IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 2 April 2024, accepted 31 May 2024, date of publication 6 June 2024, date of current version 14 June 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3410316

RESEARCH ARTICLE

Feasibility Analysis of Applying Deep Neural Network on Driving Distance Estimation

SANGHWAN LEE[1] AND JINSOO MOON[2]
[1]Department of Computer Science, Kookmin University, Seoul 02707, South Korea
[2]m2Cloud Inc., Seoul 05855, South Korea

Corresponding author: Sanghwan Lee (sanghwan@kookmin.ac.kr)

This work was supported in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant 2022R1F1A1074672; and in part by the Grant from the Ministry of Food and Drug Safety under Grant 22183MFDS436.

ABSTRACT In numerous location-based applications, such as the vehicular routing problem, driving distances play a crucial role. However, these driving distances often differ from the direct geographic distance computed using latitude and longitude. Hence, accurately estimating the driving distance between two positions is vital for the success of such services. Researchers have worked on developing efficient methods to estimate driving distances, and it has been reported that the inflation ratio (or detour index) between driving distance and direct distance is approximately 1.3. However, this simple method may not suffice for complex road networks. To address these challenges, other researchers have proposed deep learning based approaches. They show relatively good performance for real road data sets. Even though the deep learning based approach may work for real road data sets, it crucial to fully understand the behavior of the deep neural network (DNN) based approach. Therefore, in this study, We aim to thoroughly examine a deep learning-based road network distance estimation method under controlled conditions. Specifically, we define five different distance types and assess the performance of the DNN-based approach. Subsequently, we analyze the key factors that influence its performance. Through extensive simulations, we demonstrate that the DNN-based method performs well across most distance definitions. After conducting a thorough analysis of the evaluation results, we have identified a key characteristic of the road data sets that significantly impacts the accuracy of the DNN-based method. Specifically, we have found that the ‘‘discontinuity’’ of the distances plays a crucial role in achieving high accuracy. As a result, we propose that future designs of DNN-based road network distance estimation methods should prioritize careful consideration of this ‘‘discontinuity’’ aspect to optimize their performance and ensure better accuracy.

INDEX TERMS DNN, distance, driving.

I. INTRODUCTION

The advent of modern technology has brought about significant changes in a multitude of industries and applications, not least of which is the way we estimate and utilize location-based data [1]. One such application, of paramount importance in the digital era, is vehicular routing [2]. This service, integral to a wide array of sectors including logistics, food delivery, ride-sharing, and more, thrives on its ability to accurately determine driving distances between two locations. However, it is a well-documented fact that the geographical distance, calculated using traditional latitude and longitude coordinates, often does not mirror the actual driving distance [3], [4], [5], [6], [7]. This is due to various factors such as road type, physical obstacles, road rules, and more. Thus, there is a pressing need for a more advanced and reliable approach for driving distance estimation, particularly for applications involving complex road networks.

Many researchers offer methods for estimating driving distances using an inflation ratio, which involves multiplying the direct geographical distance by a factor of approximately 1.3 [4]. This approximation, albeit simple, has proven to be reasonably effective in a variety of scenarios and applications. However, this method is not without its shortcomings.

The associate editor coordinating the review of this manuscript and approving it for publication was Rajeeb Dey.

In particular, it often fails to adequately represent distances in intricate road networks with numerous turns, one-way systems, and other complexities. Additionally, this method does not account for changes in the road network that might occur over time due to construction, urban development, natural disasters, or other causes [3]. In short, the traditional method of estimating driving distances based solely on geographic distance (latitude and longitude) may not be sufficient when dealing with complex road networks.

To tackle these challenges, some researchers have proposed deep learning-based approaches [8], [9]. They propose DNN models and evaluate the model based on some real road traffic data set. Even though their works work well for certain data sets, it is not clear whether they will perform well for other environments with different road types, traffic type, etc. In other words, it is crucial to gain a comprehensive understanding of how the deep learning based approach behaves in different scenarios. Actually, we have analyzed the deep learning based approach with 2 different distance definitions to understand in what circumstances the deep learning based method works well [10]. To be specific, we have considered two distance definitions: Euclidean distance and Delaunay Triangulation based distance. We have found that DNN based method works better for Euclidean distance than Delaunay Triangulation based distance. We conjectured that the "discontinuity" in the distances has impact on the estimation performance.

Thus, in this paper, we extend our preliminary work to more thoroughly analyze the deep learning based method in estimating driving distance. Our primary objective is to conduct a thorough examination of a deep learning-based road network distance estimation method under carefully controlled conditions. To achieve this, we define *five* different distance types with specific characteristics including the two distance type defined in [10]. We generate distance data sets based on the five distance types and evaluate the performance of the DNN-based approach on each. By this, we can better understand the behaviour of the DNN based approach.

In essence, the core objective of this paper revolves around demonstrating the performance disparities exhibited by the DNN-based method across various types of distance definitions. Additionally, we delve into the main factors that influence the method's performance. Our extensive simulations demonstrate that the DNN-based method generally performs well across a wide range of distance definitions. Furthermore, we identify the "discontinuity" as a critical determinant of its performance. The main idea of "discontinuity" is that for some distance definitions the distance does not change linearly or proportionally to the amount of changes of coordinates. This characteristic greatly affects the estimation performance of DNN based approach and other approach such as linear regression based approach. Given these findings, we emphasize the importance of considering the "discontinuity" in the design of future DNN-based road network distance estimation methods to optimize their overall performance. This deeper understanding will contribute to the development of more accurate and efficient distance estimation models that can address the complexities of modern road networks effectively.

The rest of the paper is organized as follows. In Section II, we describe the background and review the related works. Section III describe the neural network model that we examine and five different distance definitions. We present the analysis results of the neural network based approach in Section IV. In Section V, we discuss the "discontinuity" characteristics of the distances. Lastly, the paper is concluded in Section VI.

II. RELATED WORKS

In this section, we discuss previous studies that offer insights into the estimation of road distances. Then, we describe applications and software that utilize such distance estimation techniques.

Efficiently estimating the road distance is crucial for some critical applications such as health care services. Reference [3] shows overhead based road distance estimation. The authors introduce a fast method for estimating travel distances between locations using an overhead graph that stores representative location ratios among a small number of selected locations. The approach proves effective in complex optimization tasks that needs distances among nodes by reducing the burden of travel distance computations. Experimental results using real-world data from the North Karelia region in Finland demonstrate an average estimation error of 0.5 km with a 512-node graph. The method significantly reduces processing time in the optimization process, achieving a reduction from 1.2 hours to 2.9 seconds per iteration, outperforming alternative estimation methods with a smaller average estimation error of 2%.

A simpler distance estimation method is to use Detour Index (DI), which is basically the ratio of actual road network distance to the Euclidean or Geodesic distance. Authors of [4] investigate the limitations of using straight-line Euclidean distance (ED) as a distance metric in spatial analysis. They estimate the relationships between ED and actual network distance (ND) across 25 Chinese cities using functional data analysis (FDA) and the detour index (DI). The results show significant linear relationships between ND and ED, with an average DI value of 1.324 observed across all cities. Similarly, [5] compares the use of straight-line (Euclidean) distance with actual travel distance and time in geographic studies. They calculate travel distance and time directly from commercial websites, eliminating the need for specialized software or street files. The comparison is conducted using a representative sample of over 66,000 locations in the United States, including Puerto Rico and the District of Columbia, with a focus on travel to community hospitals. The measures show a strong correlation ($r^2 > 0.9$), indicating high linear relationship between the straight-line distance and the actual travel distance. For non-emergency applications, simple linear regression might be a good estimation method. Reference [11] also addresses the estimation of road distances using distance functions and the need for correction through a circuity factor to approximate actual travel distances. A circuity factor is a multiplier to coordinate-calculated, or straight-line, distances to approximate actual travel distances, which is basically DI defined in [4]. Reference [11] presents circuity factors derived from a sampling of road networks in different countries and regions worldwide.

Exploiting the Euclidean distance and the linear regression can be generalized to use the weighted l_p or $l_{k,p}$ norms. Reference [6] focuses on modeling travel distances in road networks using the weighted l_p or $l_{k,p}$ norm. The parameters k and p are computed by analyzing a sample of actual road distances from the geographical area of interest. The study examines 17 diverse geographical areas and demonstrates significantly improved results. It should be noted that determining k and p is an optimization problem based on the distance samples from the 17 diverse geographical areas.

Estimating the road distance can be useful. However estimating the travel time might be more helpful for some other applications. Reference [12] focuses on the importance of accurate travel-time estimation in a green time-dependent vehicle routing optimization problem to ensure precise objective function values and accurate decision-making. The proposed model considers multiple traffic modes and assumes routes between nodes consist of different segments with varying speed levels. Statistical analyses compare the estimated travel times of random vehicles with random departure times to data from Google Maps and show that the proposed method performs well. Estimation of travel time is also useful for many transportation systems such as New York tax service. Reference [13] explores the travel time estimation problem using large-scale trajectory data, particularly New York City Taxi trips dataset. Instead of traditional route-based methods, the proposed approach utilizes a vast amount of taxi trips to estimate travel time between source and destination without intermediate trajectory points. The experiments reveal promising results, as the big-data-driven approach outperforms state-of-the-art route-based methods and online map services. The study highlights the potential of simple big data-driven approaches to serve as new baselines for traditional computational problems.

Reference [16] also addresses travel time estimation using deep learning, which has gained traction due to available large trip datasets. However, existing methods often disregard road network information. The proposed approach integrates road networks and historical data, enhancing performance, particularly with smaller training sets. Incorporating node embedding and road distance leads to improved results, especially when road distance significantly differs from Vincenty distance. Experiments on real-world datasets highlight the method's efficacy. The focus on short-term traffic time prediction in the intelligent transportation field is well-established. However, anticipating long-term traffic time, crucial for personal and commercial planning, remains challenging due to various factors like weather and congestion. Reference [14] introduces the Deep Ensemble Stacked Long Short Term Memory (DE-SLSTM) framework, which integrates weather effects to address prediction bias during congestion. Through experiments, DE-SLSTM showcases strong performance, marking it as the first long-term traffic time prediction approach that incorporates deep learning techniques.

Since the road distance estimation is quite useful, some researchers provide software to estimate the road distance. Reference [15] introduces the ''osrmtime'' command, a platform-independent method for calculating travel time and distance between two points using geographic data on latitudes and longitudes. The method utilizes the Open Source Routing Machine (OSRM), a high-performance open-source C++ routing engine that uses open-source maps from OpenStreetMap [17]. OpenStreetMap's decentralized data collection and maintenance may have advantages and disadvantages compared to maps from commercial providers, but it is widely used in scientific research. Unlike existing commands that compute geodesic distances, ''osrmtime'' calculates the optimal route over public roads by car, bicycle, or on foot. It offers the advantages of being able to handle an unlimited number of requests, working offline, and providing efficient calculations within seconds. Another application based on OpenStreetMap is Rapidex. It is a novel tool for origin-destination (OD) demand estimation and visualization [7]. Rapidex addresses the limitations of data availability and development associated with traffic assignment models. Rapidex allows users to download and visualize road networks for any city, create traffic analysis zones, and fetch travel time data from providers like TomTom and Google. Using a genetic-algorithm (GA)-based approach, Rapidex derives the OD demand pattern and produces critical outputs such as link volumes, travel times, congestion levels, and average trip length. The tool enables scenario evaluation, allowing users to assess the impacts of network and demand data changes on performance metrics.

Many of the previously methods use linear regression based approach. Since the advancement of deep learning technology, there has been many deep learning based approaches. Reference [8] proposes a deep learning model, ST-NN (Spatio-Temporal Neural Network), which utilizes deep neural networks to jointly predict travel distance and time. The ST-NN model shows improved generalization compared to other existing methods, reducing mean absolute error by approximately 17% for travel time prediction. Furthermore, ST-NN demonstrates increased robustness to outliers in the dataset, making it a promising approach for accurate and efficient transportation management. ST-NN model is rather simple. Thus, it might be interesting to see whether a more complex model shows better performance. In this regards, [9] introduces a novel travel time estimation framework that combines transformer and convolutional neural networks (CNN) to enhance accuracy. The proposed framework includes a traffic information fusion component,

TABLE 1. Summary Of Related Works.

Method	Features
Overhead based road distance estimation [3]	Use an overhead graph. Good to reduce the processing time for optimization process.
Detour Index (DI) based method [4]	Simple linear regression method. Good estimation results with very high coefficient of determination $r^2 > 0.9$
Circuity factor based method [11]	Basically the circuity factor is DI. It shows various circuity factors derived from a sampling of road networks in different countries and regions worldwide.
Weighted l_p or $l_{k,p}$ norms [6]	Modeling travel distances in road networks using the weighted l_p or $l_{k,p}$ norms
Green time-dependent vehicle routing [12]	Focuses on the importance of accurate travel-time estimation in a green time-dependent vehicle routing optimization problem to ensure precise objective function values and accurate decision-making
Large-scale trajectory data [13]	Explores the travel time estimation problem using large-scale trajectory data, particularly New York City Taxi trips dataset.
Deep learning + historical data [13]	Integrates road networks and historical data, enhancing performance, particularly with smaller training sets.
DE-SLSTM [14]	Integrates weather effects to address prediction bias during congestion
osrmtime [15]	A platform-independent method for calculating travel time and distance between two points using geographic data on latitudes and longitudes.
Rapidex [7]	A novel tool for origin-destination (OD) demand estimation and visualization
ST-NN based deep learning method [8]	Utilizes deep neural networks to jointly predict travel distance and time.
CNN based deep learning method [9]	Includes a traffic information fusion component, incorporating GPS trajectory, real road network, and external attributes for comprehensive estimation.

incorporating GPS trajectory, real road network, and external attributes for comprehensive estimation. Additionally, a multiview CNN transformer component captures spatial information at multiple regional scales. Experiments using Chengdu and Beijing datasets demonstrate competitive mean absolute percent errors (MAPE) of 11.25% and 11.78%, outperforming state-of-the-art baselines in travel time estimation. Table 1 summarizes the related works.

While existing approaches offer sufficient accuracy in estimating travel distances and times, our paper shifts its focus to a specific question: the suitability of the neural network-based approach for various types of distances. Instead of solely focusing on accuracy, our objective is to investigate the specific domains in which this neural network based approach demonstrates superior performance. By doing so, we aim to identify the key characteristics of the road data set that play a crucial role in its success. Through this exploration, we seek to gain valuable insights into the important factors that contribute to the effectiveness of the new approach in specific scenarios.

III. ANALYSIS METHODOLOGY OF DNN BASED DISTANCE ESTIMATION

In this section, we delve into the specifics of our proposed analysis methodology. First, we describe the deep neural network (DNN) based road distance estimation method that We want to analyze. Subsequently, we detail the five distance models with different characteristics. Lastly, we elucidate the key performance metric that we aim to focus in the evaluation. These metric serves as the benchmark against which the effectiveness and practicality of the DNN based method is measured.

A. DEEP NEURAL NETWORKS WITH 3 HIDDEN LAYERS

It is well known that DNN can be applied to a very diverse range of problems, but sometimes it is not easy to interpret why it works [18], [19], [20], [21], [22]. In this paper, our primary focus is not on explaining why DNN works, but rather on acknowledging the potential of DNN to capture the

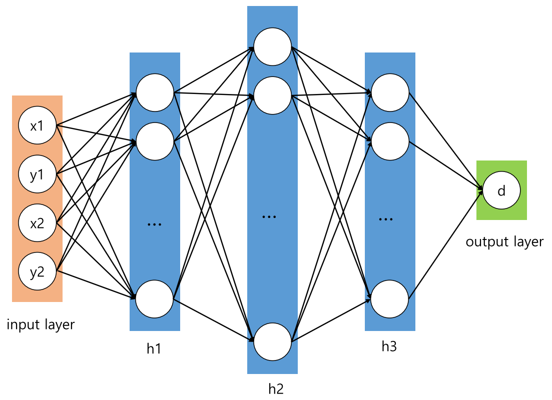


FIGURE 1. Five layer deep neural network model. h1 has 50, h2 has 100, and h3 has 50 perceptrons.

underlying characteristics of the problem. While we may not delve into the specific reasons for its success, we recognize that DNN has the capability to effectively learn and adapt to the intricate patterns present in the data. By acknowledging this ability, we can explore how DNN-based methods can be harnessed to tackle the challenges posed by road distance estimation problem and achieve promising results.

In this road distance estimation problem, given specific input, the model's task is to learn a mapping from the coordinates of the two points (source and destination) to the actual road distance. In this paper, we rather focus on a simple feed-forward deep neural network model structured as in Fig. 1. [19] and [23] provide detailed information about deep neural network model and feed-forward model. It should be noted that this kind of simple feed-forward deep neural network model has been used for various purposes in many other papers such as [8], [24], [25], [26], and [27]. In summary, in this paper, we investigate how the simple feed-forward deep neural network model works for different distance definitions. We explain the model in more detail as follows.

The input layer consists of 4 neurons corresponding to the latitudes and longitudes of the two points. The model has

3 hidden layers. These layers are fully connected (dense) layers. The first layer has 50 perceptrons, the second hidden layer has 100 perceptrons, and the third hidden layer has 50 perceptrons. Each perceptron has a non-linear activation function, ReLU (Rectified Linear Unit) [28]. The output layer consists of one neuron, which gives the estimated road distance. Since our objective is to solve a regression problem, the activation function used for the output layer is *linear*.

As a loss function, we use SmoothL1Loss() [29]. SmoothL1Loss() calculates basically the differences between the predicted and actual road distances. However, depending on a parameter β, there is small modification. Here is the definition of SmoothL1Loss. Let $L = \{l_1, \ldots, l_N\}^T$ be the loss vector for N samples. Each l_i is defined as follows.

$$l_i = \begin{cases} 0.5(x_i - y_i)^2/\beta, & \text{if } |x_i - y_i| < \beta \\ |x_i - y_i| - 0.5 \times \beta, & \text{otherwise} \end{cases} \tag{1}$$

Furthermore, we use *mean* as the reduction function so that we want to minimize the mean of the smooth l1 loss of each sample. Thus, the smooth l1 loss is finally defined as $l(x, y) = mean(L)$. We use $\beta = 1.0$ for the training.

In the model, Fig. 1, the number of weights is $10,250 (= 4 \times 50 + 50 \times 100 + 100 \times 50 + 50)$. There is no clear rule for the size of training data set. However, in this paper, we provide more data samples than the number of weights. During the training phase of the deep neural network model, the model learns to accurately estimate road distances by iteratively adjusting its internal parameters. The training process begins by feeding the model with a large dataset consisting of pairs of input data, which includes latitude and longitude coordinates of two positions, along with their corresponding road distances. The model then goes through a series of forward and backward passes, known as epochs, where it computes predicted road distances for the given inputs and compares them to the actual distances. By utilizing the smooth l1 loss function, the model quantifies the discrepancy between its predictions and the ground truth values. Through a process called back propagation, the model updates its internal parameters in a way that minimizes the loss, gradually improving its estimation capabilities. This process continues for multiple epochs, allowing the model to fine-tune its weights and biases, ultimately enhancing its ability to accurately estimate road distances between positions. For this optimization process, we use Adam optimization algorithm [30].

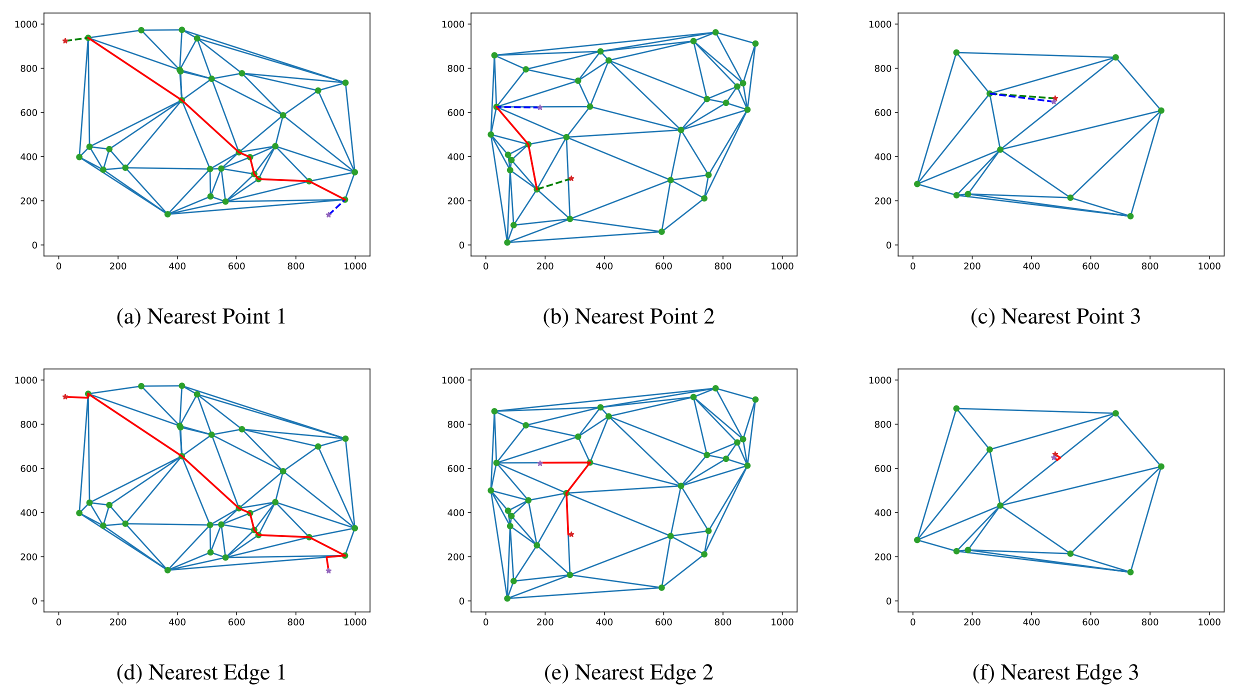
B. CONTROLLED DISTANCE DEFINITIONS

The primary objective of this paper is to thoroughly analyze a deep learning-based road network distance estimation method under controlled conditions. To achieve this, we define five distinct distance types, each characterized by specific attributes. Essentially, the focus of this study lies in exploring the relationship between the characteristics of the distance dataset and the performance of the model. As such, controlled datasets are utilized to provide a more controlled environment for the analysis, rather than relying solely on datasets collected from real-world scenarios. The five distance definitions are described as follows. Let $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ be the positions of two points.

- Euclidean distance (*D1*): $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$
- Manhattan distance (*D2*): $|x_1 - x_2| + |y_1 - y_2|$.
- Geodesic distance (*D3*): The distance between two nodes is defined as the distance on the globe [31].
- Delaunay triangulation with nearest point (*D4*): We first generate k random landmarks in an area and conduct the Delaunay triangulation among the landmarks. The distance between two points, p_1 and p_2 is defined as the Euclidean distance from p_1 to the nearest landmark of p_1 + the distance from p_2 to the nearest landmark of p_2 + the shortest distance over the Delaunay triangulated graph between the two landmarks. Fig. 2 shows some examples of such distances with title "Nearest Point".
- Delaunay triangulation with nearest edge (*D5*): We construct the Delaunay triangulation graph as before. The distance between two points, p_1 and p_2 is defined as the Euclidean distance from p_1 to the nearest *edge* of p_1 + the distance from p_2 to the nearest *edge* of p_2 + the shortest distance over the Delaunay triangulated graph between the two *cross* points. Fig. 2 shows some examples of such distances with title "Nearest Edge".

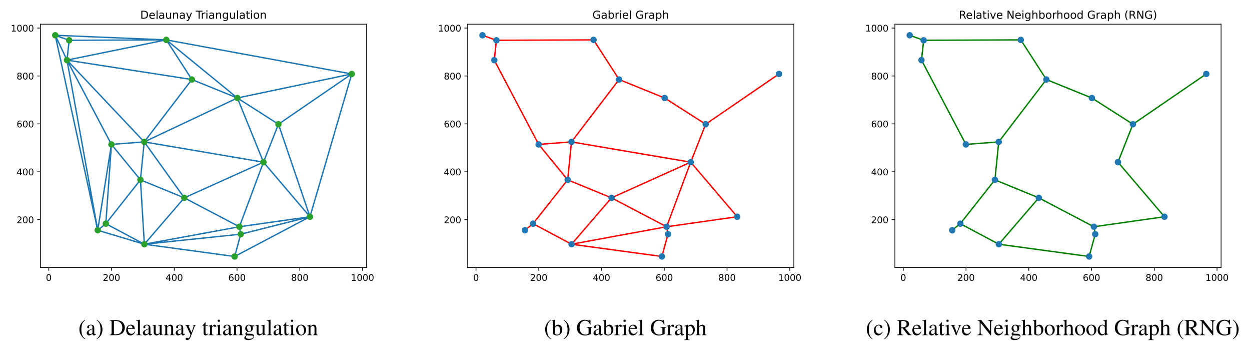
The five different distance definitions presented above exhibit distinct characteristics. Especially, we describe the difference *D4* and *D5* in more detail. Fig. 2a and Fig. 2d show the different distance of the same src-dest pairs in *D4* and *D5*. Similarly, Fig. 2b/Fig. 2e and Fig. 2c/Fig. 2f show another examples. In Fig. 2c and Fig. 2f, we can see a big difference between *D4* and *D5*. All these data sets aim to provide different characteristics of the road networks. It should be noted that the DNN model employed in this paper might display varying performance levels for each of these distance definitions. Thus, our investigation centers around understanding the relationship between the specific characteristics of the distances and the corresponding model performance.

Actually, in our analysis, we have explored various planar graphs, including Gabriel graph and RNG graph, as alternatives to Delaunay triangulation. However, we have chosen Delaunay triangulation due to its ability to provide denser edges compared to the other planar graphs. Fig. 3 shows three different planar graphs: Delaunay triangulation, Gabriel Graph, and Relative Neighborhood Graph for the same set of points. It can be easily seen that the Delaunay triangulation based graph has more edges than other two graphs. In short, in our paper, we opt for the Delaunay triangulation graph because it closely mimics the dense road networks found in developed regions of the world. This choice allows us to analyze the model's performance in a setting that reflects real-world road network characteristics accurately.


(a) Nearest Point 1 (b) Nearest Point 2 (c) Nearest Point 3

(d) Nearest Edge 1 (e) Nearest Edge 2 (f) Nearest Edge 3

FIGURE 2. Delaunay triangulation Example: Three sets of landmarks are chosen for the Delaunay triangulation and for each landmark set, a pair of src-dest are randomly chosen and the different distance computation is shown.



(a) Delaunay triangulation (b) Gabriel Graph (c) Relative Neighborhood Graph (RNG)

FIGURE 3. Examples of planar graphs with the same landmarks: Each shows different set of edges.

C. PERFORMANCE METRIC

For the performance metric of the evaluation, we consider the (absolute) relative error ϵ between estimated road distance $\hat{r}$ and the actual road distance r. The relative error is defined as follows.

$$\epsilon = \frac{|\hat{r} - r|}{r} \tag{2}$$

The relative error indicates the deviation of the estimated distance from the actual distance, taking into account the magnitude of the actual distance. It allows for a fair comparison, as it considers the ratio rather than the absolute difference. For instance, a relative error of 20% for an estimated distance of 120 km and an actual distance of 100 km is equivalent to a relative error of 20% for an estimated distance of 12 km and an actual distance of 10 km. Thus, a relative error near 0 signifies that the estimated distance closely matches the actual distance.

As we mention before, it is known that the inflation ratio of the road distance over the direct distance is about 1.3. The actual ratio may depend on the given data set. This simple estimation using the inflation ratio is basically a linear regression method over the data points of <Euclidean distance (e_d), actual distance(a_d)>. For the some distance definitions, we investigate the performance of this simple linear regression approach. For that matter, we generate data points of (e_d, a_d) for randomly generated src-dest pairs. With the set of such distance pairs, we conduct linear regression and compute the coefficient β and the intercept α. Then, the actual distance estimation a_d of a Euclidean distance e_d is computed as follows.

$$a_d = \beta \times e_d + \alpha \tag{3}$$

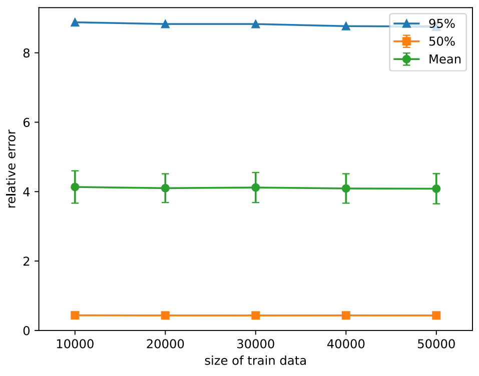
IV. ANALYSIS RESULTS

In this section, we investigate the performance of the DNN based road distance estimation method over five different distance definitions. For the evaluation, we first generate the synthetic distance data sets for the five different distance definitions. Each data set consists of tuples of 5 values: source-longitude, source-latitude, destination-longitude, destination-latitude, and actual distance. For all the distance types except Geodesic distance (D3), the sources and destinations are randomly generated from a 10000×10000 square area. The landmarks for Delaunay triangulation based methods (D4 and D5) are also randomly generated from the same area. The training data sets and test data sets are generated by using the same method.

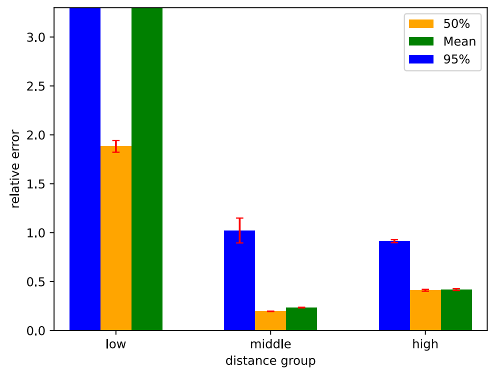
Then, we train the deep neural network with varying sizes of training data sets. We vary the size from 10,000 to 50,000. The smallest value, 10000, has been chosen to be similar to the number of weights of the DNN model in Fig. 1, which is 10,250. If the training set size is too small compared to the number of weights, the model could be over-fitted. Thus, we consider 10,000 could be the minimum possible data set size. Then, we increase the size up to some extents to see the performance over the training set sizes. The size of the test data set is fixed to the smallest training set size, which is 10,000.

We conduct the evaluation 15 times, each time with a different set of randomly selected landmarks and src-dest pairs. This allows us to establish confidence intervals and demonstrate the robustness of the repeated evaluations. We compute the confidence interval via Student's t-distribution, which requires smaller sample sizes compared to the Gaussian distribution. The number of trials or samples, 15, is half of 30, which is considered to be the minimum sample size for Central Limit Theorem. The resulting confidence intervals are small enough to show the behavior of the performance over random situations. During this evaluation, we add the estimation result of the simple linear regression method, which basically exploits DI or circuity factors, which are discussed in Section II. Even though we add linear regression method, it does not mean that the linear regression is the state of the art estimation method till now. Rather we just want to show that the simple linear regression method may not work for some distance definitions, which justifies the necessity of more sophisticated methods. Furthermore, it should be noted again that the main focus of this paper is to show the performance differences of DNN based method for different kinds of distance definitions.

Before investigating the performance of the DNN model, we conduct a training with a very trivial case. We create tuples of 5 randomly generated numbers from a range [1, 9999] and train the model using 4 numbers as input and the remaining 1 number as output. Since the output number is randomly generated, it is theoretically impossible to accurately predict the output from the given input. This simple experiment helps us understand the behavior of the DNN model when faced with an inherently unpredictable scenario. Fig. 4 shows the estimation results of such experiment. Fig. 4a shows the mean, 50th percentile, and 95th percentiles of relative errors over training data set sizes. Fig. 4b shows the error statistics



(a) Overall

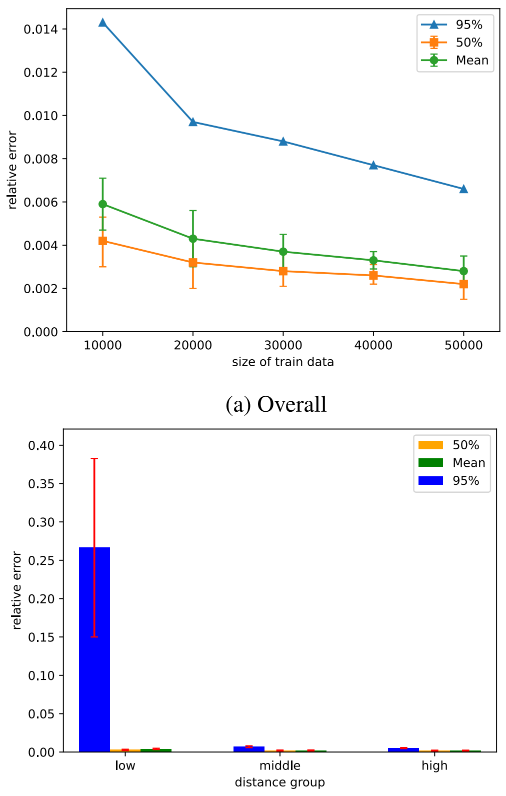


(b) Distance Group

FIGURE 4. Estimation of Randomly Generated Values with DNN.

over difference distance ranges when the training data set size is 50,000. We divide the set of actual distances into 3 groups: low, middle, high. To define the group, we find the maximum of the output values. Then, we create three groups with intervals of one third of the maximum value. Then, we compute the statistics (mean, 50th percentile and and 95th percentile) of the errors in each group. Actually the largest value of the relative errors is much higher than that is shown in Fig. 4b. But to examine smaller values more clearly, we just cut the y-axis of the group statistics to 3.3. As can be seen in Fig. 4, the mean relative errors are about 4.7 and the 50th percentiles are about 0.62. Actually, consider two integer random variables X and Y, which are uniformly distributed over $[1, 9999]$. Consider a new random variable $Z = |X - Y|/X$. The expected value of Z is 4.741 and the median is 0.618 (This can be easily computed by simple nested for loop code.) Thus, it is quite clear that the estimation through the DNN model does not show any accuracy at all. The reason for the high relative errors in the trivial case is that the given data set lacks intrinsic characteristics. However, the five distance definitions exhibit underlying structure, which leads us to anticipate significantly smaller relative errors.

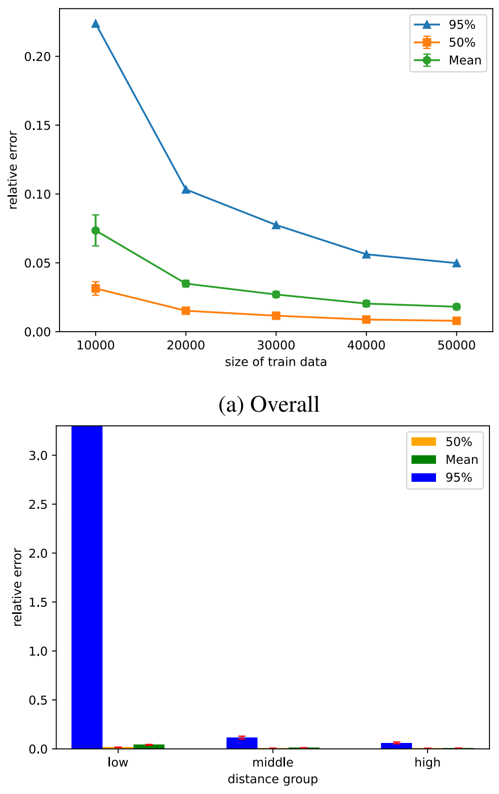
To clearly appreciate the capability of the DNN based method, we, first, investigate the performance for the euclidean data set. Fig. 5a shows the overall accuracy of the method over the size of train data sets in terms of relative errors. As the size increases, the relative error decreases.



(a) Overall

(b) Distance Group

FIGURE 5. Estimation of Euclidean Distances with DNN.



(a) Overall

(b) Distance Group

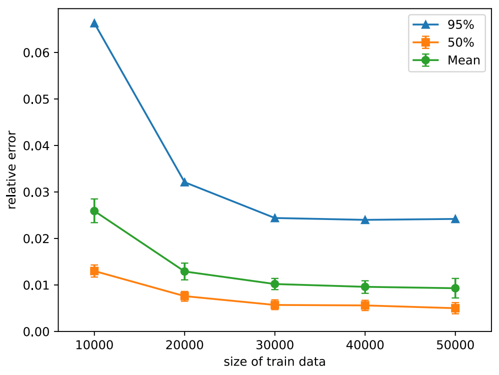
FIGURE 6. Estimation of Geodesic Distances with DNN.

To summarize the accuracy, we show the mean, median (50th percentile), and the 95th percentile errors. For each measurement point, we also show the confidence intervals for mean and median errors. Fig. 5a clearly shows that the errors are very close to 0. This experiment is to show that neural network can be used to effectively mimic a non-linear function such as Euclidean distance. Fig. 5b shows the error statistics over the three distance groups. As can be seen in Fig. 5b, when the actual distance is small, the 95th percentile relative errors are relatively high. It is quite expected due to the definition of relative error. However, for mid and high range actual distances, the relative errors are near 0. In summary, the DNN based method can estimate the Euclidean distance quite accurately.

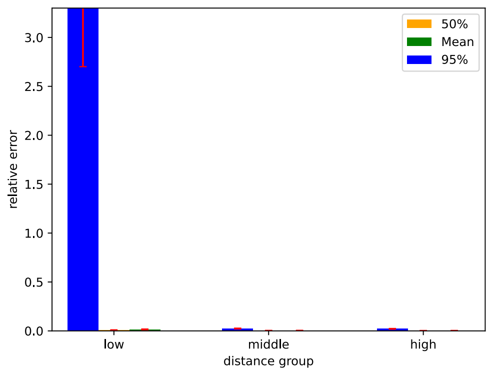
Even though the earth looks flat, it is actually a globe. So to better measure the direct distance between two points, we need to use Geodesic distance instead of Euclidean distance. For that matter, we generate random points in spherical coordinates with radius 1. The latitudes are uniformly randomly generated from $-90°$ to $90°$. The longitude are uniformly randomly generated from $-180°$ to $180°$. It should be noted that through this random selection, the points near poles are more frequently chosen. Fig. 6 shows the estimation performance over the Geodesic distances. As can be seen in Fig. 6, the relative errors are about 10 times higher than the Euclidean distance case, which is quite surprising.

Such high errors is due to the fact that the distances among the points in spherical coordinates does not resemble to Euclidean distances. However, if the area on a sphere is small, then the distances might exhibit Euclidean characteristics. Thus, we randomly generate points in a small area on the sphere. For that matter, we consider two cases. One is the area of mainland USA, where latitude varies 24° to 48° and the longitude varies from $-125°$ to $-67°$. The other is the area of Korean Peninsula. where latitude varies 33° to 43° and the longitude varies from 124° to 132°. Fig. 7 and Fig. 8 show the relative errors for the data set derived from each area. The relative errors over the smaller area are smaller than those of the overall globe. One thing to note is that Fig. 8 shows higher relative errors than Fig. 7. The difference comes from the fact that the Korean Peninsula area is longer in vertical direction, while the mainland USA area is wide in horizontally.

Now, we investigate the performance of the DNN based method for the Manhattan distances. Fig. 9 shows the mean, median, and 95th percentiles for the DNN based method ("-DNN") and the linear regression based method ("-Reg"). The relative errors of the DNN based method are similar to the Euclidean distance case. However, linear regression model does not perform well. That is because the same direct distance may have very different Manhattan distances. For example, let $a = (0, 0)$, $b = (5, 0)$, $c = (3, 0)$, $d(0, 4)$. Then, both $\overline{ab}$ and $\overline{cd}$ are 5 in terms of Euclidean distance. However, in Manhattan distance, $\overline{ab} = 5$ and $\overline{cd} = 7$. Thus, linear regression may not be able to estimate the correct distance solely based on the Euclidean distance between two points.

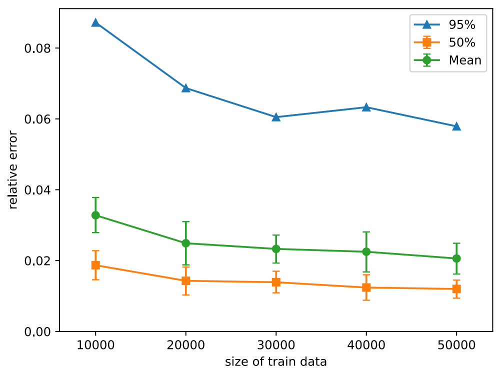


(a) Overall

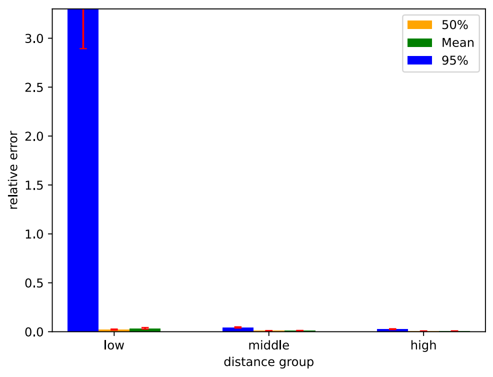


(b) Distance Group

FIGURE 7. Estimation of Geodesic Distances with DNN (Mainland USA).



(a) Overall



(b) Distance Group

FIGURE 8. Estimation of Geodesic Distances with DNN (Korean Peninsula).

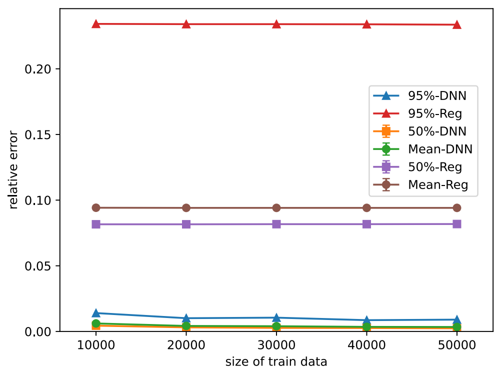
Since the largest error of linear regression method occurs when two points are on the diagonal of a square, the largest relative error could be $(2 - \sqrt{2})/2 = 0.293$. This simple mathematical analysis can be roughly confirmed by the 95th percentile of relative errors of linear regression model for Manhattan distance, a little bit larger than 0.2.

Manhattan distance represents the road network around city centers. However, for a large area, a planar graph may represent better. For that matter, we evaluate the performance of the DNN based approach for distances based on Delaunay triangulation graphs (*D4* and *D5*). Fig. 10 shows the relative errors for the distances of Delaunay triangulation with Nearest Point (*D4*). The number of nodes for the Delaunay triangulation is 30. As can be seen in Fig. 10, the mean and media relative errors of the DNN based method is less than 0.2. The 95th percentiles are also below 0.5. However, these errors are much higher compared to the Euclidean distance and Manhattan distance cases. The reason is that for the distances of the Delaunay triangulation with nearest point, there are some cases where small difference in the position may occur large actual distance. However, for Euclidean distance and Manhattan distance, the difference is usually proportional to the offset of the positions.

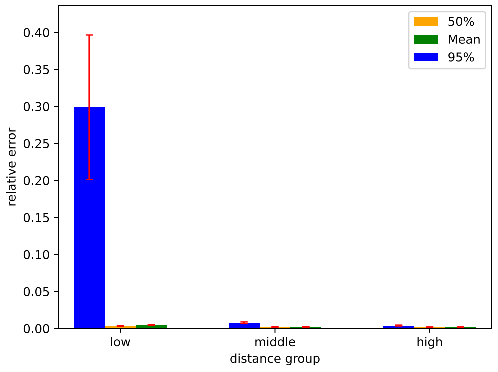
For example, let $\vec{p}$ and $\vec{q}$ be the positions of two nodes. Let $\vec{q'} = \vec{q}+\vec{\delta}$, which is a point close to $\vec{q}$. The Euclidean distance of $\vec{p}$ and $\vec{q}$ is $d_1 = \|\vec{q}-\vec{p}\|$. The Euclidean distance of $\vec{p}$ and $\vec{q'}$ is $d_2 = \|\vec{q} - \vec{p} + \vec{\delta}\|$. The distance difference between d_1 and d_2 is relatively proportional to $\|\vec{\delta}\|$. In other words, there is no large distance difference for a small change of positions,$\vec{\delta}$. However, for the distances of Delaunay triangulation with nearest points, there can be many cases where small position changes cause large distance differences. That is because two nearby points may have different nearest landmarks in the Delaunay triangulation. A specific example is given in the next section. Nevertheless, the linear regression based method shows a little bit higher relative errors than the DNN based method.

One thing to note is that the relative errors do not change much as the training data size increases. The reason is because the whole distances are based on the 30 node Delaunay triangulation. Thus, 10,000 samples can effectively represent the characteristics of the distances. Thus, increasing the number of training samples may not provide more information to the neural network. In other words, the neural network can summarize the distances among the landmarks easily with 10,000 samples. However, it is quite difficult to estimate the distances between src-dest pairs because of the non-linear distance characteristics. Thus, there are large relative errors compared to the Euclidean and Manhattan distances.

Similar phenomenon happens for the distances of Delaunay triangulation with nearest edges (*D5*). Fig. 11 shows the relative errors of the DNN based approach and the linear

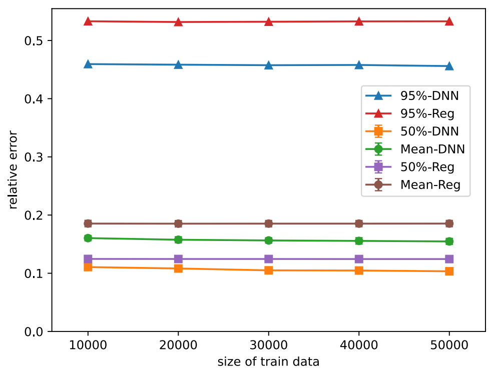


(a) Overall

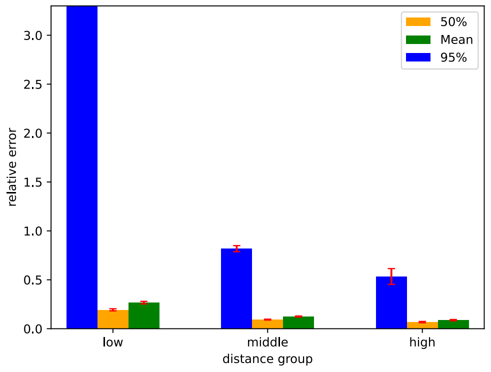


(b) Distance Group

FIGURE 9. **Estimation of Manhattan Distances with DNN.**



(a) Overall



(b) Distance Group

FIGURE 10. **Estimation of Distances by Delaunay triangulation with Nearest Poin.**

regression based approach. We can clearly see that the DNN based approach performs better than linear regression based approach.

Finally, we show the correlation coefficient and the coefficient of determination of the estimation results in Table 2. For each r and r^2, we show the mean and 95th confidence interval from the 15 runs of simulations respectively. Thus, for the Delaunay triangulation with Nearest Point, the mean of r is 0.9064376 and the 95th confidence interval is the mean ± 0.0083036. As can be easily expected from the above results, r and r^2 values are close to 1.0 for Euclidean distance and Manhattan distance. However, r values of Delaunay triangulation based approaches are a little bit row, close to 0.9. Nonetheless, the DNN based approach shows good estimation accuracy. It should be noted that r and r^2 of the random data set are almost 0, which means that the estimation is totally random.

V. IMPACT OF DISCONTINUITY

Carefully analyzing the results in Section IV, we conclude that the difference in performance comes from the degree of how much output change is incurred by the input change. To be specific, we expect that if the input change is small, the output change is small. Similarly, if the input change is large, the output should change large. Since DNN can be considered as a sort of linear regression model, if the data set shows this kind of linear (continuous) behavior, the DNN model may work well. However, if the data set does not show continuous behavior, DNN may not perform well.

The distance types that the DNN model works best are the Euclidean distance and the Manhattan distance. In the two cases, the continuity exhibits highly. However, for other distance types, there are many cases where the distances are not continuous. For example, consider the example in Table 3. There are two inputs and two outputs for the three different distance types. For the Euclidean and Manhattan distances, the two outputs differ for different inputs. However, the geodesic distance type shows the same output due to the characteristics of the sphere. To be specific, the Input 1 represents two points: (lat$=0°$, lon$=0°$) and (lat$=90°$, lon$=0°$). Similarly, the Input 2 represents two points: (lat$=0°$, lon$=0°$) and (lat$=90°$, lon$=180°$). However, the geodesic distances of those two Inputs are the same, i.e., $\pi/2$ because (lat$=90°$, lon$=0°$) and (lat$=90°$, lon$=180°$) are the same points.

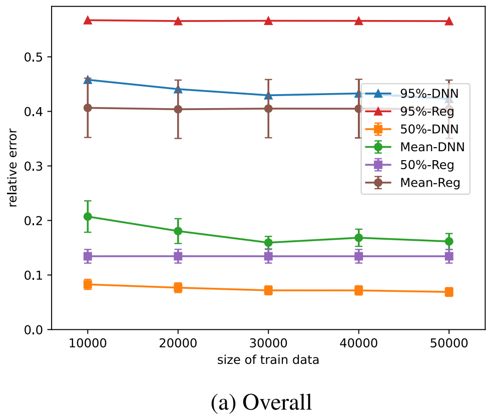
Similarly, the Delaunay triangulation based distances show similar discontinuity. Consider Fig. 12. The Delaunay triangulation has three points at (0, 0), (6, 0), (0, 6). There are three other points at (4, 5), (5, 4), (6, 1). The distance between (4, 5) and (6, 1) is $\sqrt{17}+6\sqrt{2}+1 \approx 13.6$. However, the distance between (5, 4) and (6, 1) is $\sqrt{17}+1 \approx 5.1$. This is because the nearest points in Delaunay triangulation of the two points, (4, 5) and (5, 4), are different even though the two points are close.

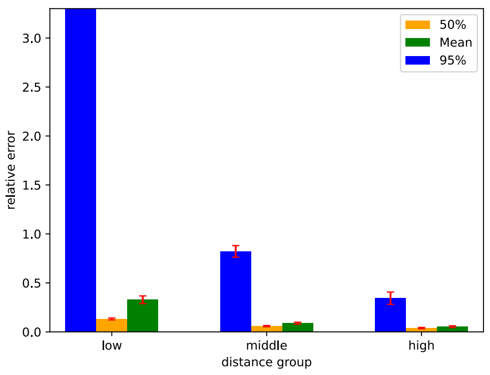
TABLE 2. Accuracy of Distance Estimation: r and r^2.

Distance Type	r	r^2
Random	-0.0016632 ± 0.0061223	0.0001250 ± 0.0000685
Euclidean Distance	0.9999911 ± 0.0000030	0.9999821 ± 0.0000061
Manhattan Distance	0.9999878 ± 0.0000035	0.9999757 ± 0.0000070
Geodesic Distance	0.9995285 ± 0.0001215	0.9990572 ± 0.0002429
Geodesic Distance (Korea)	0.9996399 ± 0.0001090	0.9992800 ± 0.0002179
Geodesic Distance (USA)	0.9999511 ± 0.0000094	0.9999021 ± 0.0000188
Delaunay triangulation with Nearest Edge	0.9546766 ± 0.0101681	0.9117445 ± 0.0192625
Delaunay triangulation with Nearest Point	0.9064376 ± 0.0083036	0.8218540 ± 0.0149508

TABLE 3. Examples of discontinuity in distances.

Distance type	Input 1	Input 2	Output 1	Output 2
Euclidean	(0, 0, 90, 0)	(0, 0, 90, 180)	90	$201.2 = \sqrt{90^2 + 180^2}$
Manhattan	(0, 0, 90, 0)	(0, 0, 90, 180)	90	270
Geodesic ($r = 1$)	(0, 0, 90, 0)	(0, 0, 90, 180)	$\pi/2$	$\pi/2$





(b) Distance Group

FIGURE 11. Estimation of Distances by Delaunay triangulation with Nearest Edge.

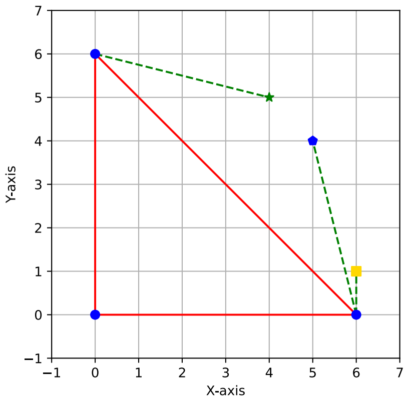


FIGURE 12. Example of Delaunay triangulation with nearest points: Two close points have very different distance to a third point.

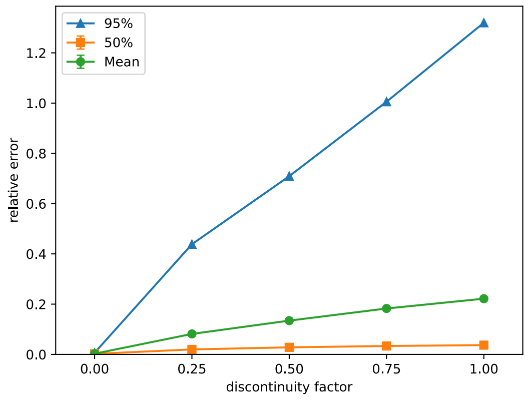
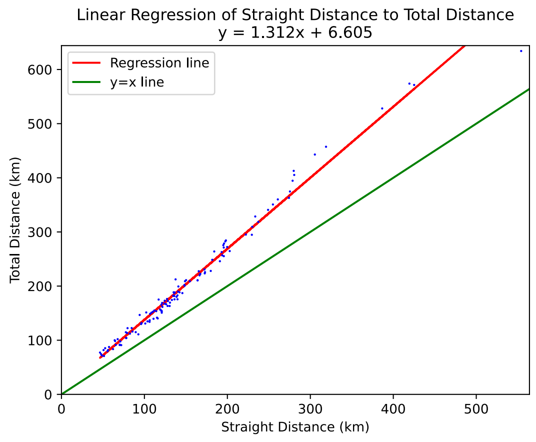


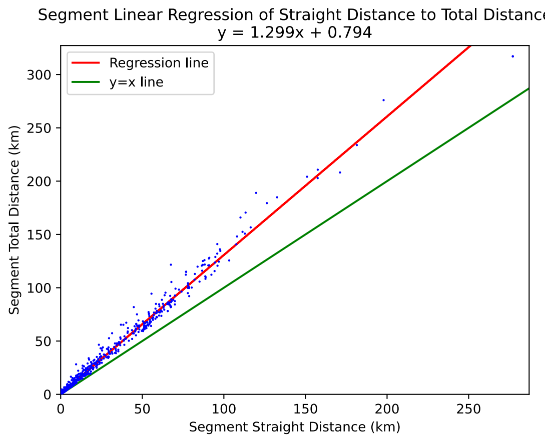
FIGURE 13. Impact of the discontinuity factor in a Euclidean space. Larger discontinuity factor generates poor performance.

To analyze the impact of discontinuity of the distance, we generate distance data sets on an Euclidean space with some discontinuity factor. To be specific, we divide a rectangular evenly along one axis. Then we select two points on the area and compute the Euclidean distance. To apply the discontinuity factor p, if the two points reside in different ''half'' of the area, we add additional distance, $p \times$ width. Thus, a slight change of a position may result in a big distance change. We vary p with 0, 0.25, 0.5, 0.75, 1.0 and check the performance. Since we only need to show the clear relationship between the errors and p, we vary p from 0 up to only 1. As can be seen in Fig. 13, a larger discontinuity factor shows higher relative errors. Thus, if the driving distances show high discontinuity, then the deep learning based method may not work well. One thing to note is that in this paper, we only consider synthetic data sets, which are generated from distance definitions. However, for the real road information, OpenStreetMap [17] could be a viable data set.

Actually, the continuity characteristics of the road distance have been reported in many papers [4], [5], [11]. We also



(a) Distances of a route



(b) Distances of each segment of a route

FIGURE 14. Geographic distance vs. Travel Distance: The slope of the linear regression method shows the inflation ratio of around 1.3.

examine the continuity of the geographic distances in real environment. We collect 123 truck routes from a drug delivery services in Korea. The total number of destinations in the routes is 1090. Thus, in each route, there are in average 8.9 destinations. Fig. 14 shows the travel distance and the geographic (straight) distance pairs. Specifically, Fig. 14a shows the overall distribution of distances of routes and Fig. 14b shows the distributions of distances of each segment in a route. The evident slope of approximately 1.3 in the plot confirms the consistent findings of previous studies. This clear continuity suggests the promising potential of DNN models for accurately estimating travel distances. The correlation coefficient of the whole travel distances is $r = 0.9916$ and the coefficient of determination is $r^2 = 0.9833$. Similarly, the correlation coefficient of the segment distances is $r = 0.9945$ and the coefficient of determination is $r^2 = 0.9891$. It means that the deep learning based method might be a promising method for the driving distance estimation.

Finally, we mention the estimation time of DNN based approach. For some applications like VRP (Vehicular Routing Problem), the number of distance estimations may be very large. For example, the number of distance estimations needed for an n node VRP problem, is $n(n-1)/2$. Thus, the computation time should be considered as a serious factor for such application. The computation of the DNN model depends on the number of weights and the number of perceptrons. In the model of this paper, the number of weights is 10, 250. Furthermore the number of perceptrons is 200, for which the ReLu operation is needed. However the ReLu operation is just a simple comparison and assignment, the number of weights is the main factor for the computation time. Ten thousands multiplication does not take much time for modern computers, so the computation time of the estimation procedure in DNN based approach is not a big problem.

Actually, we measure the execution time of the estimation in a Windows machine (Inter(R) Core(TM) i5-9500F CPU @ 3.00GHz, 32.0GB RAM, 64bit operating system). We conduct 75 runs of 10,000 estimations. The average computation time of each run is 2.818706667 seconds, which means 0.281871*msec* for each estimation. Thus, for a 100 node VRP, the computation time would be less than 2 seconds.

VI. CONCLUSION

The road distances play a crucial role in many location-based applications, particularly in the context of the vehicular routing problem. However, traditional methods relying solely on direct geographic distances calculated by latitude and longitude fail to accurately represent road distances on real road networks. Estimating road distances is therefore paramount for the success of such services. Previous research has focused on developing efficient methods for estimating road distances and has reported an inflation ratio of approximately 1.3 between road distances and direct distances. Furthermore there are DNN based approaches to improve the accuracy of the estimation.

In this paper, the main focus is on thoroughly examining a deep learning-based road network distance estimation method under controlled conditions. Five different distance types with specific characteristics are defined, and distance data sets based on these types are generated to evaluate the DNN-based approach. The goal is to better understand the behavior of the DNN-based approach and identify the main factors influencing its performance. Extensive simulations show that the DNN-based method performs well across various distance definitions. By carefully analyzing the evaluation results, we find that the discontinuity in the distances is a critical determinant of its performance. Consequently, future DNN-based road network distance estimation methods should carefully consider the discontinuity to optimize overall performance and develop more accurate and efficient distance estimation models for complex modern road networks.

REFERENCES

[1] H. Huang, G. Gartner, J. M. Krisp, M. Raubal, and N. Van de Weghe, "Location based services: Ongoing evolution and research agenda," *J. Location Based Services*, vol. 12, no. 2, pp. 63–93, Apr. 2018, doi: 10.1080/17489725.2018.1508763.

[2] K. Braekers, K. Ramaekers, and I. Van Nieuwenhuyse, ''The vehicle routing problem: State of the art classification and review,'' *Comput. Ind. Eng.*, vol. 99, pp. 300–313, Sep. 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360835215004775

[3] R. Mariescu-Istodor and P. Fränti, ''Fast travel-distance estimation using overhead graph,'' *J. Location Based Services*, vol. 15, no. 4, pp. 261–279, Oct. 2021, doi: 10.1080/17489725.2021.1889058.

[4] X. Chen and Y. Chen, ''Quantifying the relationships between network distance and straight-line distance: Applications in spatial bias correction,'' *Ann. GIS*, vol. 27, no. 4, pp. 351–369, Oct. 2021, doi: 10.1080/19475683.2021.1966503.

[5] F. P. Boscoe, K. A. Henry, and M. S. Zdeb, ''A nationwide comparison of driving distance versus straight-line distance to hospitals,'' *Prof. Geographer*, vol. 64, no. 2, pp. 188–196, May 2012, doi: 10.1080/00330124.2011.583586.

[6] J. Brimberg, J. H. Walker, and R. F. Love, ''Estimation of travel distances with the weighted l_p norm: Some empirical results,'' *J. Transp. Geogr.*, vol. 15, no. 1, pp. 62–72, Jan. 2007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0966692306000044

[7] S. T. Waller, S. Chand, A. Zlojutro, D. Nair, C. Niu, J. Wang, X. Zhang, and V. V. Dixit, ''Rapidex: A novel tool to estimate origin–destination trips using pervasive traffic data,'' *Sustainability*, vol. 13, no. 20, p. 11171, Oct. 2021. [Online]. Available: https://www.mdpi.com/2071-1050/13/20/11171

[8] I. Jindal, Tony, Qin, X. Chen, M. Nokleby, and J. Ye, ''A unified neural network approach for estimating travel time and distance for a taxi trip,'' Oct. 2017, *arXiv:1710.04350*.

[9] F. Liu, J. Yang, M. Li, and K. Wang, ''MCT-TTE: Travel time estimation based on transformer and convolution neural networks,'' *Sci. Program.*, vol. 2022, pp. 1–13, Apr. 2022.

[10] J. Moon and S. Lee, ''Analysis of deep neural network based road distance estimation,'' in *Proc. 38th Int. Conf. Inf. Netw. (ICOIN)*, Ho Chi Minh City, Vietnam, Jan. 2024, pp. 640–643.

[11] R. H. Ballou, H. Rahardja, and N. Sakai, ''Selected country circuity factors for road travel distance estimation,'' *Transp. Res. A, Policy Pract.*, vol. 36, no. 9, pp. 843–848, Nov. 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0965856401000441

[12] E. Asgharizadeh, S. Jooybar, H. A. Mahdiraji, and J. A. Garza-Reyes, ''A novel travel time estimation model for modeling a green time-dependent vehicle routing problem in food supply chain,'' *Sustainability*, vol. 14, no. 14, p. 8633, Jul. 2022. [Online]. Available: https://www.mdpi.com/2071-1050/14/14/8633

[13] H. Wang, X. Tang, Y.-H. Kuo, D. Kifer, and Z. Li, ''A simple baseline for travel time estimation using large-scale trip data,'' *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–22, Jan. 2019, doi: 10.1145/3293317.

[14] C.-H. Chou, Y. Huang, C.-Y. Huang, and V. S. Tseng, ''Long-term traffic time prediction using deep learning with integration of weather effect,'' in *Proc. 23rd Pacific-Asia Conf. Adv. Knowl. Discovery Data Mining (PAKDD)*, Macau. Berlin, Germany: Springer-Verlag, Apr. 2019, pp. 123–135, doi: 10.1007/978-3-030-16145-3_10.

[15] S. Huber and C. Rust, ''Calculate travel time and distance with openstreetmap data using the open source routing machine (OSRM),'' *Stata J., Promoting Commun. Statist. Stata*, vol. 16, no. 2, pp. 416–423, Jun. 2016, doi: 10.1177/1536867x1601600209.

[16] S. Das, R. N. Kalava, K. K. Kumar, A. Kandregula, K. Suhaas, S. Bhattacharya, and N. Ganguly, ''Map enhanced route travel time prediction using deep neural networks,'' Nov. 2019, *arXiv:1911.02623*.

[17] *Open Street Map*. Accessed: Jun. 4, 2024. [Online]. Available: https://www.openstreetmap.org/

[18] W. Samek, G. Montavon, S. Lapuschkin, C. J. Anders, and K.-R. Müller, ''Explaining deep neural networks and beyond: A review of methods and applications,'' *Proc. IEEE*, vol. 109, no. 3, pp. 247–278, Mar. 2021.

[19] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, ''Efficient processing of deep neural networks: A tutorial and survey,'' *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.

[20] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, ''Backpropagation applied to handwritten zip code recognition,'' *Neural Comput.*, vol. 1, no. 4, pp. 541–551, Dec. 1989.

[21] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, ''Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,'' *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.

[22] G. Montavon, W. Samek, and K.-R. Müller, ''Methods for interpreting and understanding deep neural networks,'' *Digit. Signal Process.*, vol. 73, pp. 1–15, Feb. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1051200417302385

[23] S. Razavi and B. A. Tolson, ''A new formulation for feedforward neural networks,'' *IEEE Trans. Neural Netw.*, vol. 22, no. 10, pp. 1588–1598, Oct. 2011.

[24] P. G. Benardos and G.-C. Vosniakos, ''Optimizing feedforward artificial neural network architecture,'' *Eng. Appl. Artif. Intell.*, vol. 20, no. 3, pp. 365–382, Apr. 2007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197606001072

[25] M. Y. El-Bakry, ''Feed forward neural networks modeling for K–P interactions,'' *Chaos, Solitons Fractals*, vol. 18, no. 5, pp. 995–1000, Dec. 2003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0960077903000687

[26] E. Ozanich, P. Gerstoft, and H. Niu, ''A feedforward neural network for direction-of-arrival estimation,'' *J. Acoust. Soc. Amer.*, vol. 147, no. 3, pp. 2035–2048, Mar. 2020, doi: 10.1121/10.0000944.

[27] J. Khan, E. Lee, and K. Kim, ''A higher prediction accuracy–based alpha–beta filter algorithm using the feedforward artificial neural network,'' *CAAI Trans. Intell. Technol.*, vol. 8, no. 4, pp. 1124–1139, Dec. 2023. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/cit2.12148

[28] *Relu*. Accessed: Jun. 4, 2024. [Online]. Available: https://pytorch.org/docs/stable/generated/torch.nn.ReLU.html

[29] *SmoothL1Loss*. Accessed: Jun. 4, 2024. [Online]. Available: https://pytorch.org/docs/stable/generated/torch.nn.SmoothL1Loss.html

[30] D. P. Kingma and J. Ba, ''Adam: A method for stochastic optimization,'' Jan. 2017, *arXiv:1412.6980*.

[31] *Geopy*. Accessed: Jun. 4, 2024. [Online]. Available: https://geopy.readthedocs.io/en/stable/

SANGHWAN LEE received the B.S. and M.S. degrees from Seoul National University, South Korea, in 1993 and 1995, respectively, and the Ph.D. degree from the Department of Computer Science and Engineering, University of Minnesota, in September 2005. After the M.S. degree, he was with Hyundai Electronics, South Korea, for five years. From June 2005 to February 2006, he was with IBM T. J. Watson Research, then he joined Kookmin University, Seoul, South Korea, in March 2006. His main research interests include software defined networking (SDN), scalable routing selection for multimedia, and various theories and services needed on the internet.

JINSOO MOON received the B.S. degree in electronic engineering from Seoul National University of Science and Technology, in 1993, and the M.S. degree from the University of Pennsylvania, in 1996. After the M.S. degree, he joined Hyundai Electronics and participated in the development of CDMA systems, in 1996, and later worked in mobile s/w development at Motorola, in 2001. He is the CEO of m2Cloud, a company providing SCM services in the bio-pharma industry in South Korea. Prior to founding m2Cloud, he was the JAVA Engineer with Oracle and Sun Microsystems, where he worked on JAVA VM development and JAVA engineering service, from December 2004 to December 2014. At m2Cloud, he has been leading the implementation of COVID-19 vaccine SCM system in South Korea, since 2021, and applying deep learning technology to improve operational efficiency.