Expert Systems With Applications 139 (2020) 112864

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

# An integrated feature learning approach using deep learning for travel time prediction

Mohammad Abdollahi[*], Tannaz Khaleghi, Kai Yang

*Department of Industrial & Systems Engineering, Wayne State University, Detroit, USA*

A R T I C L E I N F O

*Article history:*
Received 3 April 2019
Revised 29 July 2019
Accepted 5 August 2019
Available online 6 August 2019

*Keywords:*
Deep learning
Stacked autoencoder
Transportation
Travel time prediction
Deep neural network
Spatiotemporal analysis

A B S T R A C T

Travel time data is a vital factor for numbers of performance measures in transportation systems. Travel time prediction is both a challenging and interesting problem in ITS, because of the underlying traffic and events' hidden patterns. In this study, we propose a multi-step deep-learning-based algorithm for predicting travel time. Our algorithm starts with data pre-processing. Then, the data is augmented by incorporating external datasets. Moreover, extensive feature learning and engineering such as spatiotemporal feature analysis, feature extraction, and clustering algorithms is applied to improve the feature space. Furthermore, for representing features we used a deep stacked autoencoder with dropout layer as regularizer. Finally, a deep multi-layer perceptron is trained to predict travel times. For testing our predictive accuracy, we used a 5-fold cross validation to test the generalization of our predictive model. As we observed, the performance of the proposed algorithm is on average 4 min better than applying the deep neural network to the initial feature space. Furthermore, we have noticed that representation learning using stacked autoencoders makes our learner robust to overfitting. Moreover, our algorithm is capable of capturing the general dynamics of the traffic, however further works need to be done for some rare events which impact travel time prediction significantly.

## 1. Introduction

Travel time is a baseline measure in transportation systems. Hence, its accurate prediction is of great importance to the development of advanced traffic management system (ATMS), intelligent transportation system (ITS), and other transportation systems. From travelers' perspective, information about traffic and travel times can help choose better roads, estimate the expected time to arrive, and reach the destination point faster by selecting the best routes or re-routing. Additionally, transportation agencies may control the traffic and reduce the congestion in more efficient way by using an accurate travel time prediction tool. To analyze and predict the travel time, traffic data can be utilized which may be collected from various sources (Turner, Eisele, Benz, & Holdener, 1998) such as global positioning system (GPS), cellular geolocation, video cameras, automatic, number plate recognition system (ANPR), automatic vehicle identification system (AVI), and etc. These sources can be used in ITS depending on the specific purpose and task. Generally speaking, GPS is most favorable data for the travel time prediction because each GPS can record some useful information, such as latitude, longitude, time stamp, speed, and other metadata.

An unbiased and low variance estimation of travel time can be used in design and operations, transportation management and planning, and measurements and evaluation. Specially, trip duration data are considered to be crucial pre-trip or even en-route information in ATIS. As mentioned earlier, they are very informative to both travelers and drivers for making better decision or scheduling their trip more efficiently.

In general, travel time estimation's benefits are three-fold. From the travelers' perspective, trip duration information helps to save time and improve the reliability through the selection of routes before starting the trip and en-route. In logistics application, this information may increase the reliability of delivery, cut the delivery costs, and even improve the service quality (Abdollahi, Arvan, Omidvar, & Ameri, 2014). Note that for traffic planners and managers, this information is key performance index for traffic system operation.

The rest of this paper is organized as follows. In the next section we will study the relate research effort to travel time prediction followed by 3 where the problem under study and dataset used in this study is explained. Later in Section 4 we present our

[*] Corresponding author.
*E-mail addresses:* abdollahi@wayne.edu (M. Abdollahi), t.kh@wayne.edu (T. Khaleghi), Kai.Yang@wayne.edu (K. Yang).

https://doi.org/10.1016/j.eswa.2019.112864
0957-4174/© 2019 Elsevier Ltd. All rights reserved.

integrated approach to solve travel time prediction problem followed by Section 5 where lost of experimental results is conducted and insights are presented. Lastly, in Section 5 we conclude this paper and provide future directions to move.

## 2. Literature review

There are many studies applying machine learning approaches for estimating travel time. Machine learning tools are equipped with strong statistical concepts which offer computer systems the ability to *learn* the system's behaviour with data, without being explicitly programmed. Besides travel time prediction, these approaches have been widely used in many different settings such as helathcare planning (Khaleghi, Abdollahi, & Murat, 2019; Khaleghi, Murat, & Neemuchwala, 2016), public policies (Mkanta, Chumbler, Yang, Saigal, & Abdollahi, 2016; Mkanta et al., 2017), energy (Azadeh, Jafari-Marandi, Abdollahi, & Roudi, 2017; Azadeh, Taghipour, Asadzadeh, & Abdollahi, 2014), and etc.

For reviewing the literature, this study adopts a systematic approach to investigate travel time prediction. Systematic literature reviews take a more scientific and transparent approach in finding research materials, therefore the biases corresponding the selection of the relevant papers are minimized (Petticrew & Roberts, 2008). Referring to the most recent literature review in the subject provided by Oh, Byon, Jang, and Yeo (2018) the following keywords were extracted: *travel time, trip duration* keywords were presenting the research area while *deep learning* was used for research focus. Fig. 1 shows the distribution of papers and most published journals in travel time prediction field, respectively. Scopus search engine was employed in order to find relevant papers by using the aforementioned keyword structure.

Some of the researchers in the filed of travel time/traffic prediction, have classified the prediction scheme into short, medium, and long-term horizons based on duration variations. Van Lint (2004) defines the short-term prediction as the one whose time horizon ranges from 0 to 60 min, and long-term for the ones which will take more than a day. Shen (2008) claims that considering a proper time horizon is the success key in implementing a travel time prediction system. Yet another categorization to travel time prediction relates to the road characteristics — signalized (arterial) or highways. It is perceived that predicting travel time in signalized road is inherently more complex than highways due to additional factors such as signal cycles from multiple intersection which is connected to each other.

Moreover, another group can be mentioned as the spatial scope of the prediction, whether a small area or a larger section of the road network is considered for prediction task. From solution perspective, we can also classify the prediction schemes into data-driven or model-based approaches. data-driven approach make use of historical data to predict for future, while model-based systems use of simulations to mimic the real-world behavior of traffic and making predictions based on that. In general, different settings can be categorized as Table 1.

**Table 1**
Different travel time prediction settings.

| Horizon | Road section | Spatial scope | Solution method |
|---|---|---|---|
| Short | City | Small | Simulation based |
| Medium | Highway | Large | Model based |
| Long | | | |

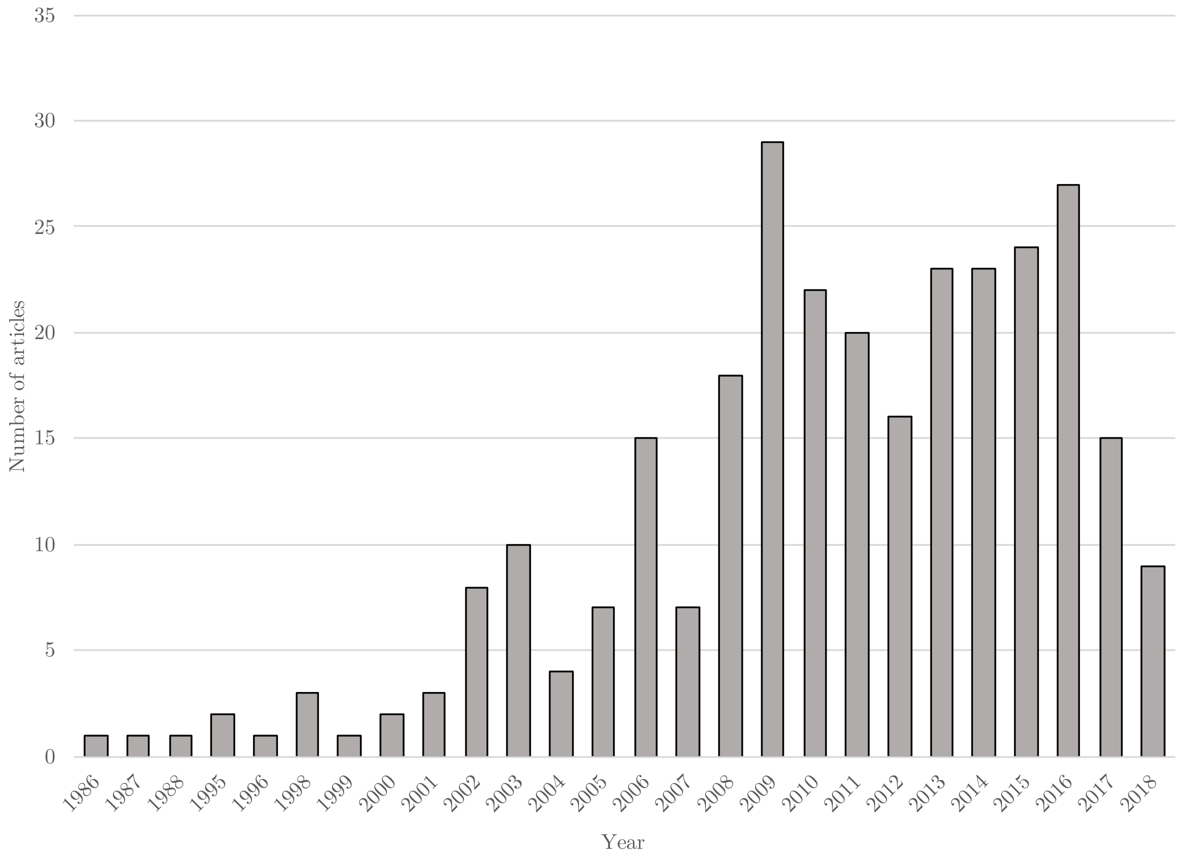


**Fig. 1.** Distribution of published papers in travel time prediction.

One of the most widely applied methods in travel time prediction is the classical time-series analysis. The base assumption of such models is that future values of data depend on the past values and random noises. The commonly used time-series models include Auto-Regressive Moving Average (ARMA) (Billings & Yang, 2006), seasonal ARIMA model (Guin, 2006), and generalized Auto-Regressive conditional heteroscedasticity model (Yang, Liu, & You, 2010). For example, Davies, Fortuna, Duke, Clarke, and Rupnik (2015) investigated the impact on predictive performance of local linear regression with using additional traffic information sources. The main contribution of this article is the extension of local linear models with higher order Auto-Regressive trip duration variables, namely speed, vehicle flow data, and density data. For reviewing more time-series related studies on travel time predictions, readers are referred to Kumar and Vanajakshi (2015) and Sun, Liu, Xiao, He, and Ran (2003).

Starting from 1980s, short-term traffic flow estimation has gained attention, which is considered to be beneficial for controlling real-time traffic (Okutani & Stephanedes, 1984). Due to their non-linearity handling power and function approximation capabilities, neural networks have been often deployed to estimate traffic flow. As an instance, Jiang and Adeli (2005) trained a time-delay recurrent wavelet neural network to forecast traffic flow. In addition, Ma, Tao, Wang, Yu, and Wang (2015) developed a LSTM architecture for traffic speed prediction and illustrated that this network is capable of catching long-term temporal dependencies in traffic data. In another study Zheng, Lee, and Shi (2006) integrated neural networks and Bayesian inference to forecast future traffic flow. In addition to the neural network models, there exist multiple other different methods such as SVR (Wang & Shi, 2013), Kalman filter (Guo, Huang, & Williams, 2014), the gradient boosting tree regression (Zhang & Haghani, 2015), KNN (Wu, Tan, Peter, Shen, & Ran, 2015), and some hybrid algorithms (Allström et al., 2016; Lopez-Garcia, Onieva, Osaba, Masegosa, & Perallos, 2016). The deficiencies of aforementioned methods stem from the contradiction between massive traffic data and their shallow structures (Lv, Duan, Kang, Li, & Wang, 2015).

To overcome the problem mentioned before, deep neural network architectures for traffic flow and travel time prediction can be developed. In this area, Huang, Song, Hong, and Xie (2014) used a deep belief network (DBN) to catch the traffic flow feature interdependency and dynamics, and proposed a multi-task learning model to do road and exit station flow prediction. Tan, Xuan, Wu, Zhong, and Ran (2016) looked into different deep neural networks pre-training schemes to predict traffic flow. Polson and Sokolov (2016) employed deep learning to forecast traffic flows during special events. These methods all belong to fully connected architectures, while there is no assumption about the features in the fully connected architecture. Hence it is hard to capture representative features from the dataset with plentiful characteristics using these types of network. Moreover, Zhang, Zheng, and Qi (2017) and Zhang, Zheng, Qi, Li, and Yi (2016) suggested a novel methodology based on the convolutional neural network to fully utilize the topological feature of urban crowd. Nevertheless, they simply treat time dimension of traffic flow as a channel of image data, which means the features of traffic flow on time dimension are ignored. Yang, Dillon, and Chen (2017) trained a stack denoise autoencoder model to learn hierarchical representation of urban traffic flow.

To sum up, travel time prediction is a challenging problem in ITS because of the underlying traffic and events hidden patterns. we noticed that trip duration can be estimated through a number of methods using various input data. Each of these architectures and solutions have their own cons and pros. While some travel time prediction exists in the literature, however, most of them are related to shallow learning architecture and they lack the feature learning ability. Hence, developing a deep architecture that fully represent the characteristics of traffic flow is yet an opening and intriguing question.

## 3. Problem definition

The main source of data in this study is New York City Taxi and Limousine Commission (TLC) trip records, which stores the record of billions of trips made in New York from January 2009 to December 2017. The yellow and green taxi trip records include columns which capture pick-up and drop-off dates and times, pick-up and drop-off locations, and etc. The full list of features is provided in Table 2. For analysis and training purposes, we just take data from January 2016 to June 2016. Pickup and drop-off points of the taxi rides are shown in Fig. 2. This data includes around 1.5 million trip records, from which we take 20% portion as hold out example for predictive performance analysis. As we stated, our horizon ranges from January to June 2016, and as such, we have taken this 20% hold-out as a stratified sample by weather dynamics, temporal (time of travel), and spatial (location) features to make sure we can capture most of the traffic patterns in the system. The selected data set (for yellow cabs) contains 19 features including redundant or unimportant features for our prediction task. As a result, we found that strengthening our feature space by using feature extraction and learning methods can significantly improve learnability. Besides that, to test our predictive performance one can define different measures. Mostly, researchers use either mean absolute error (MAE) as $\frac{1}{n}\sum_{i=1}^{n}|y_i-\hat{y}_i|$, mean absolute percentage error (MAPE) as $\frac{1}{n}\sum_{i=1}^{n}\frac{|y_i-\hat{y}_i|}{y_i}$, or root mean square error (RMSE) as $\frac{1}{n}\sqrt{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}$

As most of the researchers pointed out, the heterogeneity of the road segments plays a key role in the model evaluation process. For example, relative errors become less informative in cases where the links are not the same length. For instance, 50% MAPE on a 100 yards road segment can not be compared to a 50% MAPE on a 10 miles segment. This serves as a reason why researchers should put their focus on absolute errors more often. So, we will present our calculation and training power based on this measurement.

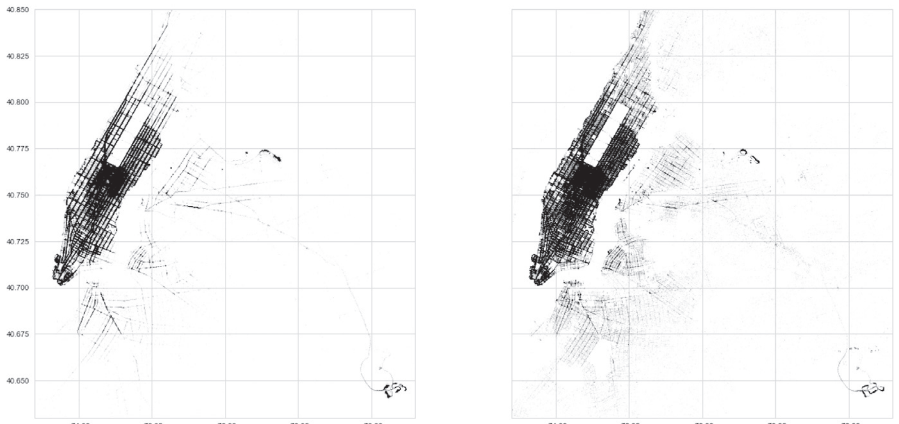
## 4. Proposed methodology

We developed an integrated multi-step methodology to predict the travel time. We provided data dictionary and feature definitions in previous section to shed lights on main and initial source of data that we have used for our prediction purpose, although, we have a multi-step approach to augment data and extract more features. The framework of our methodology is shown in Fig. 3. As shown in this figure, our prediction is composed of data cleaning and outlier detection, data augmentation, feature extraction, feature representation, and prediction steps. Further, each step will be discussed in detail.

Spatiotemporal analysis and pattern recognition can help to identify interesting patterns from both spatial and temporal data, significantly. However, this data requires substantial data cleaning and preprocessing before it can be investigated further by machine learning techniques. Such analysis helps the researchers and policy makers to understand the temporal and spatial fluctuations and variability of the information through visualization of the data. It should be noted that knowledge discovery from the spatiotemporal data with multi-dimensional parameters is a tedious effort (Mennis & Guo, 2009). It is equally time consuming task to perform preprocessing before using the data for prediction purposes (Ester, Frommelt, Kriegel, & Sander, 2000; Sharma, 2006). In this research, we

**Table 2**
TLC data dictionary.

| Feature | Definition |
|---|---|
| VendorID | A code indicating the TPEP provider that provided the record<br>• 1: Creative Mobile Technologies, LLC<br>• 2: VeriFone Inc. |
| tpep_pickup_datetime | The date and time when the meter was engaged. |
| tpep_dropoff_datetime | The date and time when the meter was disengaged. |
| Passenger_count | The number of passengers in the vehicle. This is a driver-entered value. |
| Trip_distance | The elapsed trip distance in miles reported by the taximeter. |
| Pickup_longitude | Longitude where the meter was engaged. |
| Pickup_latitude | Latitude where the meter was engaged. |
| RateCodeID | The final rate code in effect at the end of the trip.<br>• 1: Standard rate<br>• 2: JFK<br>• 3: Newark<br>• 4: Nassau or Westchester<br>• 5: Negotiated fare<br>• 6: Group ride |
| Store_and_fwd_flag | "This flag indicates whether the trip record was held in vehicle memory before sending to the vendor,aka "store and forward", because the vehicle did not have a connection to the server. "<br>• Y: store and forward trip<br>• N: not a store and forward trip |
| Dropoff_longitude | Longitude where the meter was disengaged. |
| Dropoff_latitude | Latitude where the meter was disengaged. |
| Payment_type | A numeric code signifying how the passenger paid for the trip.<br>• 1: Credit card<br>• 2: Cash<br>• 3: No charge<br>• 4: Dispute<br>• 5: Unknown<br>• 6: Voided trip |
| Fare_amount | The time-and-distance fare calculated by the meter. |
| Extra | "Miscellaneous extras and surcharges. Currently, this only includes the 0.50 *and* 1 rush hour and overnight charges." |
| MTA_tax | $0.50 MTA tax that is automatically triggered based on the metered rate in use. |
| Improvement_surcharge | $0.30 improvement surcharge assessed trips at the flag drop. The improvement surcharge began being levied in 2015. |
| Tip_amount | Tip amount This field is automatically populated for credit card tips. Cash tips are not included. |
| Tolls_amount | Total amount of all tolls paid in trip. |
| Total_amount | The total amount charged to passengers. Does not include cash tips. |



(a) Pickup distribution (b) Dropoff distribution

**Fig. 2.** Different distance measures used in this study.

did both temporal and spatial preprocessing and outlier removal for supporting the learning task more effectively.

### *4.1. Feature extraction*

Needless to say, the two most important geospatial features are distance, and direction. we use two different measures of distances between pickup and drop-off coordinates namely Haversine and Manhattan distance. Manhattan distance can be found as $|\Delta T| + |\Delta G|$ where $\Delta T = |\text{lat}_1 - \text{lat}_2|$ and $\Delta G = |\text{lng}_1 - \text{lng}_2|$. This distance is also called block distance or $L_1$-norm distance. Furthermore, due to the sphere shape of earth, for traveling between two points, we are not traversing in a straight line, but rather a curve. To consider that curvature distance (which is different than

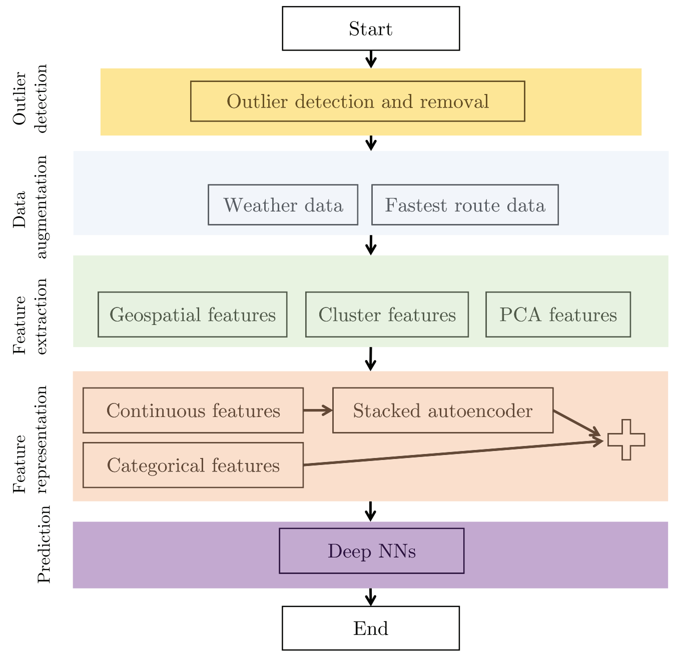


**Fig. 3.** This study's framework.

euclidean distance), Haversine distance is calculated as Eq. (1). Fig. 4 shows the difference between two distance measures.

$$2r\arcsin\left(\sqrt{\sin^2\left(\frac{\Delta T}{2}\right)+\cos(\text{lat}_1)\cos(\text{lat}_2)\sin^2\left(\frac{\Delta G}{2}\right)}\right) \quad (1)$$

In addition, bearing can be defined as direction or an angle, between the north-south line of earth (or meridian) and the line connecting the target and the reference point while heading is an angle or direction where you are currently navigating in. This means in order to reach a particular destination you need to adjust your heading direction with the bearing. Generally, a *compass* is an instrument, which provides the direction information for navigation. Particularly, current heading will vary as we follow a circle path, — final heading will be different from the initial heading by varying degrees according to latitude and distance, and it can be found in Eq. (2). Then, a bearing can be referred to an angle, measured clockwise from the north direction and can be calculated as

$$\begin{aligned}&\text{atan2}[\cos(\text{lat}_2)\sin(\Delta G), \cos(\text{lat}_1)\sin(\text{lat}_2)\\&\quad-\cos(\text{lat}_2)\sin(\text{lat}_1)\cos(\Delta G)]\end{aligned} \quad (2)$$

As we mentioned in our framework, for feature extraction we have used principal component analysis (PCA). The main idea of PCA is to reduce the data set dimension in which there are a large number of interrelated variables exist, while retaining as much as possible of the variation present in the data set. This reduction is achieved by transforming to a new set of variables, the principal components, which are uncorrelated, and which are ordered so that the first few retain most of the variation present in the original feature set. However, in this study, we use PCA to extract features related to latitude and longitude to improve our regressor's learnability. This works as projection of latitudes and longitudes on the axis which has highest variance. It has been proven that PCA features can improve neural networks and decision tree splitting (Nasution, Sitompul, & Ramli, 2018). Besides PCA, we have implemented a customized version of KMeans algorithm which initialize centroids by probabilities that maximize expected centroids' distance and uses Manhattan distance as loss function. Psuedocode of the clustering algorithm is given below, where `Distance`$(e_i, c_j)$ is the Manhattan distance between point $e_i$ and centroid $c_i$. Using these clusters, we can segmentize the city into different zones and find intra and inter cluster speeds, distances and other geospatial features. Furthermore, we calculated the speed or velocity as $v = \frac{\Delta d}{\Delta t}$ where $d$ is the distance calculated using Manhttan and Haversine formulas, and $t$ is the fastest route total travel time which has been found using OSRM fastest route API.

**Algorithm 1** K-Means++ algorithm.

```
Procedure KMeans++($E, k, MaxIter$)
 1: take one center $c_1$, chosen uniformly at random for $E$
 2: while $kk \leq k$ do
 3:     Take a new center $c_i$, choosing $e \in E$ with probability $\frac{D(e)^2}{\sum_{e\in E} D(e)^2}$
 4: for $e_i \in E$ do
 5:     $l(e_i) \leftarrow \text{argmin Distance}(e_i, c_j)\,\forall j \in \{1, 2, \cdots, k\}$
 6: $changed \leftarrow$ False
 7: $iter \leftarrow 0$
 8: while $changed =$ True and $iter \leq MaxIter$ do
 9:     for $c_i \in C$ do
10:         UpdateCluster($c_i$)
11:     for $e_i \in E$ do
12:         $minDist \leftarrow \text{argmin Distance}(e_i, c_j)\,\forall j \in \{1, 2, \cdots, k\}$
13:         if $minDist \neq l(e_i)$ then
14:             $l(e_i) \leftarrow minDist$
15:             $changed =$ True
16:     $iter++$
return $C, L$
```

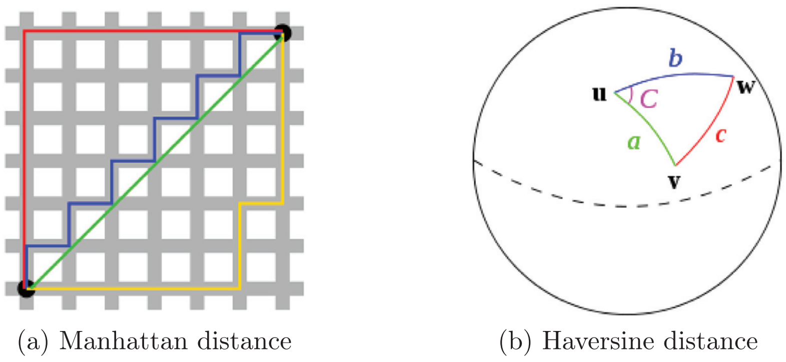


**Fig. 4.** Different distance measures used in this study.

## 4.2. Feature representation

Recent advances in machine learning, with data being extensively collected from a wide variety of resources, have increased interest in using such methods to uncover hidden patterns in these burgeoning datasets. Specially, recent advances and developments in multi-layer DNN designs combined with emerging of effective methods to train these networks have riased the opportunity to apply DNNs for novel applications which can range from autonomous vehicles to speech recognition. DNNs are multiple-layer deep architectures which can extract features in data and discover important latent or hidden structure. In our specific problem, knowing that the factors contributing to congestion, queuing delay and traffic flow originated from the vague interaction of complex features, DNNs provide powerful and new methodologies to learn how these features interact.

A traditional algorithm for training neural network for long time has been back propagation (Widrow & Lehr, 1990). More specifically, this algorithm requires the labeled data which has the form of ($x_i$, $t_i$), where the $x_i$ are the inputs and the $t_i$ are the targets. Neural networks find hypothesis or output value $h_\theta(x_i)$. After computing this hypothesis $h_\theta(x_i)$ we compare it with ground truth $t_i$ and the difference between ground truth and inferred hypothesis $h_\theta(x_i) - t_i$ is taken as model's empirical error estimate. This residual error is being *propagated backward* in the pipeline to tune and optimize the weights in order to minimize the difference $h_\theta(x_i) - t_i$ (or other measures such as MSE).

Back propagation algorithm has various impotence which essentially limits its usage to train DNNs. This algorithm does not align well with deep architectures having numerous hidden layers and the algorithm converges to poor local minima when weight initialization is done using random numbers. Another weakness of this algorithm is the requirement for labeled data where most of the data nowadays is without label (such as images, texts, and etc). The groundbreaking work of Hinton and his colleagues in 2006 (Hinton, Osindero, & Teh, 2006) has solved the above mentioned issues. Their breakthrough contribution was to show that greedy unsupervised layer-wise training of DNNs was effective to train these networks and using the backpropogation algorithm by substituting random weights with more robust concept.

The traditional autoencoder is an ANN which reproduces its input. It mostly work like compression algorithms. To be more precise, an AE takes an input vector $x \in \mathbb{R}^d$ and maps it to a latent or equivalently encoded representation $y \in \mathbb{R}^d$ by applying a deterministic mapping $y = f_\theta(x) = s(Wx + b)$, with parameters of $\theta = \{W, b\}$. The weight matrix $W$ has $d' \times d$ dimension, $b$ is a bias term and $s$ is the activation function, which usually is considered to be a sigmoid function $s(x) = \frac{1}{1+e^{-x}}$. The encoded representation $y$ can be reconstructed to vector $z \in [0, 1]^d$, where $z = g_{\theta'}(y) = s(W'y + b')$. Hence, each training sample $x(i)$ is represented to its related $y(i)$ and a reconstruction $z(i)$ in which we generally should have $y(i) \approx z(i)$. The idea is that autoencoder is constructed in a way where mapping $x(i) \rightarrow y(i)$ divulges essential structure in the input vector $x(i)$ that otherwise is not apparent. For instance, if input has more units than autoencoder, it must find a representation that compresses the input such a way that it can be reconstructed efficiently.

The parameters $\theta$ and $\theta'$ should minimize the average reconstruction error given as below:

$$R_\lambda(f_\lambda, D_n) = \frac{1}{n}\sum_{i=1}^{n} L\big(x(i), z(i)\big) = \frac{1}{n}\sum_{i=1}^{n} L\big(x(i), g_{\theta'}(f_\theta(x(i))\big) \quad (3)$$

To train the AE we need to find the parameters such that $\theta^*, \theta'^* = arg\min_{\theta,\theta'} R_\lambda(f_\lambda, D_n)$ Here, $L$ is the empirical loss function such as the one called squared error $L(x, z) = |x - z|_2^2$. Generally, this method solve learning problem as an optimization problem using empirical risk minimization (ERM) (Bengio, Courville, & Vincent, 2013). Empirical risk $\hat{R}(f_\theta, D_n) = \sum_{i=1}^{n} L\big(f_\theta(x(i)), z(i)\big)$ is generally the amount of deviation from the ground truth. In some cases, it might be crucial to set preferred values for some weights or parameters to avoid overfitting (Dietterich, 1995). Therefore, one can define a regularized empirical risk, in which regularization forces some level of sparsity on the derived encoding. Let's have a training set $D_n = \{x(1), x(2), \ldots, x(n)\}$. Then the regularized empirical risk is defined as:

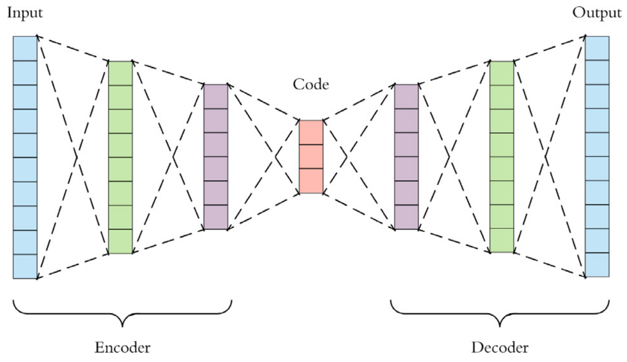


**Fig. 5.** Deep stacked autoencoder representation.

$$\hat{R}_\lambda(f_\lambda, D_n) = R_\lambda(f_\lambda, D_n) + \lambda\Omega(\theta) \quad (4)$$

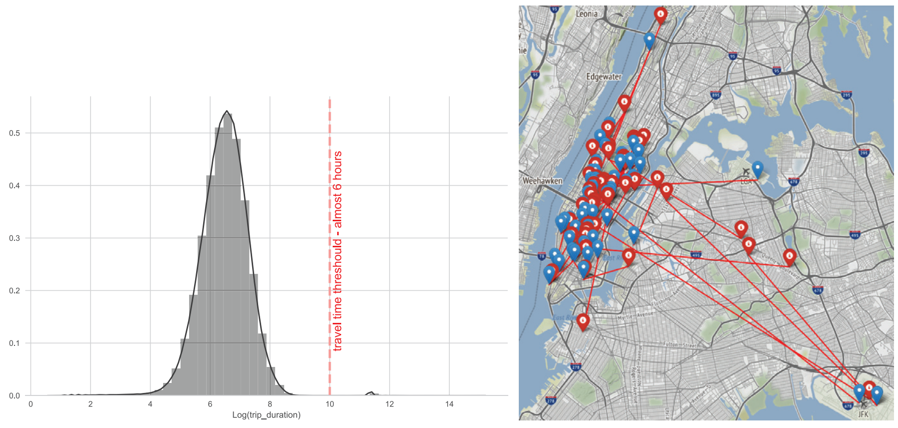
In Eq. (4) $\Omega$ penalizes specific values of the parameters (mostly large ones) and $\lambda \geq 0$ controls the regularization amount. Regularizers, in general, execute two basic functions: enforcing certain properties on the weights, and avoiding autoencoder to learn the identity function. As an instance, using the manhattan distance ($L_1$) as a regularizer, imposes sparsity on $x$ vector. This is because in order to minimize $|x|_1$, some of the $x_i$ must be zero ($|x|$ is always positive) which will imposed matrix sparsity. On the other side, $L_2$-norm influence the weights to tend to smaller values. The $p$-norm is defined to be $|x|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{\frac{1}{p}}$.

After having the autoencoders defined, now it is time to define stacked autoencoders (SAEs). SAEs are the collection of single-level autoencoders; so the SAEs have generally a deep architecture (Bengio et al., 2013). SAE deploys the above stated autoencoders to create a deep architecture (Bengio, Lamblin, Popovici, & Larochelle, 2007). Until very recently, deep architectures were thought to be too difficult to train their usage was very limited, although they could be more expressive and as such, could extract more sophisticated patterns from data. A schematic view of SAE is shown in Fig. 5.

## 5. Results and discussion

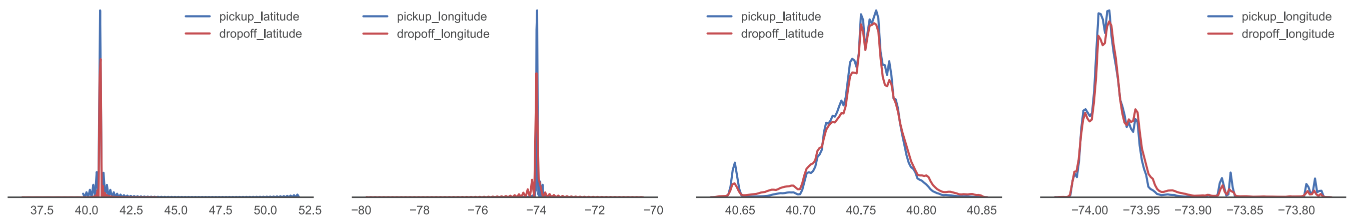
Our integrated framework starts with pre-processing and exploratory data analysis to capture nuances of the features and their relations and magnitude. As shows in density plot (Fig. 6a), trip duration follows a log-normal distribution. Most of the trips are $e^4$ s or 1 min to $e^8$ s (60 min) and probably are taken inside Manhattan or in New York area only. Few trips have very large duration, like 350,000 s which is 100 h (which is as long as a ride from NYC to SF or Alaska and it isn't an intercity taxi). At first, we focused on those trips that their total travel time take longer than $e^{10}$ s ($\tilde{6}$ h). As we can see in Fig. 6b these trips are all made in Manhattan area and seems reasonable to be considered as temporal outliers and being removed.

Moreover, we can see a density plot for pickup and drop-off latitude and longitudes in Fig. 7. It is clear that pickup and drop-off latitude are centered around 40 to 41, and longitude are



(a) Travel time distribution (b) Temporal outliers

**Fig. 6.** Trip distribution.



(a) Raw lat. dist. (b) Raw long. dist. (c) Capped lat. dist. (d) Capped long. dist.

**Fig. 7.** Distribution of raw pickup and drop off latitudes and longitudes.

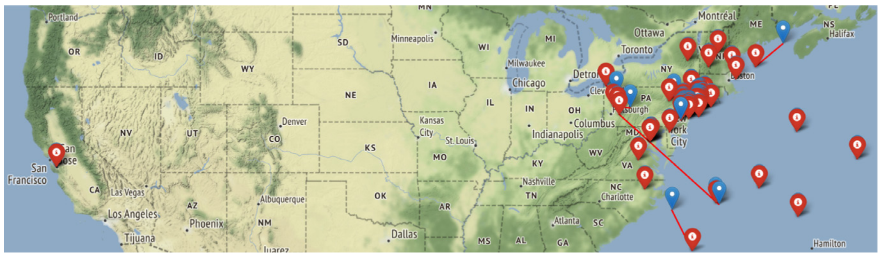


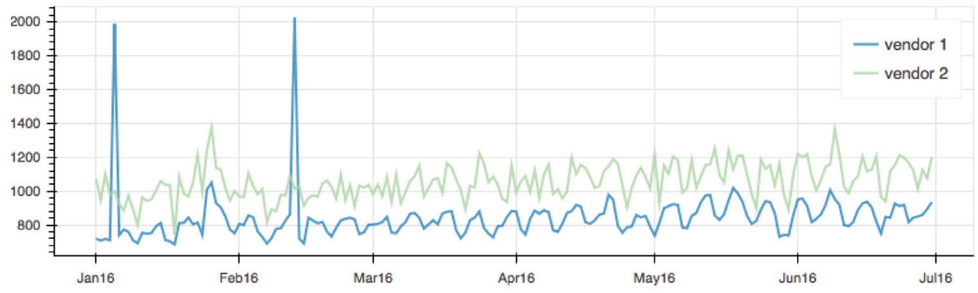
**Fig. 8.** Geographical outliers.

situated around −74 to −73, respectively. Trips which are very far from each other like latitude 32 to latitude 44, are not reasonable, and have affected this plot such that it is coming off as a spike (Fig. 7a and b). To remove some of these outliers (because we are analyzing only the Manhattan area), we remove those rides which their pickup latitude and longitude and their drop-off latitude and longitude are outside Manhattan area. These geo-outliers are shown in Fig. 8. After removing such outliers, latitude and longitude distribution are shown in Fig. 7c and d.

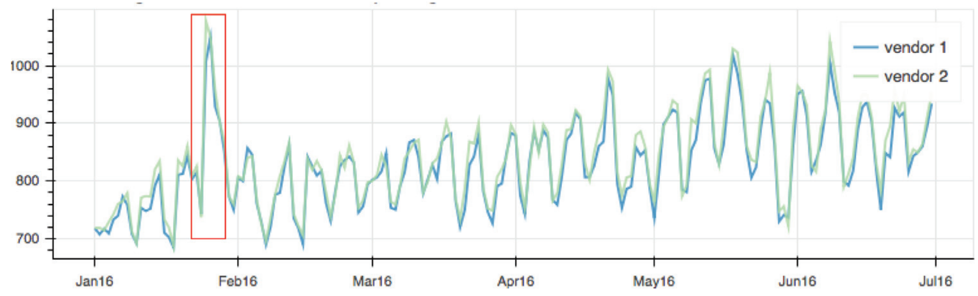
On the other hand, as previously explained in data dictionary, we have two vendors covering the Manhattan ride requests. From Fig. 9a it is clear that vendor 1 is taking more time than vendor 2 on all the days in our planning horizon. This might fire a question of why? The difference between the average time taken by vendor 1 is 250 s more than vendor 2. We have analyzed the data and observed that most of the very large trip durations reported are for vendor 1. However, after temporal outliers removal we can see that both vendors are taking almost the same amount of time as shown in Fig. 9b. Furthermore, as we can see, without removing temporal outliers we were not able to capture the long travel times in late January due to heavy snow, which is obvious from Fig. 9c. Fig. 9c shows the minimum and maximum temperature and snowfall which can be used to justify some events like the one in Fig. 9b. This shows the importance of both data cleaning and augmentation using weather data.

Besides that, we append PCA features to our feature space. We have used latitudes and longitudes (both for pickups and dropoffs), and fitted a PCA model to extract 2 principal components. We do not want to reduce dimension of input space using PCA, but to explain coordinates in an axis which describes most of the variance. Results of this transformation is shown in Fig. 10b. As we can see, PCA just mirrored the observation on $y$ axis. Besides that, 59.85% variance is explained using first principal component while second component explains 40.14% of variance. Other than original latitude and longitudes, we will consider these features in our feature space as well.

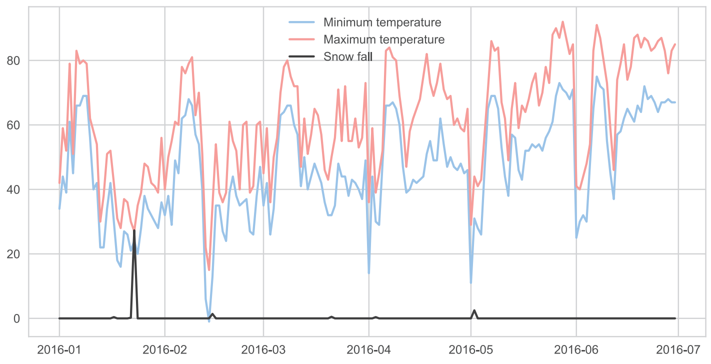
Moreover, as mentioned before, we will use our clustering algorithm to partition the city into regions or zones. This will help us



(a) Raw travel times by each vendor

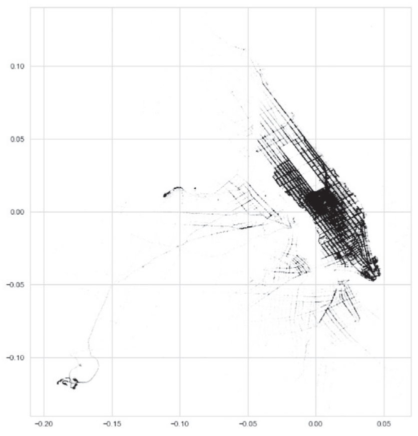


(b) Processed travel times by each vendor

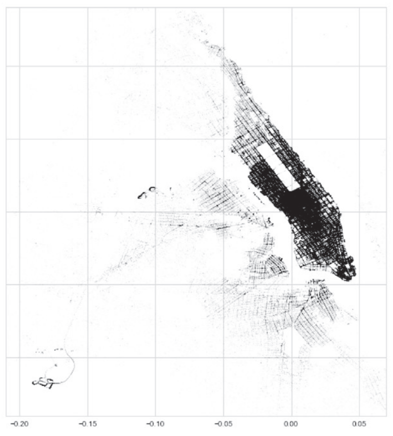


(c) Weather dynamics

**Fig. 9.** Travel time distribution by each vendor before and after temporal outlier removal.



(a) PCA transformation pickup



(b) PCA transformation dropoff

**Fig. 10.** PCA transformation.

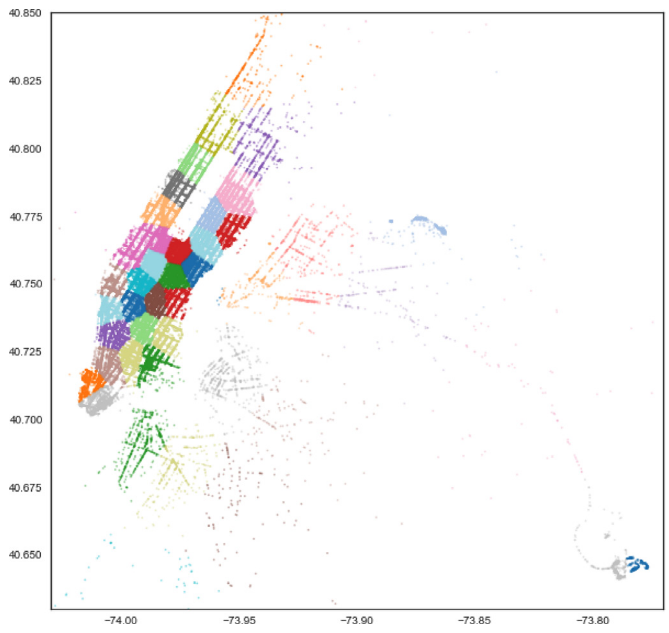


**Fig. 11.** NY trip zones.

find characteristic and dynamic of each zone such as distance between zones, distance between a request and zone center, within and between zone speeds, and etc. to include them in our feature space. For this purpose, we only have used pickup and drop-off latitude and longitude to perform clustering and define zones. To find the best number of clusters, we have used elbow method which is a popular method for finding $k$ in clustering algorithms. Recall that, the basic idea behind cluster partitioning methods, such as k-means clustering, is to define clusters such that the total intra-cluster variation (known as total within-cluster variation or total within-cluster sum of square) is minimized. The total within-cluster sum of square measures the compactness of the clustering and we want it to be as small as possible to form a good cluster, and is defined as below.

$$D_r = \sum_{r=1}^{k} \sum_{i=1}^{n_r-1} \sum_{j=i}^{n_r} \frac{1}{n_r} |d_i - d_j|_2 \tag{5}$$

Where $k$ is the number of clusters, $n_r$ is the number of points in cluster $r$ and $D_r$ is the sum of all distances in cluster $r$. We used this method for defining best number of cluster on a set of predefined $k$. The results of clustering algorithm with 50 cluster centroids is shown in Fig. 11. For extracting more feature dynamics we can increase number of clusters.

So far, we have increased feature space using our feature extraction framework from 11 feature to 44, by using the methods explained in Section 4.1, such as weather and route augmentation, temporal features such as weekend/weekdays, hour, time of day, and etc., spatial features such as distance and direction between points and their corresponding cluster, distance between and within clusters, average speed within and between clusters, bearing and direction of the travel, and PCA features. Out of these 44 features, 33 of them are continuous and rest are categorical. We learn these 33 features using a deep stacked autoencoder for which the architecture is shown in Fig. 12. As it can be seen, we have a neural network with 7 hidden layers which we train to learn the representative features in lower dimensions. For tuning the hyperparamters, we have tried network with 3, 5, and 7 layers with neurons ranging from $\frac{1}{3}$ to $\frac{2}{3}$ of the previous layer's number of neurons (As we wanted to achieve lower dimension in latent

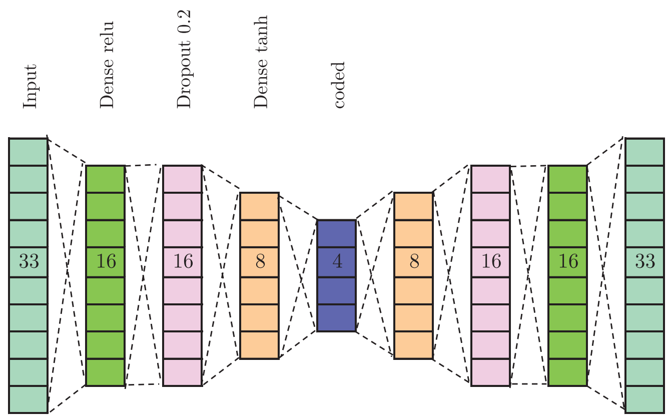


**Fig. 12.** Deep Stacked autoencoder architecture.

representation). Besides that, We also use dropout layers, that allows the model to drop some of network nodes at random (20%), to prevent overfitting and improve generalization. As represented in Table 12, we start with 33 continuous features and through each layer we will get 16, 8, 4, 16, 16, 8, 33 features. 4 encoded features are provided for the regression model afterwards.

Dropout is a regularization technique for neural networks, which was proposed by Srivastava, Hinton, Krizhevsky, Sutskever, and Salakhutdinov (2014). In this method, neurons are randomly selected to be ignored during training. They are randomly "dropped-out". This means neuron's contribution to the activation function of network's downstream is temporally block on the forward pass and in the backward pass updates are not applied to those neurons. As NN learns through epochs, weights of neurons are well tuned for specific features.

We can imagine that neurons which are not dropped out should compensate for neurons which are randomly ignored during training, and they should handle the representation required to make predictions for the missing neurons. It is believed that this will result in multiple independent internal representations being learned by the network. The effect of this mechanism is that network becomes less sensitive to specific weights. This will result in a network that is robust to overfitting and capable of better generalization.

After getting the 4 encoded features, we will add them with 11 categorical features and this 15 features will represent our feature space for travel time prediction task. Then, we trained a deep multi-layer perceptron model with 3 hidden layers and (10,20,10) neurons in each layer, and (relu, relu, relu) activation functions in 100 epochs. These hyperparameters such as several layers, and neurons are tuned using a 5-fold cross validation on training data and implementing grid search on set of predefined parameters. We have used a layer-wise training to prevent the vanishing gradient so the model is trained using stochastic gradient decent (SGD) algorithm. Furthermore, we compared the result of applying a fine-tuned MLP structure on raw data (with 8 features) to evaluate the performance of feature learning proposed in this study. Results are provided in Fig. 13.

From the graph above, it can be understood that our proposed algorithm works on average 250 s better than a deep MLP on initial feature space. Moreover, we have analyzed the predictive pattern during whole planning horizon from January to June 2016. Fig. 14 shows that our algorithm recognizes the pattern of travel time pretty well, and is capable of capturing dynamics of travel time. However, sometimes that we have very large travel times (due to rare events such as heavy snow in late January) our algorithm is underestimating them.

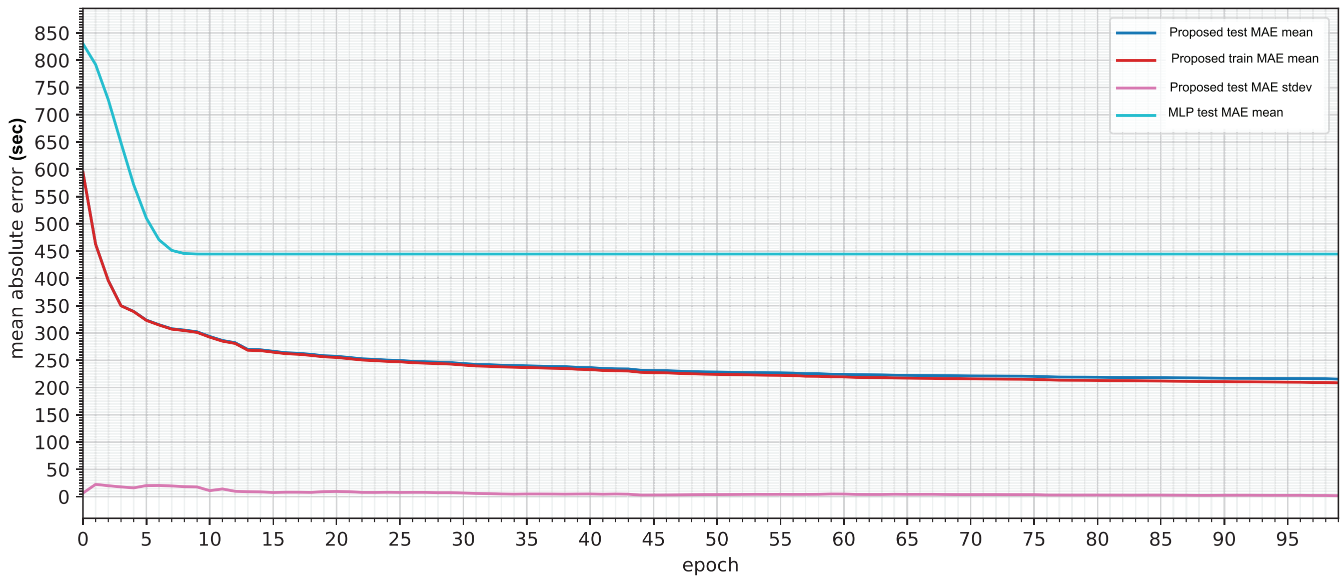


**Fig. 13.** Comparison of proposed algorithm with deep MLP.

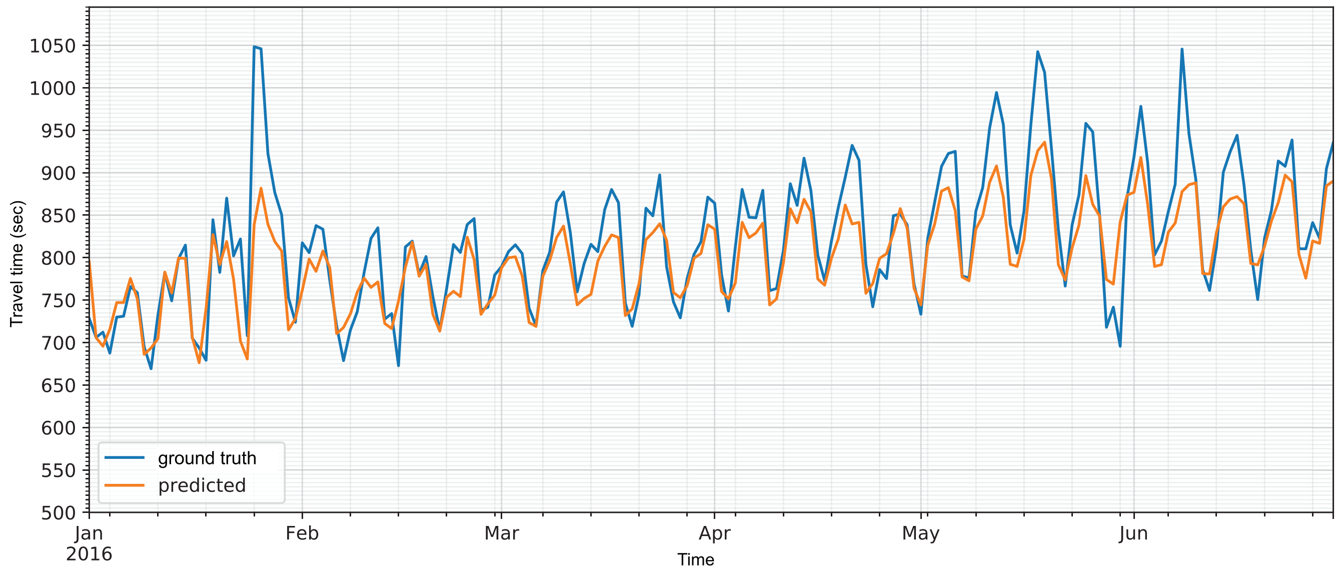


**Fig. 14.** Pattern recognition capability of proposed algorithm.

## 6. Conclusion

Predicting travel time has various application in different fields for different purposes. At one side, traffic managers are interested in travel time prediction as it is a fundamental element in traffic system operation, where its result can help traffic planners to adjust traffic flow by means of time dependent rules. Besides that, from traveller's point of view, accurate estimation of travel time can help travelers to choose best route prior to trip or even en-route. In this study, we presented an algorithm which is completely data-driven and needs not special technologies (such as ANPR, satellite, or etc.) and makes it very convenient and cost-effective for predicting travel times.

In this study, we developed an algorithm for predicting travel times for Manhattan, New York during January 2016 to June 2016. To do so, we took a multi-step approach starting by both temporal and spatial outliers removal. Then, we augmented TLC dataset by incorporating external weather and fastest route data which was publicly available. Moreover, extensive feature engineering techniques such as geospatial features analysis, statistical learning methods (PCA), and unsupervised learning algorithm (K-Means) have been applied to boost the feature space. After having a decent feature set, we used a deep stacked autoencoder to represent features in lower dimension. This will improve predictive accuracy as well as decrease overfitting chances and make the learner more robust. Finally, a deep multi-layer perceptron has been trained to predict the travel times. As we saw, the performance of the proposed algorithm (MAE $\approx$200 s) is almost 4 min on average better than applying the deep neural network to the initial training data. Moreover, we have noticed that representation learning using stacked autoencoders and reducing data dimension make both our feature set and learner model robust to overfitting. For testing our predictive accuracy, we have used a 5-fold cross validation to test the generalization of our predictive model. Moreover, our proposed algorithm is capable of capturing the general dynamics of the traffic, however it failed when we had heavy snow or some other rare event which impact travel times significantly.

To overcome the problems mentioned and for future works, we can implement different deep architectures to the dataset, and we can have different representational learning algorithms such as variational autoencoders and denoising encoders tested to compare their performance and effectiveness. Moreover, we can break down a route into its links and predict link travel time which might be a good direction to move to improve predictive accuracy. Then, we can add all the link travel times to get the total travel time of a route.

## Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.eswa.2019.112864.

## Credit authorship contribution statement

**Mohammad Abdollahi:** Conceptualization, Methodology, Software, Formal analysis, Writing - original draft, Writing - review & editing, Project administration, Visualization. **Tannaz Khaleghi:** Methodology, Software, Data curation, Writing - original draft, Writing - review & editing, Visualization. **Kai Yang:** Resources, Supervision, Validation, Writing - review & editing.

## References

Abdollahi, M., Arvan, M., Omidvar, A., & Ameri, F. (2014). A simulation optimization approach to apply value at risk analysis on the inventory routing problem with backlogged demand. *International Journal of Industrial Engineering Computations, 5*(4), 603–620.

Allström, A., Ekström, J., Gundlegård, D., Ringdahl, R., Rydergren, C., Bayen, A. M., & Patire, A. D. (2016). Hybrid approach for short-term traffic state and travel time prediction on highways. *Transportation Research Record, 2554*(1), 60–68.

Azadeh, A., Jafari-Marandi, R., Abdollahi, M., & Roudi, E. (2017). A novel benchmark methodology for estimating industrial electricity demand considering unsteady socio-economic conditions. *International Journal of Business Performance Management, 18*(2), 196–215.

Azadeh, A., Taghipour, M., Asadzadeh, S., & Abdollahi, M. (2014). Artificial immune simulation for improved forecasting of electricity consumption with random variations. *International Journal of Electrical Power & Energy Systems, 55*, 205–224.

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 35*(8), 1798–1828.

Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2007). Greedy layer-wise training of deep networks. In *Advances in neural information processing systems* (pp. 153–160).

Billings, D., & Yang, J.-S. (2006). Application of the arima models to urban roadway travel time prediction-a case study. In *Systems, man and cybernetics, 2006. SMC'06. IEEE international conference on: 3* (pp. 2529–2534). IEEE.

Davies, J., Fortuna, B., Duke, A., Clarke, S. S., & Rupnik, J. (2015). Travel time prediction on highways. In International conference on computer and information technology.

Dietterich, T. (1995). Overfitting and undercomputing in machine learning. *ACM Computing Surveys (CSUR), 27*(3), 326–327.

Ester, M., Frommelt, A., Kriegel, H.-P., & Sander, J. (2000). Spatial data mining: Database primitives, algorithms and efficient dbms support. *Data Mining and Knowledge Discovery, 4*(2–3), 193–216.

Guin, A. (2006). Travel time prediction using a seasonal autoregressive integrated moving average time series model. In *Intelligent transportation systems conference, 2006. ITSC'06. IEEE* (pp. 493–498). IEEE.

Guo, J., Huang, W., & Williams, B. M. (2014). Adaptive kalman filter approach for stochastic short-term traffic flow rate prediction and uncertainty quantification. *Transportation Research Part C: Emerging Technologies, 43*, 50–64.

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

Huang, W., Song, G., Hong, H., & Xie, K. (2014). Deep architecture for traffic flow prediction: Deep belief networks with multitask learning. *IEEE Transactions on Intelligent Transportation Systems, 15*(5), 2191–2201.

Jiang, X., & Adeli, H. (2005). Dynamic wavelet neural network model for traffic flow forecasting. *Journal of Transportation Engineering, 131*(10), 771–779.

Khaleghi, T., Abdollahi, M., & Murat, A. (2019). Machine learning and simulation/optimization approaches to improve surgical services in healthcare. In *Analytics, operations, and strategic decision making in the public sector* (pp. 138–165). IGI Global.

Khaleghi, T., Murat, A., & Neemuchwala, H. (2016). Use of simulation in managing reusable medical equipment inventory in surgical services. In *Proceedings of the summer computer simulation conference* (p. 39). Society for Computer Simulation International.

Kumar, S. V., & Vanajakshi, L. (2015). Short-term traffic flow prediction using seasonal arima model with limited input data. *European Transport Research Review, 7*(3), 21.

Lopez-Garcia, P., Onieva, E., Osaba, E., Masegosa, A. D., & Perallos, A. (2016). A hybrid method for short-term traffic congestion forecasting using genetic algorithms and cross entropy. *IEEE Transactions on Intelligent Transportation Systems, 17*(2), 557–569.

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

Mennis, J., & Guo, D. (2009). Spatial data mining and geographic knowledge discoveryan introduction. *Computers, Environment and Urban Systems, 33*(6), 403–408.

Mkanta, W. N., Chumbler, N. R., Yang, K., Saigal, R., & Abdollahi, M. (2016). Cost and predictors of hospitalizations for ambulatory care-sensitive conditions among medicaid enrollees in comprehensive managed care plans. *Health Services Research and Managerial Epidemiology, 3*. 2333392816670301

Mkanta, W. N., Chumbler, N. R., Yang, K., Saigal, R., Abdollahi, M., Mejia de Grubb, M. C., & Ezekekwu, E. U. (2017). An examination of the likelihood of home discharge after general hospitalizations among medicaid recipients. *INQUIRY: The Journal of Health Care Organization, Provision, and Financing, 54*. 0046958017711783

Nasution, M., Sitompul, O., & Ramli, M. (2018). Pca based feature reduction to improve the accuracy of decision tree c4. 5 classification. In *Journal of physics: Conference series: 978* (p. 012058). IOP Publishing.

Oh, S., Byon, Y.-J., Jang, K., & Yeo, H. (2018). Short-term travel-time prediction on highway: A review on model-based approach. *KSCE Journal of Civil Engineering, 22*(1), 298–310.

Okutani, I., & Stephanedes, Y. J. (1984). Dynamic prediction of traffic volume through kalman filtering theory. *Transportation Research Part B: Methodological, 18*(1), 1–11.

Petticrew, M., & Roberts, H. (2008). *Systematic reviews in the social sciences: A practical guide*. John Wiley & Sons.

Polson, N., & Sokolov, V. (2016). Deep learning predictors for traffic flows. arXiv:1604.04527v1.

Sharma, A. (2006). *Spatial data mining for drought monitoring: An approach using temporal ndvi and rainfall relationship*. ITC.

Shen, L. (2008). Freeway travel time estimation and prediction using dynamic neural networks.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research, 15*(1), 1929–1958.

Sun, H., Liu, H. X., Xiao, H., He, R. R., & Ran, B. (2003). Use of local linear regression model for short-term traffic forecasting. *Transportation Research Record, 1836*(1), 143–150.

Tan, H., Xuan, X., Wu, Y., Zhong, Z., & Ran, B. (2016). A comparison of traffic flow prediction methods based on dbn. In *CICTP 2016* (pp. 273–283).

Turner, S. M., Eisele, W. L., Benz, R. J., & Holdener, D. J. (1998). Travel time data collection handbook. *Technical Report*.

Van Lint, J. (2004). *Reliable travel time prediction for freeways*. Netherlands TRAIL Research School.

Wang, J., & Shi, Q. (2013). Short-term traffic speed forecasting hybrid model based on chaos–wavelet analysis-support vector machine theory. *Transportation Research Part C: Emerging Technologies, 27*, 219–232.

Widrow, B., & Lehr, M. A. (1990). 30 years of adaptive neural networks: Perceptron, madaline, and backpropagation. *Proceedings of the IEEE, 78*(9), 1415–1442.

Wu, Y., Tan, H., Peter, J., Shen, B., & Ran, B. (2015). Short-term traffic flow prediction based on multilinear analysis and k-nearest neighbor regression. In *CICTP 2015* (pp. 556–569).

Yang, H.-F., Dillon, T. S., & Chen, Y.-P. P. (2017). Optimized structure of the traffic flow forecasting model with a deep learning approach. *IEEE Transactions on Neural Networks and Learning Systems, 28*(10), 2371–2381.

Yang, M., Liu, Y., & You, Z. (2010). The reliability of travel time forecasting. *IEEE Transactions on Intelligent Transportation Systems, 11*(1), 162–171.

Zhang, J., Zheng, Y., & Qi, D. (2017). Deep spatio-temporal residual networks for citywide crowd flows prediction.. In *AAAI* (pp. 1655–1661).

Zhang, J., Zheng, Y., Qi, D., Li, R., & Yi, X. (2016). Dnn-based prediction model for spatio-temporal data. In *Proceedings of the 24th ACM sigspatial international conference on advances in geographic information systems* (p. 92). ACM.

Zhang, Y., & Haghani, A. (2015). A gradient boosting method to improve travel time prediction. *Transportation Research Part C: Emerging Technologies, 58*, 308–324.

Zheng, W., Lee, D.-H., & Shi, Q. (2006). Short-term freeway traffic flow prediction: Bayesian combined neural network approach. *Journal of Transportation Engineering, 132*(2), 114–121.