Vol.67: e24220670, 2024
https://doi.org/10.1590/1678-4324-2024220670
ISSN 1678-4324 Online Edition

BABT
Brazilian Archives of Biology and Technology

Article - Engineering, Technology and Techniques

H2O-Based Stochastic Gradient Descent Grid Search Using Ridge Regression Techniques for Traffic Flow Forecasting

Rajalakshmi Gurusamy[1*]
https://orcid.org/0000-0003-1722-373X

Siva Ranjani Seenivasan[2]
https://orcid.org/0000-0002-0958-1062

[1]Sethu Institute of Technology, Department of Information Technology, Tamilnadu, India; [2]Sethu Institute of Technology, Department of Computer Science and Engineering, Tamilnadu, India.

Editor-in-Chief: Paulo Vitor Farago
Associate Editor: Paulo Vitor Farago

Received: 31-Aug-2022; Accepted: 18-Dec-2023

*Correspondence: gurusamyrajalakshmi89@gmail.com (R.G.).

HIGHLIGHTS

- H2O's parallel grid SGD approach builds a series of DNN models in parallel.
- Ridge regression reduces multicollinearity and minimizes variance and bias.
- The SGD parallel grid returns optimal hyperparameters for DNN training.
- Parallel DNN models predict traffic flows with minimal error metrics.

Abstract: Efficient and exact traffic flow forecasting is critical for intelligent transportation systems. The number of model generations increases the computational complexity of deep neural network (DNN) models. Overfitting occurs when hyperparameters are used excessively to train neural network models, and this has a major influence on prediction accuracy. To address these limitations, this approach employed H2O grid-based stochastic gradient descent with ridge regression in deep neural network (GSGDRR-DNN). This technique efficiently distributes memory across several clusters, runs independently, and simultaneously creates numerous DNN models. To remove multicollinearity and achieve better computational efficiency and lower variance, GSGDRR-DNN utilizes stochastic gradient descent (SGD) with ridge regression in the H2O cluster. Finally, we evaluate the performance of the recommended GSGDRR-DNN approach against several DNN methods, including LSTM, Bi-LSTM, GRU, and current state-of-the-art methods. Additionally, the run-time performance of the parallel GSGDRR-DNN model was compared with the run-time performance of the sequential GSGDRR-DNN model. The suggested system has a minimum MSE, a minimum RMSE, a minimum MAE, a minimum RMSLE, and maximum R^2 values of 0.012, 0.108, 0.096, 0.015, and 0.99. This demonstrates that the GSGDRR-DNN model of traffic flows outperforms other state-of-the-art approaches in terms of prediction accuracy.

Keywords: Deep Neural Network (DNN); Ridge Regression; Grid Search; Stochastic Gradient Descent; Parallel Distributed Processing; Traffic flow forecasting.

INTRODUCTION

Due to the non-linearity of traffic data, accurate and well-organized traffic flow estimates are a difficult task in Intelligent Transportation System (ITS). Nonlinear traffic data gathered from a variety of sources, including radars, inductive loops, CCTV cameras, and other social networks. Congestion, accident prevention, navigation time savings, and navigation safety considerations are all aided by accurate traffic flow predictions. Both parametric and nonparametric statistical techniques fall into the category of forecasting methods. Traditional statistical parametric techniques (SPTs) like Kalman filtering, ARIMA (Auto Regressive Integrated Moving Average), and SVR (Support Vector Regression) are not ideal for traffic data non-linearity, although they are less computationally complex. SARIMA (Seasonal ARIMA) was contemplated by Williams BM and coauthors [1] and is used to estimate univariate traffic state data. Extreme values were difficult to forecast using this strategy.

Okutani I and coauthors [2] suggested Kalman Filtering that could not infer the stochastic features of the data. Castro-Neto M and coauthors [3] presented SVR for estimating short-term traffic flow from ordinal data. Emami A and coauthors [4] introduced kalman filter using Vissim microscopic traffic simulator to fully predict traffic flow when transport boreholes are reduced. However, it is not suitable for high perforation conveyance rate. The non-statistical parametric methods such as the K-nearest neighbor (KNN) model (Gong X and coauthors [5]) Artificial Neural Networks (ANN) (Zheng W and coauthors [6]; Zhong M and coauthors [7]; Dia H [8]; Yin H and coauthors [9]; Kumar K and coauthors [10]; Dougherty M [11]) and various deep learning techniques such as LSTM, GRU, SAE have been designed for linear and non-linear flow of traffic. However, these approaches have complex network structures that can accurately predict traffic flow and take more time to train neural network models.

Yu B and coauthors [12] suggested KNN as a multi-step predictive model for predicting short term traffic. Ghosh B and coauthors [13] to estimate real-time traffic flow at different crossings, researchers introduced a Multivariate Structural Time-series (MST) model. The usage of several unknown parameters and the stochastic character of traffic data have an impact on model training and forecast accuracy. These methods are computationally intensive and inefficient when generating various models for predicting short-term traffic. A data-driven approach has emerged to address the stochastic nature of traffic data (Zhang J and coauthors [14]). Recently, various types of non-statistical parametric deep learning (NPDL) methods have been introduced to predict traffic flow. LSTM-RNN (Long Short-Term Memory-Recurrent Neural Network), which memorize the features as a long-term dependency and accurately predict short-term traffic (Ma X and coauthors [15]; Zhao Z and coauthors [16]).

The GRU (Gated Recurrent Unit) (Fu R and coauthors [17]), SAE (Stacked Auto Encoder) (Lv Y and coauthors [18]) methods efficiently process the non-linear traffic flow data. The DNN (Krizhevsky A and coauthors [19]; Li P and coauthors [20]; Atrey K and coauthors [21]) have excellent performance in various fields. If the dataset is very large, the first major difficulty with non-parametric methods is the inability to process large amount of data in a timely and insightful manner. The second drawback is when the regression uses multiple predictors. As a result, errors can easily spread to the next layer, leading to overfitting or underfitting. In addition, it receives more training parameters and requires higher memory bandwidth to train the model, which makes the model more complex. Most researchers rely on back propagation to minimize model errors by computing involute derivatives (Annamalai M and coauthors [22]). To overcome these problems, this article suggested the GSGDRR method is a combination of four techniques. It includes grid search, the stochastic gradient descent (SGD) algorithm, the kernel ridge regression method, and the parallel distributed in-memory computations on DNN.

The main contributions of the GSGDRR-DNN model consist of the following:

1. The recommended approach can be exploited to expand real-time traffic data quickly. The sparkling water (H2O) is utilized to store and interpret real-time traffic data with exponential oscillations. The number of processors in an H2O cluster works in parallel, allowing for appropriate task distribution and traffic flow estimation.
2. On an H2O cluster, the model used the parallel grid SGD approach to construct a sequence of DNN models in parallel. The optimal combination of hyperparameters is returned by the SGD parallel grid, which is utilized to train DNNs and accurately anticipate traffic flow.
3. Using ridge regression, the suggested model decreases multicollinearity, overfitting, and achieves minimal variance and bias.

4. Evaluate the performance of our predictive model compared to non-parametric DNN methods such as LSTMs, Bi-LSTMs and GRUs. The finding shows that the GSGDRR-DNN model predicts traffic flow more accurately than other state-of-the-art deep learning approaches and has lower error metrics.
5. The runtime performance of the GSGDRR-DNN sequential model is further compared to the GSGDRR-DNN parallel model. The GSGDRR-DNN parallel model makes good use of memory and performs the fastest calculations to predict traffic flow.

The article's content is structured as follows: A review of the literature is described in Section II. The methodology is covered in Section III. Section IV presents the experimental findings and related discussion. Conclusions are presented in Section V.

LITERATURE REVIEW

Vlahogianni EI and coauthors [23] suggested an optimized static MLP (Multilayer Perceptron) using a genetic algorithm to predict the performance of univariate and multivariate traffic flow. When training a model that predicts time (univariate) and spatial (multivariate), the computational process is complex and time-consuming. With this method, is not known how much each independent variable is affected by the dependent variable. Our recommended model achieved minimal variance and bias by reducing the coefficients using ridge regression. It considerably reduces multicollinearity, overfitting, gradient descent issue and model involution. Chan KY and coauthors [24] presented hybrid neural networks to estimate short-term traffic flow, which combine classical exponential smoothing (ES) with the Levenberg-Marquardt (LM) technique. This method effectively uses ES to pre-processes time-series data, which is then fed into the LM algorithm as an input. It enhances the model's ability to generalize. This methodology is ineffective when dealing with non-linear and multivariate traffic flow data.

A fully automated procedure, the Dynamic k-NN procedure (DP k-NN) was modeled by Sun B and coauthors [25]. It is a self-adjusted parameter that anticipates and trains short-term traffic flow without relying on past models. Considering short-term traffic flow, our recommended model outperforms traditional parametric methods based on grid search in terms of accuracy. Ma X and coauthors [15]; Gurusamy R and coauthors [26] presented LSTM NN to effectively capture non-linear traffic. The contemplated model uses the parallel grid SGD method to effectively process and store non-linear traffic data. Tian Y and coauthors [27] suggested LSTM RNN improves accuracy by dynamically estimating the ideal delay using three multiplication units in the memory block. Our recommended designed approach achieved the highest predictive performance in terms of accuracy and stability.

Fu R and coauthors [17] investigated RNN based on deep learning methodologies such as LSTM and GRU to estimate short-term traffic flow. Our model obtained lower MAPE, MAE, MSE and RMSE values than existing state-of-the-art methods. Krizhevsky A and coauthors [19] presented deep convolutional neural network (CNN). The researcher trained this algorithm using ImageNet's 1.2 million images and divided it into 1,000 different classes. The model used the GPU (Graphics Processing Unit) to speed up the training process. The model used a dropout regularization technique to avoid overfitting. Various CNN models were suggested by Li P and coauthors [20] to extract distinct facial features from the face. The retrieved features were then used to train the model. Finally, Bayesian probability combines all of the characteristics to produce a precise forecast.

According to Lv Y and coauthors [18], the SAE DNN architecture is used to predict traffic flow. Jiang H and coauthors [28] presented numerous machine learning techniques. It includes Backpropagation Neural Networks (BP NN), Non-linear Exogenous Autoregressive NN (NARX), Radial Basis Function with Support Vector Machine with as Kernel Function (RBFSVM), Linear Function with Support Vector Machine with (LinSVM), and Multilinear Regression (MLR) to predict the short-term flow. Huang W and coauthors [29] for spark on YARN, Resilient Distributed Data (RDD) and a strip-oriented data model were presented. Because shuffling is the system's most time-consuming action, this architecture is not suited for joining heavy algorithms.

Nabavinejad SM and coauthors [30] innovated with a mnemonic approach that cannot fully utilize the available computing resources such as memory and CPU. Tang S and coauthors [31] suggested max-min fairness, LTRF (Long Term Resource Fairness) and H-LTRF (Hierarchical-LTRF). Using the LTRF method, there is a possibility of the starvation problem. To overcome the problem in LTRF problem, H-LTRF introduced it based on time-out techniques. Niu Z and coauthors [32] introduced FLEX, a Hadoop meta-scheduler that strikes a compromise between efficiency and fairness. Chen CP and coauthors [33] presented

various tools and techniques for big data. In the context of big data, Chen D [34] suggested using an enhanced RBF neural network to predict traffic flow. To estimate long-term traffic, Wang Z and coauthors [35] introduced the Attention Calibration Encoder-Decoder (ACE-D) model. Our model attained greater scalability, maximum memory efficiency, more expeditious computations, and accurate prediction in DNN.

MATERIAL AND METHODS

This article suggests that the GSGDRR method is a combination of four methods. This includes grid search, stochastic gradient descent (SGD) algorithms, kernel ridge regression methods, and in-memory distributed parallel computations in deep neural networks (DNNs). Ridge regression, SGD predicted by the grid, was accumulated to train the model in DNN to forecast traffic flow. For scalability, non-linear information stored in spark RDD, and for expeditious computation, the proposed GSGDRR-DNN model uses multiple H2O clusters with parallel distributed nodes.

DEEP NEURAL NETWORK (DNN)

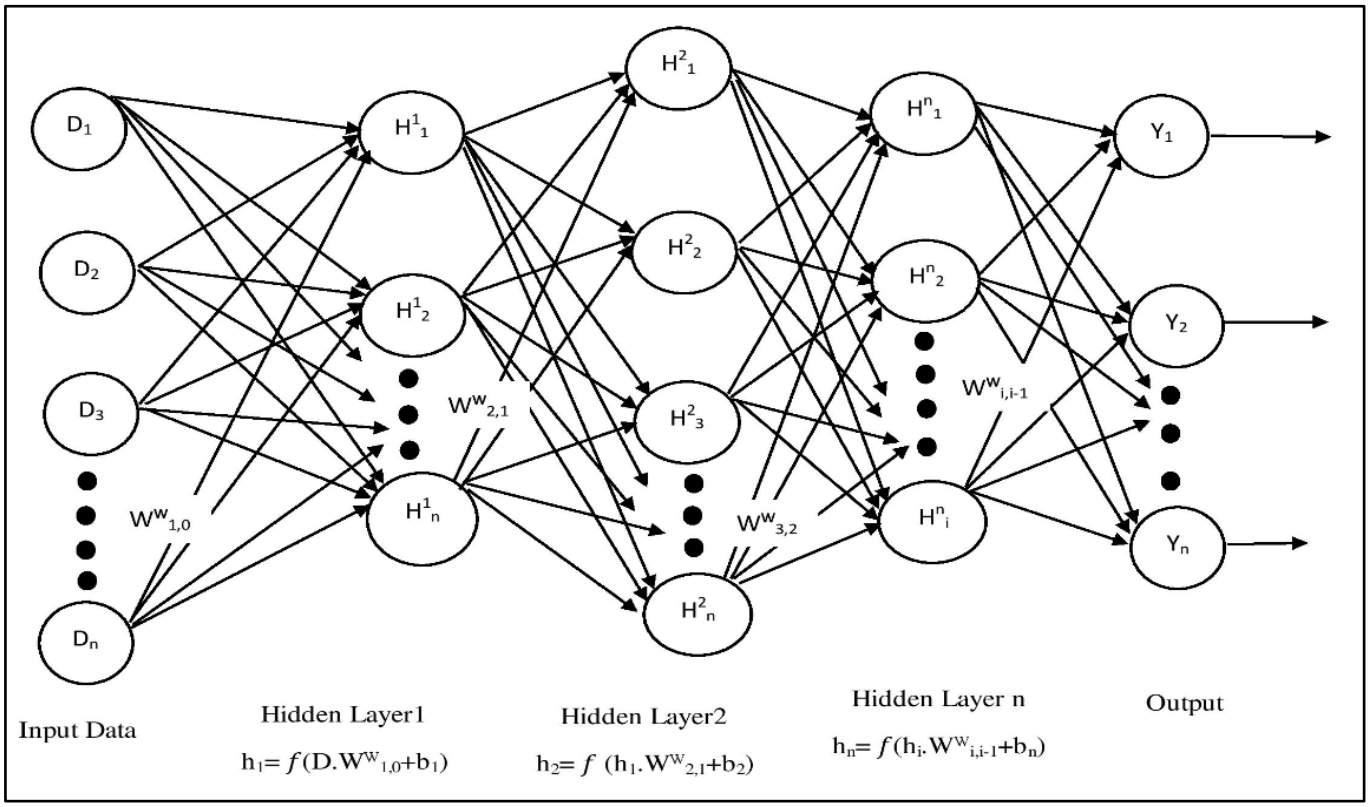


Figure 1. A DNN Architecture includes interconnected input, hidden, and output layers to deepen and improve model precision.

Figure 1 shows Deep Neural Network (DNN) consists of more than one hidden layer to engender a deeper model and amend the precision of the model. It consists of several interconnected neurons associated with input (D_1, D_2 …, D_i), hidden (h_1, h_2 …, h_i) and output units (Y_1, Y_2 …, Y_i). The weights ω and bias β defines the network output, denoted as:

$$Y_i = f(D_i\omega_i + \beta_i) \quad \textbf{(1)}$$

KERNAL RIDGE REGRESSION

The problem of multiple regressions is that the multiple independent variables are dependent on other several independent variables (multicollinearity). It considerably affects the model's performance and makes the model unstable. To eliminate multicollinearity first step is to find a relationship among the independent variables by using correlation heat map operations. It will perform on all nodes in the H2O frame. This method is Exploratory Data Analysis (EDA) for analyzing the correlations and relationships between variables in training data (TD). A high correlation feature has a high linear dependency and has the same effect on the dependent variable. Ergo, we extract the features that have a high correlation, and it calculates as follows:

$$Corr(TD) = \frac{\sum_{n=1}^{m}(D_n - mean(D))(Y_n - mean(Y))}{\sqrt{\sum_{n=1}^{m}(D_n - mean(D))^2 \sum_{n=1}^{m}(Y_n - mean(Y))^2}} \quad \textbf{(2)}$$

The next step is to identify and remove instability (high variance) features, and it calculates as follows:

$$L(\omega, \beta|Y) = \sum_{i=1}^{m}\{Y_i - \sum_{j=0}^{n} \omega_j D_{ij} + \beta_j)\}^2 + \lambda \sum_{j=0}^{n} |\omega_j|^2 \quad \textbf{(3)}$$

Equation 3 is applied repeatedly to all non-linear data stored in spark resilient distributed data (RDD). Equation 3 converts the features from high variance to low variance. It minimizes overfitting and withal minimizes the intricacy of the model. Spark RDD is responsible to distribute the minimal variance data across the different clusters in RDD. All operations such as job scheduling, task dispatching, and basic input/output operations running inside RDD.

GRID SEARCH

In very large data environments, real-time traffic data is non-linear and non-parametric methods are affected by Extract Transform and Load (ETL) performance. The exhaustive search was used to process the non-linear traffic data. This search aims to find all possible solutions with minimal errors. It works as follows:

1. Set of parameters (p) with the possible values (v) supplies as input.
2. Exhaustive search (grid search), select parameters desultorily from the given set.
3. If p=10 and v=40, this engenders parallel accumulations of v^p parameters. The possible amalgamations of parameters distributed among the different groups of H2O clusters are processed independently and return a possible DNN model solution with minimal error.

Table 1. Set of parameters- Grid Search

Set of parameters(p)	Description	Values(v)
Activation	Transfers the weighted sum input to the next layer of the network. Used to train intricate data models.	Rectifier, Tanh, Rectifier with Dropout, Tanh with Dropout
Hidden Layer	Size of hidden layers.	{100, 100, 100} {200, 200, 200}
Epochs	Number of training iterations.	100, 200
L1 Regularization	Lasso technique: Minimize errors.	0, 0.00001, 0.0001
L2 Regularization	Ridge regularization: Minimize errors.	0, 0.00001, 0.0001
Learning rate (η)	Small value of optimization algorithm to minimize loss function.	0, 01, 0.005, 0.001
Rate annealing	η Reduced after number of training iterations.	1e-8, 1e-7, 1e-6
Rho	Decay factor.	0.9, 0.95, 0.99, 0.999
Epsilon	Update gradient.	1e-10, 1e-8, 1e-6
Momentum starts	Improves training rapidity and precision.	0, 0.5
Momentum stable	Speed up.	0.99, 0.5, 0
Input dropout ratio	Prevent model from over-fitting.	0, 0.1, 0.2
Max w2	Maximum sum of squared weights of neuron.	10, 100, 1000, 3.4028235e+38
Search Criteria	Select optimum model.	Random Discrete

DISTRIBUTED IN-MEMORY RANDOMIZED PARALLEL GRID SEARCH IN DNN

Our GSGDRR-DNN model allows greater scalability between spark clusters by adjusting the number of available processors and efficiently using memory resources for computation. GSGDRR-DNN performs arbitrary parallel distributed grid search computations accelerated on the multi-node cluster shown in Figure 3. This multi-node parallel GSGDRR-DNN cluster is superior to the sequential model. The models are trained in parallel in the H2O cluster to minimize processing time. Parallelism engenders several models concurrently. To improve the forecast accuracy, the suggested model uses an exhaustive search. This search works to finds the optimal combinations of hyperparameters using random discrete.

GSGDRR-DNN effectively deepens the neural network models and greatly improves the prediction accuracy. The GSGDRR-DNN model performs the fastest calculations during the grid search process based on fine-grained parallelization methods. The GSGDRR-DNN method uses the structure of sparkling water (Spark+H2O) to achieve higher scalability, maximum memory efficiency, faster computation and accurate predictions in deep neural networks (DNNs). GSGDRR-DNN architecture illustrates in Figure 2.

This works as follows: Load a large amount of data into the spark cluster. Spark cluster's RDD enhances in-memory computing. RDDs are responsible for splitting, storing, and distributing data as storage objects between different tasks. Data exchange in memory is faster than hard disks and networks. Spark and H2O connect to sparkling water via an external server connection. Spark and H2O have separate clusters to store and process non-linear traffic data. Send data between spark and H2O cluster over the network by calling the spark driver on the H2O node. The H2O cluster contains a large number of distributed H2O in-memory nodes that make the computation very fast and efficient. All processes such as DNN training, DNN modelling, and prediction are performed in a distributed H2O cluster. For quick estimation, all methods such as training, model building, forecasting, etc. are performed on the number of H2O nodes. Feature selection is a subsequent factor in DNN model design and significantly changes the accuracy of forecast. The prediction accuracy of DNN models is positively correlated with the quality of the input function. GSGDRR-DNN model algorithm 1 is well-structured as follows.

Algorithm 1: GSGDRR-DNN model

Input: Traffic Data set (TD), No of clusters (N), p,v

Output: Optimum model (OM)

1. N<-No of Spark Clusters
2. D<-Split (Traffic Data)
3. RDD (N) <-store (D)
4. H2O Cluster <-Transfer (RDD (N)
5. All nodes in H2O Cluster do { Parallel: SGD with Ridge Regression in DNN}
6. Return optimum model(p,v)
7. Prediction<- OM (p,v)
8. Return Prediction
9. RDD Frame <- Prediction

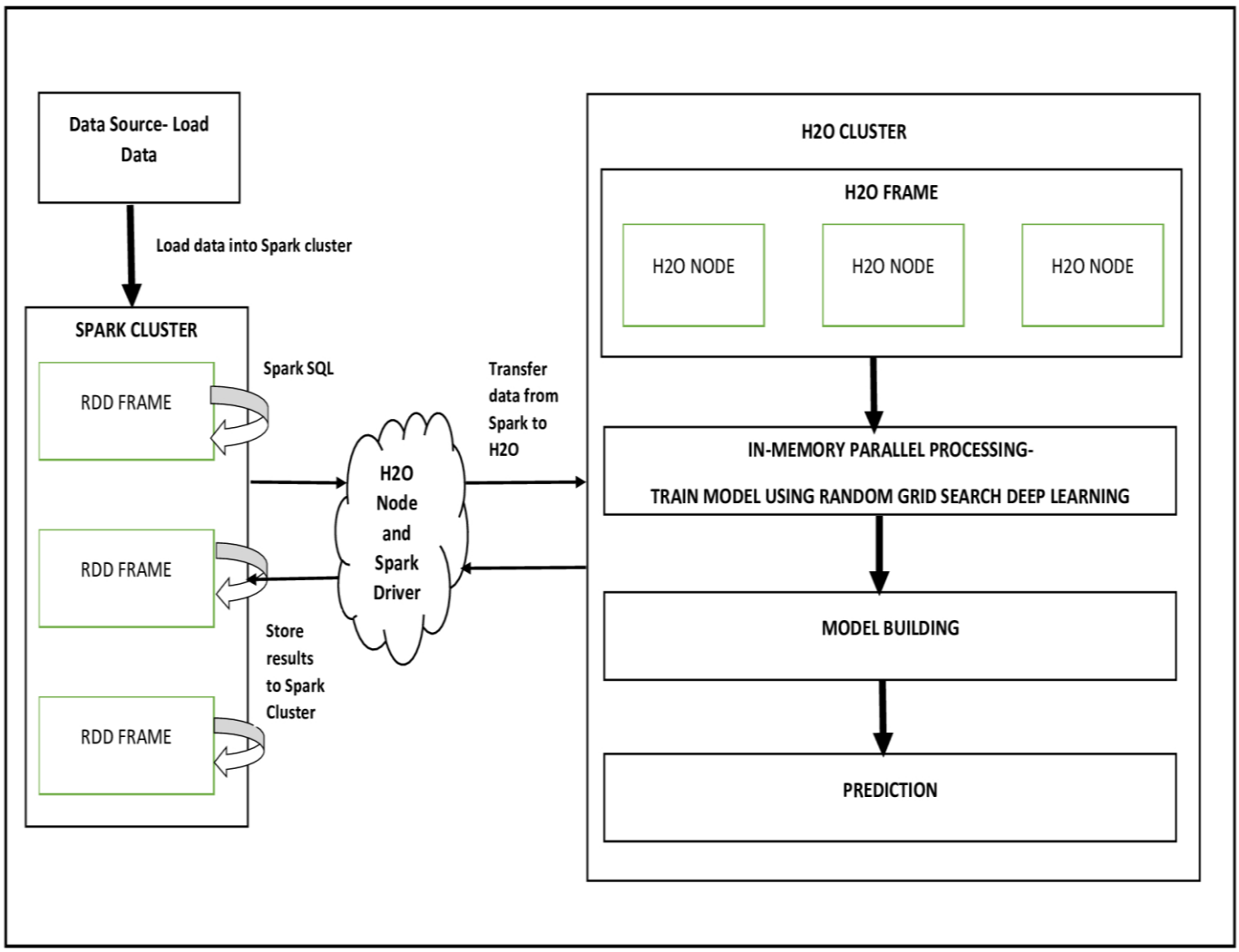


Figure 2. The GSGDRR-DNN architecture uses Spark+H2O cluster structure to improve scalability, maximize memory efficiency, speed up computation, and produce more accurate DNN predictions. These include grid search, kernel ridge regression techniques, stochastic gradient descent (SGD) approaches, and simultaneous in-memory distributed computing in DNNs.

STOCHASTIC GRADIENT DESCENT WITH RIDGE REGRESSION IN DNN

Increasing the number of layers in the DNN creates the problem of over fitting which greatly affects the accuracy of the forecast. The suggested GSGDRR-DNN diminishes multicollinearity, overfitting, and involution of the model. The model trained with 13 sets of hyperparameters (p) with 40 different values (v) is shown in Table 1. This will generate a combination of 6.711e+20 that will be evaluated and compared. All nodes in the H2O clusters perform parallel SGD ridge regression. This will generate the optimal set of parameters using the values listed in Table 2. To avoid CPU and memory constraints and minimize computational time, multiple tasks are dispatched in parallel during the grid search process shown in Figure 3. GSGDRR-DNN can provide weight updates $\nabla L(\omega, \beta|Y)$ asynchronously for each training data (TD) independently. SGD with ridge regression $\nabla L(\omega, \beta|Y)$ calculated by back propagation.

GSGDRR-DNN provides faster computational speeds than sequential mechanisms without sacrificing performance. In-memory model training and parallel distributed multiprocessor using parallel distributed grid SGD in the H2O clusters illustrated in Figure 3. For DNN training, the presented GSGDRR-DNN model uses parallel grid stochastic gradient descent (SGD) with ridge regression optimization techniques to minimize the loss function. It yields the lowest error in training and validation shown in Figure 4. The training model is deepened and the prediction accuracy is greatly improved. When training DNN with SGD, the H2O clusters use the rho and epsilon parameters to balance global and local search performance. Other combinations of manual hyperparameters such as rate, rate_annealing, momentum_start, momentum_stable are the most needed to improve the accuracy of the DNN model.

SGD with Ridge Regression in DNN Pseudo code

Step 1: Initialize weight ω, bias β, learning rate η, regularization parameter λ.

Step 2: For each H2O cluster, get training data Tx ∈ [D_1, D_2, D_3 ……, D_n] to perform SGD with Ridge Regression defined in equation (3).

$$\mathrm{f(D)} = \begin{cases} 0.01\mathrm{D}, & \mathrm{D} < 0 \\ \mathrm{D}, & \mathrm{D} > 0 \end{cases} \quad \textbf{(4)}$$

where $f(D)$ Ramp activation function.

Step 3: Repeat until minimum variance and bias is obtained.

for i =1, 2, ……., m do:

$\nabla L(\omega, \beta|\mathrm{Y})$ with respect to ω using SGD with Ridge Regression and Ramp activation function calculated as follows:

$$\frac{\nabla L}{\nabla \omega} = \frac{\nabla L}{\nabla \tilde{Y}} * \frac{\nabla \tilde{Y}}{\nabla \omega} \quad \textbf{(5)}$$

From equation (3),

$$\tilde{Y} = \sum_{j=0}^{n} \omega_j D_{ij} + \beta_j \quad \textbf{(6)}$$

where $\tilde{Y}$ predicted output.

$$\frac{\nabla L}{\nabla \tilde{Y}} = -2\left(1 - \tilde{Y}\right) + 2\,\lambda\omega \quad \textbf{(7)}$$

$$\frac{\nabla \tilde{Y}}{\nabla \omega} = \frac{\nabla \tilde{Y}}{\nabla \mathrm{h}} * \frac{\nabla \mathrm{h}}{\nabla \omega} \quad \textbf{(8)}$$

similarly, from hidden layer h calculated as follows:

$$\tilde{Y} = \sum_{j=0}^{n} \omega_j h_{ij} + \beta_j \quad \textbf{(9)}$$

$$\frac{\nabla \tilde{Y}}{\nabla \mathrm{h}} = \omega * f'(\omega_j h_{ij} + \beta_j) \quad \textbf{(10)}$$

$$\frac{\nabla \mathrm{h}}{\nabla \omega} = D_{ij} * f'(\omega_j D_{ij} + \beta_j) \quad \textbf{(11)}$$

where $h_{ij} = f(\omega_j D_{ij} + \beta_j)$, $f'(D) = \begin{cases} 0.01, & D < 0 \\ 1, & D > 0 \end{cases}$

$$\frac{\nabla L}{\nabla \omega} = -2\sum_{i=1}^{m} D_{ij}\left\{Y_i - \sum_{j=0}^{n} \omega_j D_{ij}\right\} + 2\,\lambda\beta_j \quad \textbf{(12)}$$

Step 4: Calculate updated weight,

$$\omega_j^{t+1} = \omega_j^t + \eta\{-2\sum_{i=1}^{m} D_{ij}\left\{Y_i - \sum_{j=0}^{n} \omega_j D_{ij}\right\} + 2\,\lambda\beta_j\} \quad \textbf{(13)}$$

Step 5: Go back to Step 2.

Table 2. GSGDRR-DNN Optimum hyperparameters

P	v
Activation	Rectifier
Hidden Layer	{200,200,200}
Epochs	100
L1 Regularization	1.0e-5
L2 Regularization	0.0
Learning rate (η)	1
Rate annealing	1.0e-6
Rho	0.999
Epsilon	1.0e-8
Momentum starts	0.5
Momentum stable	0.5
Input dropout ratio	0.0
Max w2	10
Search Criteria	Random Discrete

Table 3. GSGDRR-DNN status of neuron layers

S.No	Layer	Units	Type	Dropout	L1	L2	Mean_rate	Rate_rms	Momentum	Mean_weight	Weight_rms	Mean_bias	Bias_rms
1	1	7	Input	0.0%	nap	nap	nap	nap	nap	nap	nap	nap	nap
2	2	200	Rectifier	0.0%	0.00001	0.0	0.839	0.185	0.0	0.0015	0.058	0.002	0.03
3	3	200	Rectifier	0.0%	0.00001	0.0	0.834	00276	0.0	-0.0046	0.0269	0.0153	0.2182
4	4	200	Rectifier	0.0%	0.00001	0.0	0.446	0.452	0.0	-0.0089	0.0547	-0.299	0.4208
5	5	1	Linear	Nap	0.00001	0.0	0.612	0.385	0.0	-0.0062	0.0605	-0.102	0.0000

*nap – not applicable

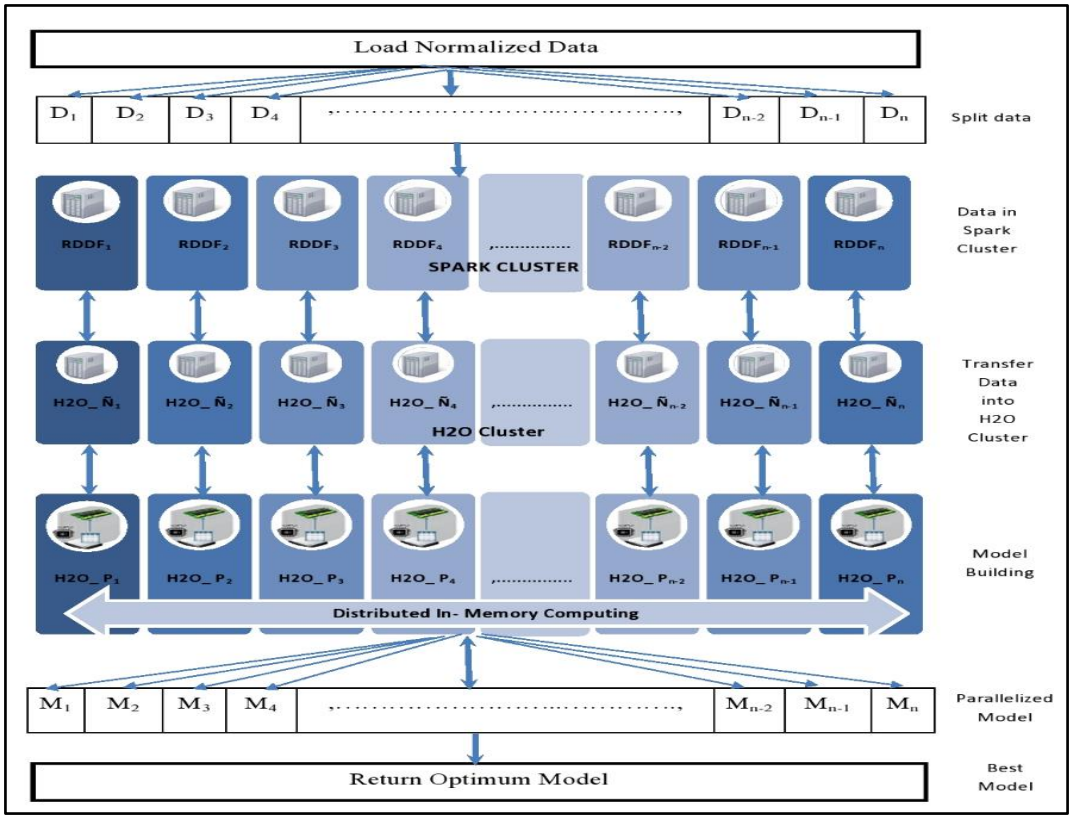


Figure 3. The GSGDRR-DNN parallel distributed grid search model efficiently handles large combinations of parameters that can be assigned to different groups of H2O clusters, independently managed by adjusting the processors and memory resources available for computation. It generates the ideal DNN model with minimal error.

Our investigated model accomplishes higher computational efficiency and greater forecasting accuracy. Finally, we evaluated the suggested model's performance with state-of-art techniques and other NPDL techniques such as LSTM, Bi-LSTM (Bi-directional LSTM) and GRU. The result shows that our GSGDRR-DNN model efficaciously presages traffic flow with more minute error metrics than other state-of-art and NPDL techniques. We also compared the runtime performance of the GSGDRR-DNN sequential model to the GSGDRR-DNN parallel model shown in Figure 6. The multiple node parallel GSGDRR-DNN model exploits the in-memory meritoriously and accomplishes the most expeditious computations to forecast traffic flow.

EXPERIMENTAL RESULTS AND DISCUSSION

DATASET

The GSGDRR-DNN model was implemented as 5-minutes traffic flow in the data collected from PeMS (Caltrans Performance Measurement System) database (https://pems.dot.ca.gov). To assess the effectiveness of the suggested GSGDRR-DNN model, we consider that five minutes of traffic and speed data were composed between January 01 and February 25, 2021. To measure the predicted performance of the proposed model, entire data was randomly split into training data by 60%, validation data by 20% and the remaining 20% by testing data.

PREDICTIVE EVALUATION MEASURES

In this article, we examine five performance metrics for predictive error analysis to assess the predictive effectiveness of the proposed model, including mean squared error (MSE), root mean squared error (RMSE), the mean absolute error (MAE), R^2 (R-squared Error) and the root mean squared logarithmic error (RMSLE). It defined as follows:

$$MSE = \frac{1}{D}\sum_{i=i}^{D}(\widetilde{Y} - Y_i)^2 \quad \textbf{(14)}$$

$$RMSE = \sqrt{\frac{1}{D}\sum_{i=i}^{D}(\widetilde{Y} - Y_i)^2} \quad \textbf{(15)}$$

$$MAE = \frac{1}{D}\sum_{i=1}^{D}|(\widetilde{Y} - Y_i)| \quad \textbf{(16)}$$

$$R^2 = \frac{\sum \widetilde{Y}^2}{\sum {Y_i}^2} \quad \textbf{(17)}$$

$$RMSLE = \sqrt{\frac{1}{n}\sum_{i=i}^{n}(\log(\widetilde{Y} + 1) - (\log(Y_i + 1))^2} \quad \textbf{(18)}$$

Where, D = Range of data, Y_i = Actual Output, $\widetilde{Y}$ = Predicted Output.

ANALYSIS OF FORECAST PERFORMANCE

The GSGDRR-DNN model trained with all possible 6.711e+20 combinations of hyperparameters tabulates is in Table 2. GSGDRR-DNN model used deep learning techniques to randomly searches for parallel meshes to find patterns. Optimal hyperparameters for training the model greatly increase prediction accuracy. The suggested model structure and the status of neuron layers shows in Table 3. GSGDRR-DNN model used H2O ridge regression deviance techniques obtained consistent training and validation predictions illustrated in Figure 4. The model achieves the smallest RMSE, deviance and MAE error values of 0.1309, 0.0536 and 0.0123 at epoch's size 9.

By evaluating the GSGDRR-DNN model, the traffic flow reproduced stochastically every 5 minutes. It assesses through the ramp activation function since the mean & the variance of the traffic flow change significantly from time to time. This approach considerably decreases the complication in model building. Feature selection plays a decisive role in predicting forecast accuracy. To shorten the execution time of the training process, the GSGDRR-DNN model excludes the variables that make a small contribution during the training process using ridge regression. This process is known as the significance of the variable. The variable Lane 3 Flow (Veh/5 Minutes) highly contributed to build and train the DNN model.

We compared the indicators of training and testing of various regression models with different values of λ- parameters. In comparison, ridge regression achieved the minimum bias and variance between training and test metrics than other regression models shown in Table 4, which considerably improves prediction accuracy.

To compare the performance of our GSGDRR-DNN model with other familiar DNN models like the LSTM, Bi-LSTM and GRU model shown in Figure 5. Our model performs well and correctly predicts the actual traffic flow. It achieves the greater prediction accuracy for predicting the traffic flow shown in Figure 5. The figure shows that our model achieved the highest prediction accuracy and fastest computation by parallelizing the multiple tasks across multiple H2O clusters using the random grid search method and returns the best DNN model to forecast the traffic flow. The whole process carries out through an in-memory computation. By comparison, the model achieves the consistent traffic flow prediction with the smallest RMSE, MAE, deviance and MAE.

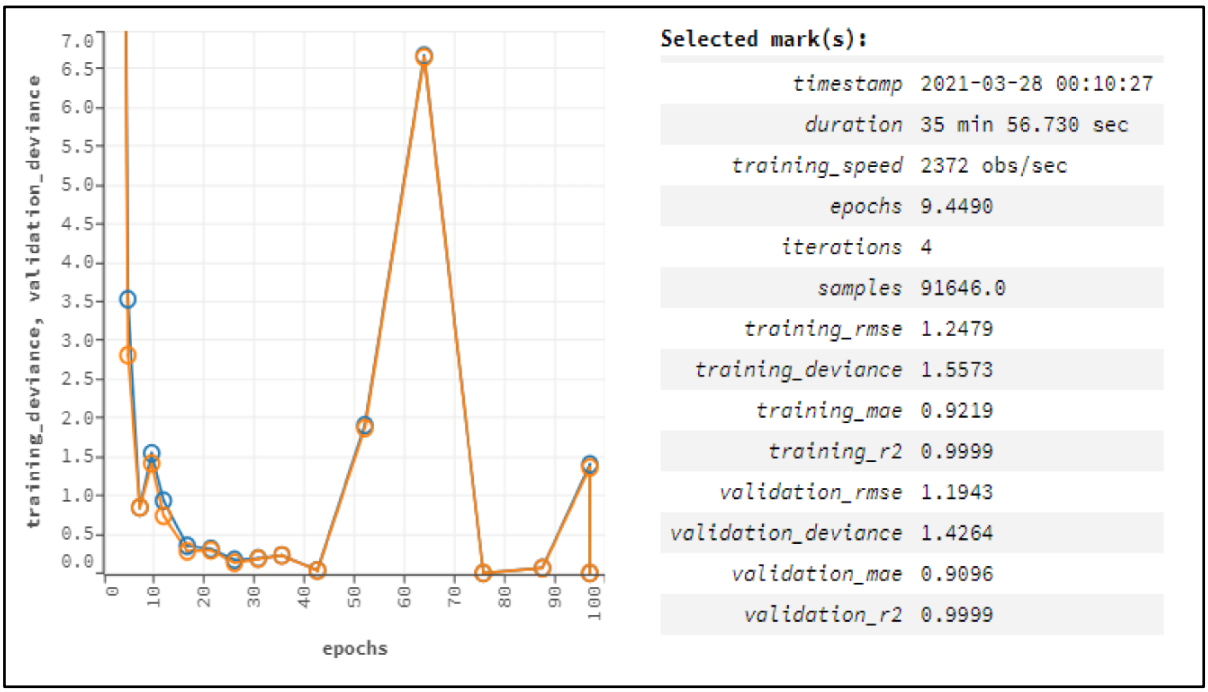
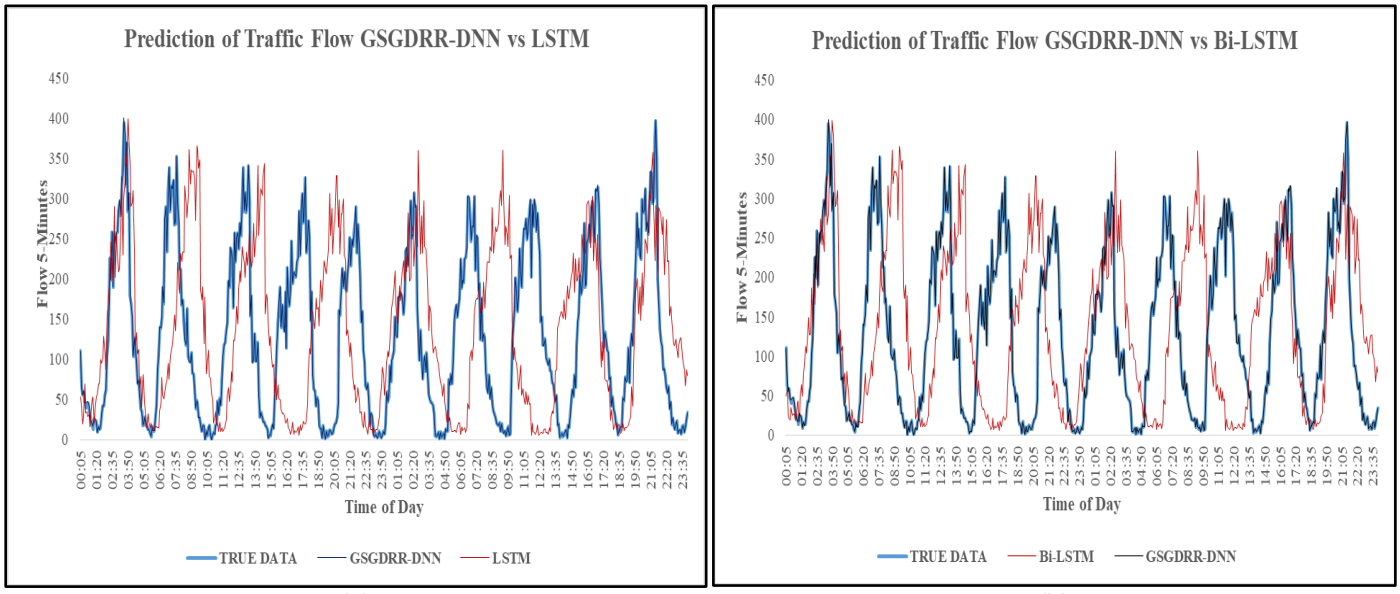
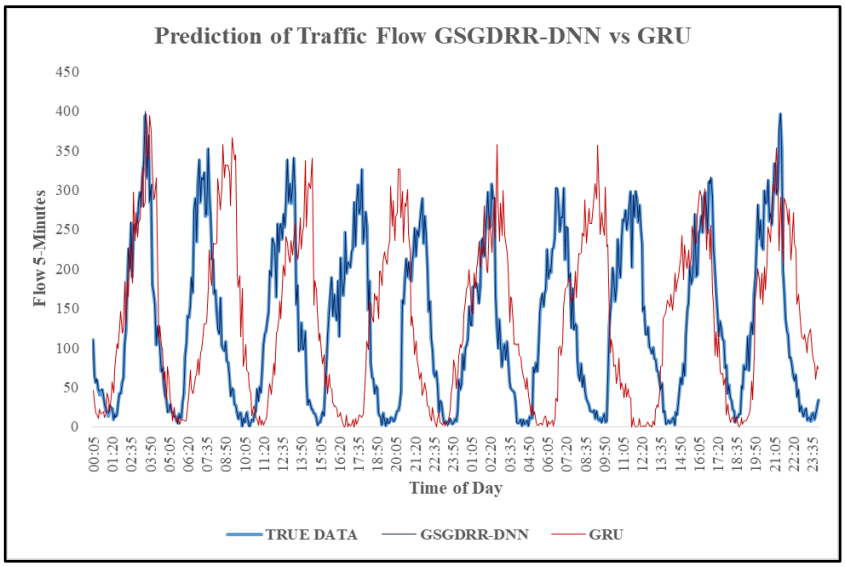


Figure 4. Scoring History: Training and Validation Deviation. The model obtains the lowest RMSE, deviance, and MAE values of 0.1309, 0.0536, and 0.0123 for epochs of size 9.44 and provides consistent validation with training predictions using the H2O ridge regression deviance technique.



(a) (b)



(c)

Figure 5. 5-minute performance comparison of actual and predicted traffic flows for different models such as LSTM, Bi-LSTM, and GRU. (a) GSGDRR-DNN vs. LSTM (b) GSGDRR-DNN vs. Bi-LSTM (c) GSGDRR-DNN vs. GRU. The GSGDRR-DNN model provides the lowest error value for 5-minute traffic flow prediction as the MSE, RMSE, MAE, and RMSLE values are 0.011, 0.108, 0.0959, and 0.0147, respectively.

Traffic flow prediction of 5-minutes, 10-minutes, 15-minutes, 20-minutes, 25-minutes, and 30-minutes intervals compared with other well-known non-parametric methods shown in Table 5. The average generalizability of LSTM, Bi-LSTM, and GRU for 10-minutes MAE time series prediction is 17%, 18%, and 17.89% respectively. The average generalizability of LSTM, Bi-LSTM, and GRU for 30-minutes MAE time series prediction is 19%, 28%, and 23% respectively. The average generalizability of LSTM, Bi-LSTM, and GRU for RMSE time series prediction is 25.25%, 29.9%, and 25.66% respectively. Results show that our model accomplished extreme mean prediction accuracy enhancement of MAE up to 26% compared with Bi-LSTM, over 17% compared with LSTM and over 21% compared with GRU. Average accuracy enhancement of RMSE up to 24.25% compared with LSTM, above 28.9% compared with Bi-LSTM and up to 24.66% compared with GRU.

Table 4. Trade-off between training and testing error metrics using various Regression Models

Alpha Value λ	Regression	Training Error Metrics				Testing Error Metrics			
		MSE	RMSE	MAE	R^2	MSE	RMSE	MAE	R^2
0.5	Ridge	73.8876	8.5958	6.3750	0.9932	73.2866	8.56076	6.3981	0.9933
	Lasso	82.1628	9.0644	6.6681	0.9924	83.1347	9.1178	6.7687	0.9924
	Elastic net	5758.997	75.8881	66.5125	0.4669	5821.781	76.3006	66.8389	0.4652
0.1	Ridge	73.8795	8.5953	6.3790	0.9932	73.2542	8.5589	6.4019	0.9933
	Lasso	75.7629	8.7042	6.4504	0.9930	75.3253	8.6790	6.4625	0.9931
	Elastic net	1396.588	37.3709	32.4334	0.8707	1421.064	37.6970	32.7319	0.8695
0.01	Ridge	73.8792	8.5953	6.3799	0.9932	73.2484	8.5585	6.4028	0.9933
	Lasso	74.1484	8.6106	6.4004	0.9931	73.4811	8.5718	6.4123	0.9933
	Elastic net	116.7115	10.6679	8.3435	0.9895	116.7115	10.8033	8.3435	0.9893
0.001	Ridge	73.8792	8.5953	6.3801	0.9932	73.2478	8.5585	6.4029	0.999
	Lasso	73.8824	8.5955	6.3817	0.9932	73.2452	8.5583	6.4034	0.9933
	Elastic net	74.7066	8.6433	6.3545	0.9931	74.4361	8.6277	6.3879	0.9932

Our GSGDRR-DNN model provides a higher minimum error than other DNN models. Conversely, the prediction error rate contributes to the accuracy of the prediction. As the error decreases, the accuracy of the prediction automatically increases. Our model gives very minimal error values for 5 minutes traffic flow prediction such as MSE, RMSE, MAE, and RMSLE values are 0.011, 0.108, 0.0959, and 0.0147 respectively. It is the minimum error no any other model gives such low error values. Our model hit an R^2 value of 0.999 which, is an adequate estimation as shown in Table 5. Creating a DNN model is a time-consuming process. To minimize the computation time distributes the multiple tasks in parallel on the H2O cluster during the training process. Our model generates multiple distributed DNN models either by sequential or parallel random grid search.

The parallel grid search used four cores on a single CPU with up to 4GB of memory. It allows multiple calculations performed at the same time. These CPUs have a large amount of memory to avoid the computing work related to memory. The suggested GSGDRR-DNN model achieves high performance by computing in parallel execution shown in Figure 6.

We compared the execution time for the sequential building of the DNN model with the grid search parallel DNN models. The results demonstrate that compared to the sequential grid search approach, the parallel grid search DNN model offers faster execution time. Training of various DNN models is very complicated and takes 24- 48 hours. Our model computes several DNN training models with combinations of entire hyperparameters in less than 5 hours using 12 parallel single CPU cores. It supports up to 64GB of memory. With the grid search parallel technique, we generate several models within the fraction of seconds shown in Figure 6.

Finally, we compared our method to other analogous state-of-art methods to show that the GSGDRR-DNN model outperforms well with minimal errors shown in Table 6. For MSE, the model achieved the highest minimum error rate of 2.8% it is relatively reduced compared with other state-of-art techniques such as Ma X and coauthors [15], Fu R and coauthors [17] (LSTM) and Fu R and coauthors [17] (GRU) error values are 4.08,710.05,668.93 respectively. RMSE, our model outperformed other techniques of Lv Y and coauthors [18] and Emami A and coauthors [4] and our model improves forecasting accuracy in all aspects of error metrics such as MAE, RMSLE, R^2, MAPE, PERR, MRE and MAD.

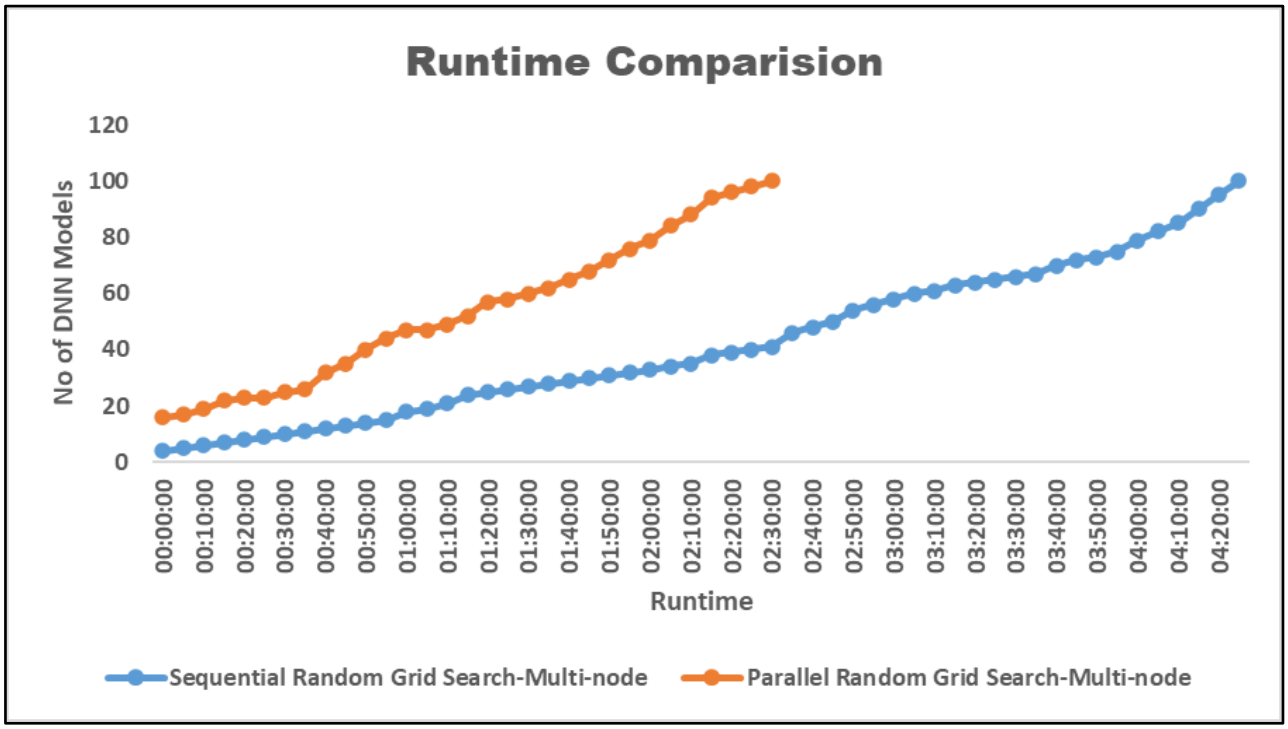


Figure 6. A runtime comparison of multi-node sequential grid search and multi-node parallel grid search. The parallel grid search approach requires a runtime of 04:27:13:1 hours to produce the optimal DNN models with a combination of hyperparameters, while the sequential grid search method requires 32:24:12:6 hours to produce the same optimal DNN models.

Table 5: Comparison of traffic flows at different time intervals using different models

S. No	Traffic Flow Prediction	Prediction error metrics																			
		GSGDRR-DNN					LSTM					Bi-LSTM					GRU				
		MSE	RMSE	MAE	RMSLE	R^2	MSE	RMSE	MAE	RMSLE	R^2	MSE	RMSE	MAE	RMSLE	R^2	MSE	RMSE	MAE	RMSLE	R^2
1	5- minutes	0.012	0.108	0.96	0.02	0.99	5.86	24.2	17.4	2.3	0.92	5.88	24.2	17.6	2.4	0.91	5.70	23.8	17.2	2.1	0.90
2	10-minutes	0.017	0.110	1.3	0.024	0.99	5.90	24.6	17.56	2.45	0.91	5.86	24.8	18.2	2.6	0.95	5.78	24.6	17.89	2.6	0.91
3	15-minutes	0.019	0.124	1.5	0.03	0.98	6.25	25.3	17.92	2.98	0.95	5.46	26.5	18.9	2.9	0.94	5.91	25.2	20.3	2.678	0.92
4	20-minutes	0.026	0.154	1.89	0.035	0.99	6.54	25.46	18.62	3.64	0.94	5.21	30.2	22.53	3.2	0.96	6.05	26.23	20.6	2.78	0.94
5	25-minutes	0.032	0.189	2.2	0.056	0.99	6.56	25.84	18.74	3.68	0.95	5.90	36.4	24.57	3.8	0.96	6.06	26.54	21.23	2.96	0.97
6	30-minutes	0.062	0.2	2.31	0.06	0.98	6.59	26.08	19.23	4.5	0.96	6.03	37.5	28.9	3.95	0.97	6.52	27.58	23.45	3.24	0.94

CONCLUSION

We came up with distributed in-memory randomized parallel Grid search SGD with Ridge Regression amalgamated together to train the DNN model. It greatly truncates multilinearity, overfitting, and involution of the model. The model also focuses on scalability, effectively distributing in-memory over numerous H2O clusters to perform quick computations and determining the appropriate DNN hyperparameters. The method obtained more preponderant precision in the prognostication of traffic flow. It achieves a minimum number of errors by adjusting parameter λ in Ridge Regression**.** To train DNN models, parallelize several DNN models across the multiple H2O clusters utilizing the GSGDRR-DNN parallel multi-node SGD training method. GSGDRR-DNN grid search parallel model offers more expeditious execution times through efficient use of H2O cluster memory and achieves faster processing speed than the GSGDRR-DNN sequential model. GSGDRR-DNN model compared to other familiar models such as LSTM, Bi-LSTM and GRU. Compared with other models, our model got the lowest MSE, RMSE, MAE, RMSLE and highest R2 values such as 0.012, 0.108, 0.096, 0.015 and 0.99 accordingly, which results in better precision in the prognostication of traffic flow.

Table 6. Performance comparisons of various DNN state-of-art techniques

State-of-art	Performance metrics								
	MSE	**RMSE**	**MAE**	**RMSLE**	**R^2**	**MAPE**	**PERR**	**MRE**	**MAD**
GSGDRR-DNN [Proposed Method]	0.028	0.1475	1.693	0.038	0.99	nap	nap	nap	nap
Williams BM et.al. [1]	nap	nap	nap	nap	nap	8.6	nap	nap	384
Castro-Neto M et al. [3]	nap	nap	nap	nap	nap	13.1	nap	nap	nap
Emami A et al. [4]	nap	0.3313	0.0366	nap	0.9797	0.0050	nap	nap	nap
Ma X et al. [15]	4.08	nap	nap	nap	nap	2.88	nap	nap	nap
Fu R et.al. (LSTM) [17]	710.0502	nap	18.13	nap	nap	nap	nap	nap	nap
Fu R et.al. (GRU) [17]	668.93	nap	17.2116	nap	nap	nap	nap	nap	nap
Lv Y et al. [18]	nap	183.9	122.8	nap	nap	nap	nap	6.21	nap
Vlahogianni EI et al. [23]	nap	nap	6	nap	0.95	nap	nap	8.54	nap
Chen D [34]	nap	nap	0.0025	nap	nap	nap	0.0869	nap	nap

*nap-not applicable.

Data Availability: The data that support the findings of this study are openly available Caltrans Performance Measurement System at https://pems.dot.ca.gov.
Funding Information: No funding was received for this article.
Ethical Statement: There are no human or animal experiments in this research article.
Conflicts of Interest: No conflicts of interest in this work.

REFERENCES

1. Williams BM, Hoel LA. Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results. J Transp Eng. 2003 Nov;129(6):664-72.
2. Okutani I, Stephanedes YJ. Dynamic prediction of traffic volume through Kalman filtering theory. Transp Res B Methodol. 1984 Feb 1;18(1):1.
3. Castro-Neto M, Jeong YS, Jeong MK, Han LD. Online-SVR for short-term traffic flow prediction under typical and atypical traffic conditions. Expert Syst Appl. 2009 Apr 1;36(3):6164-73.
4. Emami A, Sarvi M, Asadi Bagloee S. Using Kalman filter algorithm for short-term traffic flow prediction in a connected vehicle environment. J Mod Transp. 2019 Sep 4;27:222-32.
5. Gong X, Wang F. Three improvements on KNN-NPR for traffic flow forecasting. InProceedings. The IEEE 5th International Conference on Intell Transp Syst 2002 Sep 6 (pp. 736-740). IEEE.
6. Zheng W, Lee DH, Shi Q. Short-term freeway traffic flow prediction: Bayesian combined neural network approach. J Transp Eng. 2006 Feb;132(2):114-21.
7. Zhong M, Sharma S, Lingras P. Short-term traffic prediction on different types of roads with genetically designed regression and time delay neural network models. J Comput Civ Eng. 2005 Jan;19(1):94-103.
8. Dia H. An object-oriented neural network approach to short-term traffic forecasting. Eur J Oper Res. 2001 Jun 1;131(2):253-61.

9. Yin H, Wong S, Xu J, Wong CK. Urban traffic flow prediction using a fuzzy-neural approach. Transp Res Part C Emerg. 2002 Apr 1;10(2):85-98.
10. Kumar K, Parida M, Katiyar VK. Short term traffic flow prediction for a non-urban highway using artificial neural network. Procedia Soc Behav Sci. 2013 Dec 2;104:755-64.
11. Dougherty M. A review of neural networks applied to transport. Transp Res Part C Emerg. 1995 Aug 1;3(4):247-60.
12. Yu B, Song X, Guan F, Yang Z, Yao B. k-Nearest neighbor model for multiple-time-step prediction of short-term traffic condition. J Transp Eng. 2016 Jun 1;142(6):04016018.
13. Ghosh B, Basu B, O'Mahony M. Multivariate short-term traffic flow forecasting using time-series analysis. IEEE Trans Intell Transp Syst. 2009 May 5;10(2):246-54.
14. Zhang J, Wang FY, Wang K, Lin WH, Xu X, Chen C. Data-driven intelligent transportation systems: A survey. IEEE Trans Intell Transp Syst. 2011 Jul 21;12(4):1624-39.
15. Ma X, Tao Z, Wang Y, Yu H, Wang Y. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. Transp Res Part C Emerg. 2015 May 1;54:187-97.
16. Zhao Z, Chen W, Wu X, Chen PC, Liu J. LSTM network: a deep learning approach for short-term traffic forecast. IET Intell Transp Syst. 2017 Mar;11(2):68-75.
17. Fu R, Zhang Z, Li L. Using LSTM and GRU neural network methods for traffic flow prediction. In2016 31st Youth academic annual conference of Chinese association of automation (YAC) 2016 Nov 11 (pp. 324-328). IEEE.
18. Lv Y, Duan Y, Kang W, Li Z, Wang FY. Traffic flow prediction with big data: A deep learning approach. IEEE Trans Intell Transp Syst. 2014 Sep 9;16(2):865-73.
19. Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Commun ACM. 2017 May 24;60(6):84-90.
20. Li P, Xie J, Yan W, Li Z, Kuang G. Living face verification via multi-CNNs. Int J Comput Intell Syst. 2018 Nov 1;12(1):183-9.
21. Atrey K, Singh BK, Bodhey NK. Multi-Feature Classification of Breast Cancer Histopathology Images: An Experimental Investigation in Machine Learning and Deep Learning Paradigm. Braz Arch Biol Technol. 2023 May 22;66:e23220297.
22. Annamalai M, Muthiah PB. An early prediction of tumor in heart by cardiac masses classification in echocardiogram images using robust back propagation neural network classifier. Braz Arch Biol Technol. 2022 Apr 22;65:e22210316.
23. Vlahogianni EI, Karlaftis MG, Golias JC. Optimized and meta-optimized neural networks for short-term traffic flow prediction: A genetic approach. Transp Res Part C Emerg. 2005 Jun 1;13(3):211-34.
24. Chan KY, Dillon TS, Singh J, Chang E. Neural-network-based models for short-term traffic flow forecasting using a hybrid exponential smoothing and Levenberg–Marquardt algorithm. IEEE Trans Intell Transp Syst. 2011 Nov 28;13(2):644-54.
25. Sun B, Cheng W, Goswami P, Bai G. Short-term traffic forecasting using self-adjusting k-nearest neighbours. IET Intell Transp Syst. 2018 Feb;12(1):41-8.
26. Gurusamy R, Seenivasan SR. DGSLSTM: deep gated stacked long short-term memory neural network for traffic flow forecasting of transportation networks on big data environment. Big Data. 2022 Feb 10.
27. Tian Y, Pan L. Predicting short-term traffic flow by long short-term memory recurrent neural network. In2015 IEEE international conference on smart city/SocialCom/SustainCom (SmartCity) 2015 Dec 19 (pp. 153-158). IEEE.
28. Jiang H, Zou Y, Zhang S, Tang J, Wang Y. Short-term speed prediction using remote microwave sensor data: machine learning versus statistical model. Math Probl Eng. 2016 Feb 2;2016.
29. Huang W, Meng L, Zhang D, Zhang W. In-memory parallel processing of massive remotely sensed data using an apache spark on hadoop yarn model. IEEE J Sel Top in Appl Earth Obs Remote Sens. 2016 May 4;10(1):3-19.
30. Nabavinejad SM, Goudarzi M, Mozaffari S. The memory challenge in reduce phase of MapReduce applications. IEEE Trans Big Data. 2016 Sep 9;2(4):380-6.
31. Tang S, Lee BS, He B. Fair resource allocation for data-intensive computing in the cloud. IEEE Trans Serv Comput. 2016 Feb 18;11(1):20-33.
32. Niu Z, Tang S, He B. An adaptive efficiency-fairness meta-scheduler for data-intensive computing. IEEE Trans Serv Comput. 2016 Dec 2;12(6):865-79.
33. Chen CP, Zhang CY. Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. Inf Sci. 2014 Aug 10;275:314-47.
34. Chen D. Research on traffic flow prediction in the big data environment based on the improved RBF neural network. IEEE Trans Ind Inform. 2017 Mar 15;13(4):2000-8.
35. Wang Z, Su X, Ding Z. Long-term traffic prediction based on lstm encoder-decoder architecture. IEEE Trans Intell Transp Syst. 2020 Jun 3;22(10):6561-71.