was divided by 28 different uses encompassing residence, commercial, office, exhibition, and so on. Whereas the size of the walkable catchment area around a subway station is known as 400 m (0.25 mile), few studies have dealt with the relevant catchment area for a bus stop. According to the convention of the Seoul metropolitan government, a 250 m radius around a bus stop was determined to be the catchment area. Thus, building floor areas were aggregated within this spatial unit. In addition, the difference in building uses between the target line-stop and other downstream line-stops was hypothesized to be a driving force to generate bus demand. Euclidian distances between two different bus stops were computed in the feature space that 28 building-use variables represented, and then were adopted as variables to account for bus demand. It was expected that a longer distance would equate to more bus ridership.

The supply-related variables were again broken down into three groups: overall supply variables for a specific bus line, stop variables shared with other bus lines, and line-stop variables only for a specific bus line. Variables specific to a bus line included a daily averaged headway as well as headways during the morning peak, evening peak, and off-peak hours, all of which were obtained from the agency that operates the bus management system (BMS) in the Seoul metropolitan area.

The ridership at a specific bus stop of a specific bus line was assumed to be affected by that bus stop and other downstream bus stops along that bus line. Accordingly, the number of feature variables at the stop level was different according to which bus line a target line-stop belonged to and which position a target line-stop was located at along its bus line. The line-stop that had the maximum number of input variables governed the number of input nodes. This made most of the input feature vectors very sparse, which was a main reason that the computing time was increased when training a DNN.

This assumption was somewhat relaxed when forecasting bus ridership at the stop-to-stop level, since the origin and destination stops were specified. Activity-related variables only for both origin and destination stops were utilized in this case. Nonetheless, the length of input feature vectors varied across stop-to-stop pairs, since the input feature vector should cover supply-related variables for bus stops between a target stop-to-stop pair. Details of input variables are described for each category in Table 1. The maximum number of input variables for each variable category is shown within parentheses in the table.

Table 1. Input features and their maximum dimension

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Category | | Input feature | Stop level | Stop-to-stop level |
| Activity-related variables | | Building floor areas | Building floor areas for 28 types within the catchment area of 250 m radius around the target bus stop and its downstream bus stops along the target bus line (4,984) | Building floor areas for 28 types within both bus stops corresponding to the origin-destination pair (56) |
| Difference in building-uses | Euclidian distances in a feature space with 28 axes between building-use characteristics of the target bus stop and those of other downstream stops along the target bus line (178) | The Euclidian distance in a feature space with 28 axes between building-use characteristics of both bus stops corresponding to the target origin-destination pair (1) |
| Supply-related variables | Overall variables for a bus line | Headways | Headways for each segment of the target bus line. They are broken down into a daily averaged headway and headways during the morning peak, evening peak, and off-peak hours (712) | Headways for the origin bus stop of the target bus line. They are broken down into a daily averaged headway and headways during the morning peak, evening peak, and off-peak hours (4) |
| Variables shared with other bus lines | Stop sequence | Numbers indicating the location order of downstream stops along the target bus line from the target bus stop, with the target bus stop given 0 (178) | N/A |
| The number of transferrable bus lines | The number of bus lines that can be transferred at the target bus stop and its downstream stops along the target bus line (178)  The number of bus lines that can be transferred at neighbor bus stops within the radius of 50 m around the target bus stop and its downstream bus stops along the target bus line (178) | The number of bus lines that can be transferred at the origin bus stop and its neighbor stops within the radius of 50 m (2)  The number of bus lines that can be transferred at the destination bus stop and its neighbor stops within the radius of 50 m (2) |
| The total frequency of bus lines that can be transferred from the target line | The frequency of bus lines that can be transferred at the target bus stop and its downstream stops of the target bus line (712)  The frequency of bus lines that can be transferred at neighbor bus stops within the radius of 50 m around the target bus stop and its downstream bus stops along the target bus line (712)  (frequencies during the morning and evening peak hours and the off-peak hours, and on a daily average basis) | The frequency of bus lines that can be transferred at the origin bus stop and its neighbor stops within the radius of 50 m (8)  The frequency of bus lines that can be transferred at the destination bus stop and its neighbor stops within the radius of 50 m (8)  (frequencies during the morning and evening peak hours and the off-peak hours, and on a daily average basis) |
| The number of transferable subway lines nearby | The number of subway lines within the radius of 250 m around the target bus stop and its downstream stops along the target bus line (178) | The number of subway lines within the radius of 250 m around both end bus stops corresponding to the target origin-destination pair (2) |
| Variables for a specific line-stop | The frequency of competitive bus lines (Overlapping level) | The frequency of competitive bus lines passing downstream segments of the target bus stop. Each segment should connect consecutive bus stops along the target bus line.  (frequencies during the morning and evening peak hours and the off-peak hours, and on a daily average basis) (712) | The frequency of competitive bus lines passing segments within the itinerary that connects the target origin-destination pair.  (frequencies during the morning and evening peak hours and the off-peak hours, and on a daily average basis)  (68) |
| Network Distances between stations along the target bus line | The network distance from the target bus stop to each downstream bus stop along the target bus line (178) | The network distance between the target origin-destination bus stops (1) |
| Travel times | The travel time from the target bus stop to each downstream bus stop along the target bus line during the morning and evening peak hours and the off-peak hours, and on a daily average basis (712) | The travel time between the target origin-destination bus stops during the morning and evening peak hours and the off-peak hours, and on a daily average basis (4) |
| Aerial distances | The aerial distance from the target bus stop to each downstream bus stop along the target bus line (178) | The aerial distance between the target origin-destination bus stops (1) |

The total number of variables was 9,790 and 157 for the stop-level and the stop-to-stop-level models, respectively. The existing models have never attempted to mobilize such a vast number of explanatory variables to forecast travel demand. When adopting a shallow-learning model like a linear regression, multicollinearity should be avoided in order to acquire a robust modeling result. Thus, it has long been recommended in the field of trip generation studies that independent variables that correlate with one another should not be included. However, this is no longer a critical issue in deep-learning modes. That is, no matter how many features are employed to feed a DNN, their influences could separately contribute to estimating travel demand, once a large amount of data are available. A deep-learning model is also capable of intrinsically considering nonlinear inter-relations between feature variables. There is no need to account for multicollinearities among potential variables that are expected to affect the target variable.

A sufficient amount of data is necessary for a DNN to yield a robust result. However, at the stop level the number of feature variables was relatively large compared with the number of line-stops available. The number of feature variables depended upon the length of the target bus line and where the target bus stop was located. In the Seoul metropolitan area, the length of bus lines is determined by counting the number of bus stops along the line, and the number of stops can range from 15 to 178. The lengths were examined in only one direction if a bus line offered round-trip service. Since the dimension of input features for each target line-stop had to be set as the maximum length (=178), the feature vector for most target line-stops took a very sparse form. This had a negative impact on the model performance, and entailed a computational burden. The results are discussed in the section that follows.

**5. Modeling results and sensitivity analysis**

**5.1 Training and testing results**

The sample size for both training and testing was 26,551 for the stop-level model and 462,525 for the stop-to-stop-level model. The former corresponded to the number of available line-stops, and the latter to the number of stop-to-stop pairs of all available bus lines. Among the data of the stop-level model, 1,000 were randomly chosen for testing, and the rest were used for training. From the sample data of the stop-to-stop model, 5,000 were chosen at random for testing. For the stop-level model, each sample example was composed of a line-stop’s ridership level and the features of the line-stop and its downstream line-stops were observed. For the stop-to-stop model, each sample example represented the bus demand for a specific pair of line-stops and influential features for the line-stop pair.

The base architecture of the stop-level model was built such that the number of hidden layers was 3, the number of output nodes was 5, the number of hidden nodes in a layer was 100, and the number of input nodes was 9,790. In a similar manner, the base structure for the stop-to-stop-level model was established such that the number of hidden layers was 3, the number of output nodes was 5, the number of hidden nodes in a layer was 20, and the number of input nodes was 157. The relative weight factor for the regularization was set at 1 for the base models. These hyper-parameters for the two base architectures were determined via sensitivity analysis, the details of which will be described in the next subsection. The matching accuracy rate between the estimated and observed ridership levels was selected as the overall performance measure.

A back-propagation algorithm was chosen as the base training algorithm. Two methods were adopted in order to provide the initial weight parameters for the algorithm. In the first method the initial weights were randomly chosen, and in the second version the pre-training process was based on a DBN. The latter method is known to obtain a more robust initial solution that could avert the frequent phenomenon whereby a solution becomes stuck at a local minimum. However, the present study encountered a handicap when adopting this method. Although an extra set of unlabeled data is necessary for unsupervised pre-training, the same data set that was used for pre-training had to be re-used in the training process. In this case, we examined the possibility that the model performance could be enhanced.

Fig. 5 depicts the training and testing results for both the stop-level model and the stop-to-stop version, where hyper-parameters of the base case were applied. The stop-level model showed a typical problem of over-fitting, wherein the test accuracy slightly deteriorates or remains invariant as the training accuracy increases. The rate of estimated demand levels matched the observed level 94.76% during training, while the final matching rate for the corresponding test was at most 52.3% [see Fig.5 (a)]. This could have been due to an insufficient amount of training data in contrast to the relatively large number of features. There was a strong limitation whereby the training sample size for the stop level model could not exceed the number of line-stops. On the other hand, the length of a feature vector was very long since it had to cover all possible factors that might affect the line-stop ridership demand (see Table 1).



1. Results from the stop-level model with a randomly chosen initial solution



1. Results from the stop-level model with the initial solution from pre-training



1. Results from the stop-to-stop-level model with a randomly chosen initial solution



1. Results from the stop-to-stop-level model with the initial solution from Pre-training

Fig. 5 Training and testing results

The stop-to-stop-level model showed a more accurate matching result than the stop-level version. The matching rate for the test sample rose to 71.94%, keeping pace with the trend of the matching rate (=73.1%) for training. Unlike the stop-level model, the stop-to-stop version had a relatively small number of input features, since the spatial unit of the target bus demand specified a single pair of origin and destination inputs. This specification made it possible to secure a large amount of training data. In summary, the data set for the stop-to-stop level was horizontally narrow and vertically long, while that for the stop level was horizontally wide and vertically short.

Regarding the pre-training method, Fig 6 depicts the change of the log-likelihood of DBNs in terms of learning epochs. Since a DBN depended upon a simulation to evaluate the derivative of the log-likelihood, the convergence curve was not smooth. However, the trend was apparently toward maximizing the log-likelihood. Table 2 shows an environment in which a DBN was trained to acquire an initial solution. As a result, the pre-training method did not have a positive impact on improving the forecasting accuracy in the present study [see Fig. 5 (b) and (d)]. The accuracy was not improved since the same training data were used for both pre-training and in the formal training later. It is expected that the result could be ameliorated if another data set for pre-training were available for bus-demand forecasting, which was confirmed in the previous studies for the pattern recognitions of images, voices and text. On the other hand, if a larger amount of labeled data were available, a pre-training process would be unnecessary (Deng and Yu, 2014).

|  |  |  |
| --- | --- | --- |
| Gaussian-Bernoulli RBM(the 1st layer) | Bernoulli-Bernoulli RBM(the 2nd layer) | Bernoulli-Bernoulli RBM(the 3rd layer) |
| C:\Users\Junghan\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\stop_level_01_02.png | C:\Users\Junghan\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\stop_level_02_03.png | C:\Users\Junghan\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\stop_level_03_04.png |

(a) Stop-level model training

|  |  |  |
| --- | --- | --- |
| Gaussian-Bernoulli RBM(the 1st layer) | Bernoulli-Bernoulli RBM(the 2nd layer) | Bernoulli-Bernoulli RBM(the 3rd layer) |
| C:\Users\Junghan\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\stop_to_stop_level_01_02.png | C:\Users\Junghan\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\stop_to_level_02_03.png | C:\Users\Junghan\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\stop_to_stop_level_03_04.png |

(b) Stop-to-stop-level model training

Fig. 6 Convergence of DBNs

Table 2. Environment for pre-training (Learning DBNs)

1. Stop-level model

|  | # of observations  (m) | # of hidden units (n) | Sample size | CD-k | # of iterations | Training rate( |
| --- | --- | --- | --- | --- | --- | --- |
| 1st layer  (bottom) | 9790 | 1000 | 26551 | 2 | 200 | 0.00000001 |
| 2nd layer | 100 | 100 | 26551 | 2 | 100 | 0.00001 |
| 3rd layer3  (top) | 100 | 100 | 26551 | 2 | 100 | 0.00005 |

1. Stop-to-stop-level model

|  | # of observations  (m) | # of hidden units (n) | Sample size | CD-k | # of iterations | Training rate( |
| --- | --- | --- | --- | --- | --- | --- |
| 1st layer  (bottom) | 157 | 20 | 462525 | 2 | 200 | 0.00000001 |
| 2nd layer | 20 | 20 | 462525 | 2 | 100 | 0.00001 |
| 3rd layer3  (top) | 20 | 20 | 462525 | 2 | 100 | 0.00005 |

The matching rate for testing the stop-level model seemed unsatisfactory. However, the potential for deep-learning with respect to the transportation-demand analysis can be found in the cross-validation analysis. Fig. 7 plots the cost function value [see Eq. (12)] without the regularization term for both training and test data sets by altering the size of the training data set, with the test data set fixed. To save computing time, the results from only 2,000 iterations were recorded. As shown in Fig. 7 (a), the final discrepancy between two cost function values were relatively large when using all available training data (=100%), which means the model overfits for the training data with high variance. In other words, there was a possibility that the test error rate can be reduced if more training data were provided, even though the training error will increase. As mentioned earlier, the data set for the stop-level model has a limitation whereby more data cannot be collected in the testbed unless new bus stops are installed. Actually, even though the bus ridership data were collected form a Big-data (=smart-card data), it was difficult to show the power of Big-data as a key to train a DNN since they were aggregated for the limited number of bus stops. Data from other cities is expected to enlarge the stop-level data set in the near future.

The test matching rate was much more accurate for the stop-to-stop-level model than it was for the stop-level model. However, the stop-to-stop-level model also was unsatisfactory compared with the success of deep-learning technologies in the field of artificial intelligence. Unfortunately, as shown in Fig. 7 (b), increasing the amount of training data is not expected to dramatically reduce the cost-function value, since the final gap between cost-function values for training and testing was relatively small. In this case, the model underfits the training data, so vertically extending training data cannot contribute the model performance. Rather, it would be more important to extend the horizontal width of training data. That is, new influential features should be excavated and then the model architecture could also be extended to accommodate them. As far as we could ascertain, this seemingly unsatisfactory results were better than that from a shallow model (see the next subsection), and better than those from previous direct demand forecasting models (Choi et al., 2012; Sohn and Shim, 2010).

(a) Stop-level model (b) Stop-to-stop-level model

Fig. 7 Results from cross-validation

Table 3 shows the details of the matching results from the testing. The first two bins contained the most ridership data (78% for the stop level and 87% for the stop-to-stop level) and revealed very high matching rates, which deteriorated as the demand level increased. This might have been due to a lack of data from levels with a higher demand. For example, the amount of data for level 5 was less than 0.5% out of the total amount of data for the stop-to-stop-level model. This limitation could be resolved if more data for the higher-demand levels were secured in the future.

Table 3. Normal and relaxed matching rates for test

1. Stop-level model

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Observed Predicted | Level 1 | Level 2 | Level 3 | Level 4 | Level 5 |
| Level 1 | 368 | 89 | 38 | 11 | 11 |
| Level 2 | 90 | 93 | 45 | 8 | 8 |
| Level 3 | 24 | 42 | 29 | 16 | 22 |
| Level 4 | 6 | 8 | 10 | 6 | 7 |
| Level 5 | 4 | 14 | 17 | 10 | 24 |
| Normal matching rate | **74.80%** | **37.80%** | 20.86% | 11.76% | 33.33% |

1. Stop-to-stop-level model

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Observed Predicted | Level 1 | Level 2 | Level 3 | Level 4 | Level 5 |
| Level 1 | 3,157 | 631 | 166 | 25 | 3 |
| Level 2 | 190 | 324 | 188 | 33 | 8 |
| Level 3 | 21 | 72 | 107 | 46 | 10 |
| Level 4 | 0 | 3 | 2 | 9 | 5 |
| Level 5 | 0 | 0 | 0 | 0 | 0 |
| Normal matching rate | **93.74%** | **31.46%** | 23.11% | 7.96% | 0.00% |

**5.2 Sensitivity analysis with respect to three hyper-parameters**

Three important hyper-parameters were chosen to investigate their influence on the model performance: the number of hidden layers, the number of hidden nodes within a hidden layer, and the regularization factor. Six different levels of each hyper-parameter were tested including the base case defined in the previous subsection. The sensitivity analysis was conducted by altering the parameter values one-by-one, with other parameter values fixed at the base case. Accordingly, 18 different cases were examined. Results from the sensitivity analysis are shown in Fig. 8.

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
| 1. Stop level model | 1. Stop-to-stop level |

Fig. 8. Sensitivity results

Optimal hyper-parameters for the base cases were determined, so that the test-matching rate should have been maximized. Every optimal hyper-parameter value for testing was consistent with that for training. As shown in Fig. 8, it was apparent that three hidden layers yielded the best matching rate for both training and testing. This is the most meaningful finding of the present study, since thus far transport researchers have dealt with neural networks with a single hidden layer. As far as we could ascertain, this is the first confirmation that a deep-learning architecture outperforms its shallow-learning counterpart with a single hidden layer in forecasting travel demand. It turned out that the stop-level model required a much larger number of hidden nodes within a hidden layer than the stop-to-stop-level model, which implies that it should be proportional to the number of input features. The optimal regularization factor proved to be 1 for both models.

**6. Conclusions**

Forecasting travel demand has historically depended upon the conventional 4-step models, which are based on a specific logic or theory. The present study proposed a novel approach for forecasting bus demand using only data with no specific theory. Two deep-learning architectures were presented to deal with a relatively large data set acquired from smart-card data. Various potential variables were excavated, and a robust methodology was presented to incorporate them into a deep-learning model. As a result, for stop-to-stop-level forecasting, the test results showed promise, while stop-level forecasting yielded somewhat unsatisfactory results. A pre-training methodology was also adopted with the expectation that formal training could be more successful, but the model performance was not enhanced. As a potential conclusion, it turned out that providing as much labeled data as possible was more important than pre-training based on unlabeled data.

The performance of the stop-level model is expected to improve when additional labeled data can be obtained from other cities. On the other hand, the stop-to-stop-level model required additional feature variables that should enhance its explanation power. It is apparent that a deep-learning approach will be a great success in forecasting travel demand given that the data necessary to fulfill these two requirements are secured in the near future. The present study takes a step forward in a new direction by using a data-driven approach in forecasting travel demand.

**Acknowledgement**

This research was supported by the Chung-Ang University Research Grants in 2016, and also supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIP) (2015005843).

**References**

Chu, X., 2004. Ridership models at the stop level National Center of Transit Research. University of South Florida, Tampa.

Choi, J., Lee, Y. J., Kim, T., & Sohn, K., 2012. An analysis of Metro ridership at the station-to-station level in Seoul. Transportation, 39(3), 705-722.

Dargay, J. M., & Hanly, M., 2002. The demand for local bus services in England. Journal of Transport Economics and Policy, 73-91.

Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., ... & Ng, A. Y., 2012. Large scale distributed deep networks. In Advances in Neural Information Processing Systems (pp. 1223-1231).

Deng, L., & Yu, D., 2014. Deep learning: Methods and applications. Foundations and Trends in Signal Processing, 7(3–4), 197-387.

Dill, J., Schlossberg, M., Ma, L., & Meyer, C., 2013. Predicting transit ridership at the stop level: The role of service and urban form. Transportation Research Board.

Estupiñán, N., & Rodríguez, D. A., 2008. The relationship between urban form and station boardings for Bogota’s BRT. Transportation Research Part A: Policy and Practice, 42(2), 296-306.

Fischer, A., & Igel, C., 2012. An introduction to restricted Boltzmann machines. In Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications (pp. 14-36). Springer Berlin Heidelberg.

Hashemi, R. R., Le Blanc, L. A., Rucks, C. T., & Shearry, A., 1995. A neural network for transportation safety modeling. Expert Systems with Applications, 9(3), 247-256.

Hinton, G. E., Osindero, S., & Teh, Y. W., 2006. A fast learning algorithm for deep belief nets. Neural computation, 18(7), 1527-1554.

Hoogendoorn-Lanser, S., van Nes, R., & Bovy, P., 2005. Path size modeling in multimodal route choice analysis. Transportation Research Record: Journal of the Transportation Research Board, (1921), 27-34.

Huang, W., Hong, H., Li, M., Hu, W., Song, G., & Xie, K., 2013. Deep architecture for traffic flow prediction. In Advanced Data Mining and Applications (pp. 165-176). Springer Berlin Heidelberg.

Kuby, M., Barranda, A., & Upchurch, C., 2004. Factors influencing light-rail station boardings in the United States. Transportation Research Part A: Policy and Practice, 38(3), 223-247.

Lam, S. H., & Xie, F., 2002. Transit path-choice models that use revealed preference and stated preference data. Transportation Research Record: Journal of the Transportation Research Board, (1799), 58-65.

Ledoux, C., 1997. An urban traffic flow model integrating neural networks. Transportation Research Part C: Emerging Technologies, 5(5), 287-300.

Lemieux, J., & Ma, Y., 2015. Vehicle Speed Prediction using Deep Learning. In Vehicle Power and Propulsion Conference (VPPC), 2015 IEEE, 1-5

Lv, Y., Duan, Y., Kang, W., Li, Z., & Wang, F. Y., 2015. Traffic flow prediction with big data: a deep learning approach. Intelligent Transportation Systems, IEEE Transactions on, 16(2), 865-873.

Ma, X., Tao, Z., Wang, Y., Yu, H., & Wang, Y., 2015. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. Transportation Research Part C: Emerging Technologies, 54, 187-197.

Moniruzzaman, M., Maoh, H., & Anderson, W., 2016. Short-term prediction of border crossing time and traffic volume for commercial trucks: A case study for the Ambassador Bridge. Transportation Research Part C: Emerging Technologies, 63, 182-194.

Ng, A., 2011 CS294A Lecture notes, Stanford University, https://web.stanford.edu/class/cs294a/sparseAutoencoder\_2011new.pdf

Park, B., Messer, C., & Urbanik II, T., 1998. Short-term freeway traffic volume forecasting using radial basis function neural network. Transportation Research Record: Journal of the Transportation Research Board, (1651), 39-47.

Parsons Brinckerhoff Quade and Douglas, Inc., 1996. Transit and Urban Form. National Academy Press, Washington, DC.

Pulugurtha, S. S., & Agurla, M., 2012. Assessment of Models to Estimate Bus-Stop Level Transit Ridership using Spatial Modeling Methods. Journal of Public Transportation, 15(1), 3.

Ramos-Santiago, L. E., & Brown, J., 2015. A comparative assessment of the factors associated with station-level streetcar versus light rail transit ridership in the United States. Urban Studies, 0042098015571057.

Ritchie, S. G., & Cheu, R. L., 1993. Simulation of freeway incident detection using artificial neural networks. Transportation Research Part C: Emerging Technologies, 1(3), 203-217

Smith, B. L., & Demetsky, M. J., 1994. Short-term traffic flow prediction: neural network approach. Transportation Research Record, (1453)

Sohn, K., & Shim, H., 2010. Factors generating boardings at metro stations in the Seoul metropolitan area. Cities, 27(5), 358-368.

Yai, T., Iwakura, S., & Morichi, S., 1997. Multinomial probit with structured covariance for route choice behavior. Transportation Research Part B: Methodological, 31(3), 195-207.

Zhao, J., Deng, W., Song, Y., & Zhu, Y., 2013. What influences Metro station ridership in China? Insights from Nanjing. Cities, 35, 114-124.

Zhao, J., Deng, W., Song, Y., & Zhu, Y., 2014. Analysis of Metro ridership at station level and station-to-station level in Nanjing: an approach based on direct demand models. Transportation, 41(1), 133-155.

Zhu, J. Z., Cao, J. X., & Zhu, Y., 2014. Traffic volume forecasting based on radial basis function neural network with the consideration of traffic flows at the adjacent intersections. Transportation Research Part C: Emerging Technologies, 47, 139-154.