International Journal of Geo-Information

MDPI

Article

Ndist2vec: Node with Landmark and New Distance to Vector Method for Predicting Shortest Path Distance along Road Networks

Xu Chen [1], Shaohua Wang [2,3,4,*], Huilai Li [5], Fangzheng Lyu [6], Haojian Liang [1], Xueyan Zhang [7] and Yang Zhong [8]

1 School of Artificial Intelligence, Jilin University, Changchun 130015, China
2 International Research Center of Big Data for Sustainable Development Goals, Beijing 100094, China
3 State Key Laboratory of Remote Sensing Science, Aerospace Information Research Institute, Chinese Academy of Sciences, Beijing 100094, China
4 Key Laboratory of Digital Earth Science, Aerospace Information Research Institute, Chinese Academy of Sciences, Beijing 100094, China
5 School of Mathematics, Jilin University, Changchun 130015, China
6 Department of Geography and Geographic Information Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA
7 School of Letters and Science, University of California, Santa Barbara, CA 93106, USA
8 School of Information Systems and Technology, Claremont Graduate University, Claremont, CA 91711, USA
* Correspondence: wangshaohua@aircas.ac.cn; Tel.: +86-010-8217-8178

Abstract: The ability to quickly calculate or query the shortest path distance between nodes on a road network is essential for many real-world applications. However, the traditional graph traversal shortest path algorithm methods, such as Dijkstra and Floyd–Warshall, cannot be extended to large-scale road networks, or the traversal speed on large-scale networks is very slow, which is computational and memory intensive. Therefore, researchers have developed many approximate methods, such as the landmark method and the embedding method, to speed up the processing time of graphs and the shortest path query. This study proposes a new method based on landmarks and embedding technology, and it proposes a multilayer neural network model to solve this problem. On the one hand, we generate distance-preserving embedding for each node, and on the other hand, we predict the shortest path distance between two nodes of a given embedment. Our approach significantly reduces training time costs and is able to approximate the real distance with a relatively low Mean Absolute Error (MAE). The experimental results on a real road network confirm these advantages.

Keywords: road networks; shortest path distance; landmarks; graph embedding; neural networks

Citation: Chen, X.; Wang, S.; Li, H.; Lyu, F.; Liang, H.; Zhang, X.; Zhong, Y. Ndist2vec: Node with Landmark and New Distance to Vector Method for Predicting Shortest Path Distance along Road Networks. *ISPRS Int. J. Geo-Inf.* **2022**, *11*, 514. https://doi.org/10.3390/ijgi11100514

Academic Editors: Wolfgang Kainz and Sisi Zlatanova

Received: 11 July 2022
Accepted: 4 October 2022
Published: 9 October 2022

1. Introduction

In the context of a large-scale road network [1,2] with a large number of users sending out remote distance queries at the same time, determining how to provide a timely response to such a large number of queries is a very important research question for many navigation applications. The aforementioned problem of efficiently and accurately predicting the shortest path distance between nodes in a road network [3,4] has attracted researchers' attention, and a number of exact methods [5–9] capable of performing error-free distance prediction have been proposed, as well as approximate methods [10–14] that sacrifice some prediction accuracy to reduce computation and memory costs.

The traditional Dijkstra algorithm [5] has a time complexity of $O(nlogn + m)$ and a space complexity of $O(n)$, using big O notation [15], where m and n are the number of edges and the number of nodes in the graph, respectively. Moreover, the Floyd–Warshall [6] algorithm has a time complexity of $O(n^3)$ and a space complexity of $O(n^2)$. Such a time complexity is acceptable for small graphs, but for large million-node graphs, the calculation

of a single node distance requires massive computational resources and time [16]. In order to speed up the query times compared with traditional methods, a number of labeling methods have been proposed [7,8,17], all of which use distance labels. The basic idea is to calculate the distance from each node to other nodes (in extreme cases, this may be all the remaining nodes) in advance in the data preprocessing stage in order to form a tuple as the distance label of the node. Then, by checking the distance label, the distance between any two nodes can be calculated in $O(1)$ time. However, the difficulty of these labeling methods is to find the minimum node set that needs to calculate and store the distance so as to accurately calculate all the shortest paths. Finding the optimal node set of a graph has been proved to be an NP-hard problem. At the same time, the memory cost consumed by these methods is still $O(n^2)$ [18].

In order to reduce the memory cost, researchers proposed approximate shortest path distance methods [12,13], which further reduce the memory and computation costs by sacrificing some prediction accuracy. The sacrifice is worthwhile, because in many practical applications, or some special graphs (large road network graphs), if the exact distance is not necessary, it is enough to find the approximate distance between nodes. A typical representative of the approximate shortest path distance method is the landmark-based method [16,19–21]. This method usually selects l nodes as landmarks, and then, similar to the marking method, assigns a distance label to each node, which contains the distance from the node to these landmark nodes. When querying the distance between any two nodes, the approximation is the sum of the minimum distances between these two nodes and the same landmark node. Although the landmark-based methods can reduce the memory cost to $O(ln)$, these methods cannot guarantee the approximation quality in theory [22], and the accuracy of the prediction distance largely depends on the selection of landmarks. Therefore, landmark selection is critical to improve the landmark-based method, but it is also NP-hard [16] to select the best landmarks.

The embedding method [13,23–25] is another representative approximate shortest path distance method. In the data preprocessing stage, this method learns the vector embedding of each node through embedding technology [26–29] to maintain the shortest path distance; that is, each node is embedded into the k-dimensional mapping space, such as Euclidean space [30] and hyperbolic space [31], to calculate the shortest path distance between nodes. Therefore, each node has a corresponding k-dimensional embedding vector. When querying the distance, the embedding method uses a more effective distance approximation function, such as directly calculating the p-norm between the embedding vectors or training the neural network to predict the distance according to the embedding vectors and the pre-calculated real shortest path distance. It is precisely because of this that the embedding method can make the query speed faster than other approximation methods. In addition, different embedding technologies, embedding dimensions and embedding spaces have a great impact on the accuracy of prediction distance. Therefore, it is also a challenging problem to select the appropriate embedding technology, dimension and space for different graph data.

Inspired by the above approximate shortest path distance methods, such as the landmark-based method and the embedding method, we proposed a new approximate shortest path distance model based on neural networks. The model integrates the landmark-based method, the embedding method and a neural network model, which greatly reduces the time cost by training.

The remainder of the paper is structured as follows: The preliminary knowledge and related work are reviewed in Section 2. Section 3 introduces our model ndist2vec in detail. We arrange the experiment in Section 4, and we describe the experimental dataset, evaluation index, experimental parameters and experimental results. Finally, conclusions are drawn in Section 5.

2. Preliminary Knowledge and Related Work

2.1. Preliminary Knowledge

Let $G = (V, E, W)$ be an undirected road network graph with $n = |V|$ nodes and $m = |E|$ edges. For each node (road intersection), $v_i \in V$ has a pair of geocoordinates, and the edge (roads) $e_{ij} \in E$ connects nodes v_i and v_j, indicating that they are adjacent and have the weight $e_{ij.w} \in W$, which represents the distance across the edge, i.e., the distance between the two road intersections. Figure 1 shows an example, which contains 5 nodes and 6 edges; v_1 and v_2 are two nodes of the graph, and the presence of edge e_{12} shows that they are adjacent and have the weight $e_{12.w}$, indicating that the distance between them is 2.

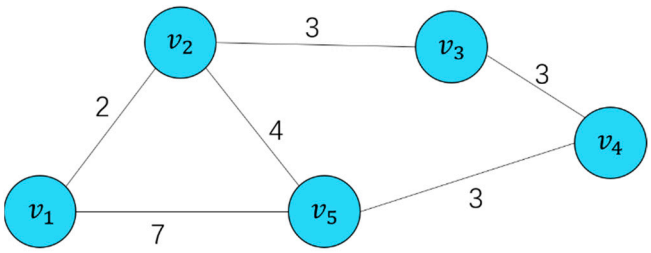


Figure 1. An example of undirected road network graph.

Given node v_i and node v_j, there exists at least one path p_{ij}; this is connected by a series of adjacent nodes, which can make v_i reach v_j. The path length $|p_{ij}|$ is the sum of the weights between these series of nodes. For example, there are three paths from node v_1 to node v_4, which are $v_1 - v_2 - v_3 - v_4$, $v_1 - v_2 - v_5 - v_4$ and $v_1 - v_5 - v_4$, and the corresponding path lengths are 8, 9 and 10, respectively. We specify that the shortest path between node v_i and node v_j is p^*_{ij} and that $d_{ij} = |p^*_{ij}|$ denotes the real shortest path length, so for node v_1 and node v_4, the shortest path length is 8.

2.2. Related Work

Researchers at the University of Passau proposed a new method [13] for approximating the shortest path distance between two nodes in a social graph based on a landmark approach, and they used simple neural networks with node2vec [27] or Poincare [28] embeddings and obtained better results than Orion and Rigel on a social graph dataset. For convenience, we name this method node2vec-Sg. In detail, in the first step, they utilized node embedding technology to learn the vector embedding $\phi(v) \in R^d$ of each node $v \in V$ in the graph. In the second step, they extracted training sample pairs in the entire graph G. They randomly selected $l (l \ll n)$ nodes from V as landmarks, and they applied the breadth-first search (BFS) algorithm to calculate the true distance d_{uv} from each landmark node u to the remaining nodes v so as to obtain $l(n - l)$ sample pairs similar to $((u, v), d_{uv})$. In order to approximately calculate the distance between two nodes, u and v, they combined their embeddings, $\phi(u)$ and $\phi(v)$, through some binary operation $\langle , \cdot \rangle$ (the operations included subtraction, averaging, multiplication or concatenating between vectors) so as to form a training sample pair, such as $(\langle \phi(u), \phi(v) \rangle, d_{uv})$. Finally, the feedforward neural network composed of a single hidden layer was trained through these training samples, and the neural network output the actual value prediction of the shortest path distance $\hat{d}_{uv}$. The specific process is shown in Figure 2.

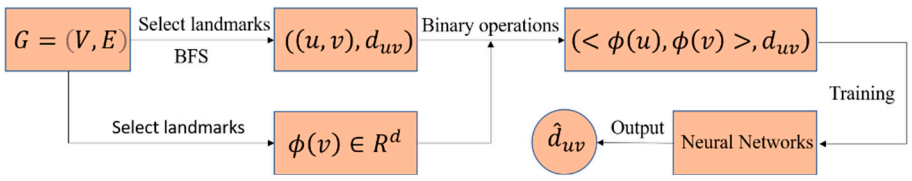


Figure 2. Base architecture of node2vec-Sg.

In addition, since the neural network performs a regression task, the Mean Square Error (MSE) is used as the loss function, and the Stochastic Gradient Descent (SGD) [32] is used as the optimizer. In all their experiments, they confirmed that the node2vec embedding gives better results than the Poincare embedding, adding that the node2vec embedding is not able to learn the structural features of distant nodes, so it is not suitable for graphical structures with many distant nodes, such as road networks.

Researchers have proposed a learning-based model called vdist2vec [25]. The model can effectively and accurately predict the shortest path distance between two nodes in a road network, and the distance prediction time and the storage space of the model are $O(k)$ and $O(nk)$, respectively, in which k is the dimension of node embedding. As shown in Figure 3, in the $2|V|$-dimensional one-hot layer, vdist2vec takes the n-dimensional one-hot vectors $\mathbf{h_i}$ and $\mathbf{h_j}$ of nodes v_i and v_j as inputs, and the next layer is an embedding layer composed of k nodes. In this layer, the embedding of each node is learned to generate the weight matrix $\mathbf{V}$. Through the formula $\mathbf{v_i} = \mathbf{h_i}\mathbf{V}$, the embedding of each node can be obtained; in this case, $\mathbf{v_i}$ and $\mathbf{v_j}$ are obtained, and then $\mathbf{v_i}$. and $\mathbf{v_j}$ are connected as input to train a multilayer perceptron (MLP) in order to predict the shortest path distance for a given embedding of two nodes. Moreover, researchers have proposed improved models vdist2vec-L and vdist2vec-S of the base model vdist2vec, where vdist2vec-L uses Huber loss as the loss function and is able to reduce more errors than the base model vdist2vec, which uses the Mean Square Error (MSE). The model vdist2vec-S is driven by ensemble learning. Four independent MLPs are replaced with the last hidden layer of vdist2vec to focus on the distances in different ranges, and their outputs are added to obtain the final distance prediction.

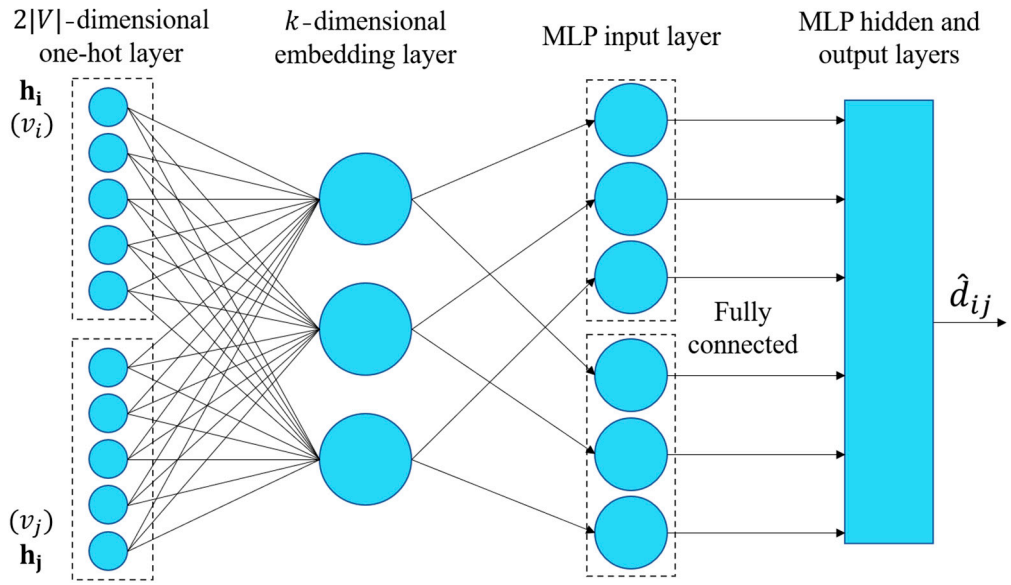


Figure 3. Base architecture of vdist2vec.

The above models achieve fast distance prediction without increasing the spatial cost. The advantages of the three models have been confirmed in experiments on several different real road networks, and vdist2vec-S is the best one of the three models. However, in order to better learn the node embeddings and to obtain a higher prediction accuracy, the models use all $n(n-1)$ node pairs as training samples to train the neural network, thus significantly increasing the training time. Inspired by the above-mentioned use of embedding methods and landmark-based methods in the approximate shortest path distance problem, we propose a new approximate shortest path distance prediction model, ndist2vec. The goal is to maintain a relatively high prediction accuracy and fast query time while greatly reducing the training time. In the next section, we elaborate on the details of ndist2vec.

3. Ndist2vec

In this section, we proposed the ndist2vec model. As shown in Figure 4, we used the method of randomly selecting landmarks to divide the set of nodes V into a set of landmark nodes V^L and a set of remaining nodes V^R. Then, our ndist2vec model selects two nodes, $v_i^L \in V^L, i = 1, \ldots, l$ and $v_j^R \in V^R, j = 1, \ldots, n-l$, and obtains their corresponding embedding vectors, $\mathbf{v_i^L}$ and $\mathbf{v_j^R}$, respectively, by using the vector embedding matrix $\mathbf{H} \in R^{n*k}$ (each node has a corresponding embedding vector), and it finally connects the vectors $\mathbf{v_i^L}$ and $\mathbf{v_j^R}$ as input to train a multilayer neural network to output the distance $\hat{d}_{ij}$ between the two given nodes, v_i^L and v_j^R.

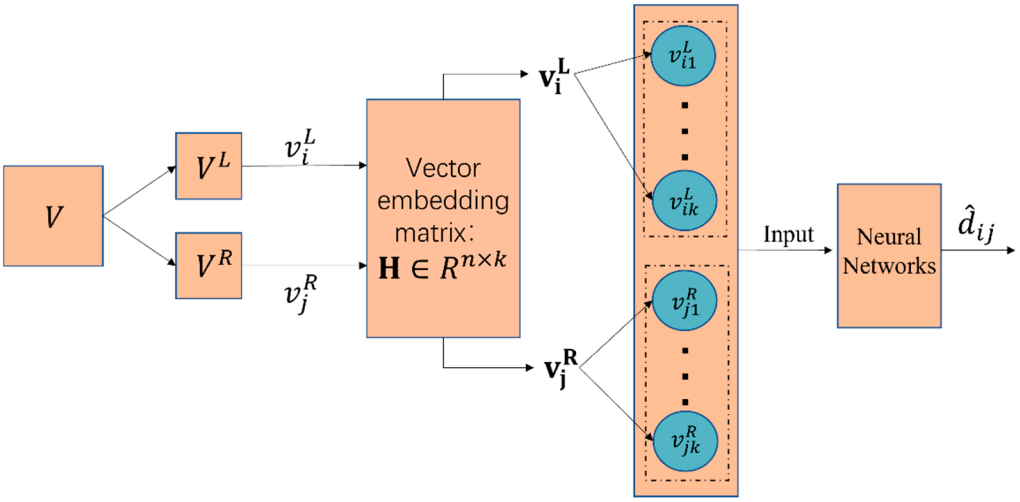


Figure 4. Base architecture of ndist2vec.

Specifically, our goal was to extract training pairs from the entire graph G to train a multilayer neural network and then to predict the shortest path distance $\hat{d}_{ij}$ between any two nodes v_i and v_j in a graph G. In the first stage, we selected l (where $l < n$) nodes in the node set V as landmarks and generated the set of landmark nodes V^L, and the remaining nodes in V generated the set of remaining nodes V^R; i.e., the set V is divided into sets V^L and V^R (where $|V^L| = l$ and $|V^R| = n - l$). In the second stage, we randomly initialized a vector embedding matrix $\mathbf{H} \in R^{n \times k}$ so that we could obtain an embedding vector $\mathbf{v_i} \in R^k$ for each node $v_i \in V$. Since our embeddings can be guided directly by distance prediction, we can update $\mathbf{H}$ based on the back propagation of the training predictions for each epoch, which also means updating each embedding vector $\mathbf{v_i}$. In the third stage, using the vector embedding matrix $\mathbf{H}$, we could obtain the embedding vector $\mathbf{v_i^L}$ corresponding to node v_i^L of the landmark node set V^L and the embedding vector $\mathbf{v_j^R}$ corresponding to node v_j^R of the remaining node set V^R, and we connected them together as training samples while calculating the actual shortest path distance d_{ij} of these two nodes as the supervision information (label) of this sample. Then, we utilized the above method to traverse each landmark node and the remaining nodes to obtain $l(n-l)$ training samples, along with their corresponding supervision information. Finally, the training samples were used as input to a multilayer neural network. The neural network maps the input training samples to real-valued distances.

As shown in Figure 5, we designed a four-layer neural network consisting of an input layer, two hidden layers and an output layer. The size of the input layer depends on the dimension k of the vector embedding, and since two vectors are connected in series, $2k$ neurons are required. Since the ReLU [33] function is effective in training neural networks, we set the activation function for the first three layers to the ReLU function. In the output layer, we learned the distances in different ranges in the form of ensemble learning, and we added their outputs to obtain the final distance prediction.

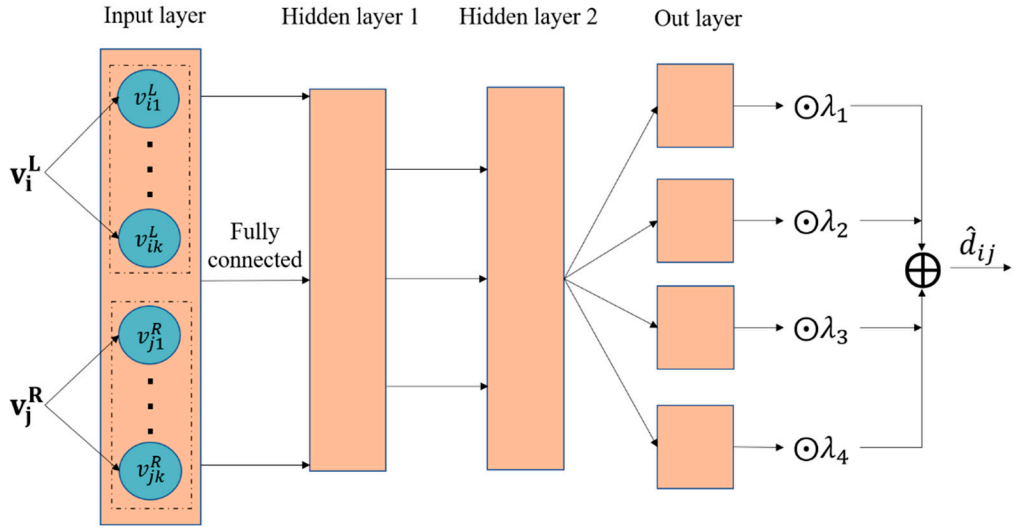


Figure 5. Base architecture of neural networks.

We used the Mean Square Error (MSE) to measure the quality of the predictor because the network performs a regression task, so the Mean Square Error of the actual node distance d_{ij} and the predicted distance $\hat{d}_{ij}$ was taken as our training loss function L_d:

$$L_d = \frac{1}{N}\sum\left(d_{ij} - \hat{d}_{ij}\right)^2 \quad (1)$$

Finally, we used adaptive moment estimation (ADAM) [34] as an optimizer, which controls the learning rate after bias correction with a defined range of learning rates per iteration. We made the parameters relatively smooth, and their effectiveness has been verified in a large number of deep neural network experiments. During training, all parameters of the neural network are randomly initialized. The training samples are fed into the network in batches for training. The training loss L_d is passed back to optimize all parameters in the network.

4. Experiment

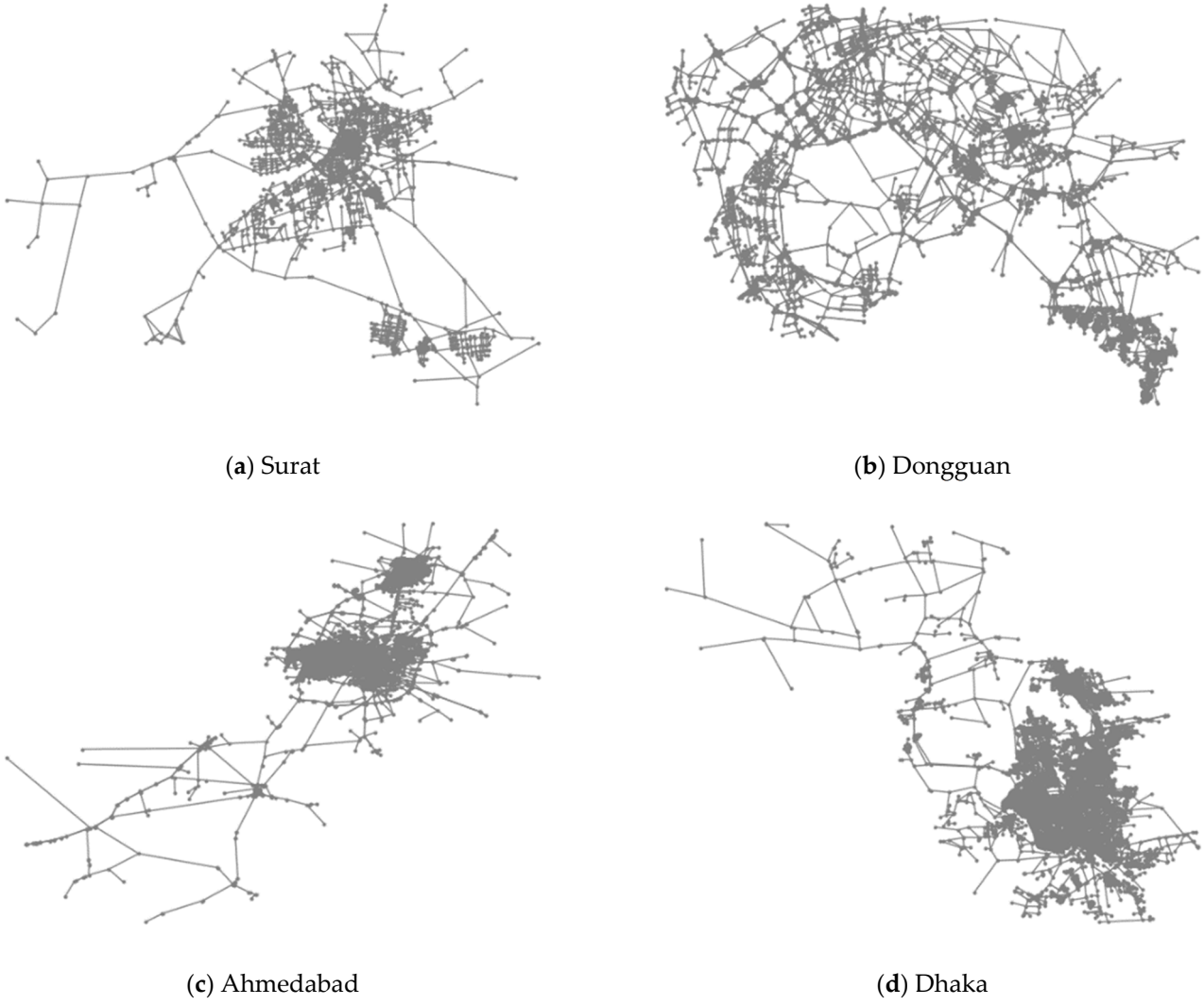
In this section, we tested our proposed ndist2vec on four road network datasets and compared it with node2vec-sg and vdis2vec. All these methods were implemented by Python 3.7 on a PC with an Intel Core Duo Processor (double 4.2 GHz) with 16 GB RAM. Next, we first described the datasets, some parameter settings and evaluation metrics; then, we described the experiments that we conducted and presented the results; and, finally, we presented the conclusions drawn from the experimental results and the reasons why they turned out the way they did. The source code of the program and the experimental data were archived on figshare [35].

4.1. Datasets and Hyperparameters

We used four different road network graphs [2] for the experiments. We extracted the maximum connected component from these road graphs, renamed the node name and specified the coordinates. Therefore, all datasets contain weighted edges and unique map coordinates for each node. In addition, they are all undirected, and the number of nodes $n = |V|$, number of edges $m = |E|$, the maximum distance d_{max}, the minimum distance d_{min} and the average distance d_{mean} between the nodes are summarized in Table 1, and Figure 6 shows the original road network.

Table 1. Statistics of road network datasets.

Datasets	n	m	d_{max}	d_{min}	d_{mean}
Surat, India (SU)	2508	3591	50,954.24 m	1.24 m	8866.22 m
Dongguan, China (DG)	7658	10,542	96,547.07 m	1.43 m	35,096.00 m
Ahmedabad, India (AH)	12,747	18,117	130,910.82 m	1.05 m	17,634.84 m
Dhaka, Bangladesh (DH)	14,689	19,457	74,156.86 m	1.02 m	10,468.47 m



(**a**) Surat (**b**) Dongguan

(**c**) Ahmedabad (**d**) Dhaka

Figure 6. Road networks.

For all comparison methods, we used their recommended hyperparameter settings. In our ndist2vec model, the neural network consists of four layers: an input layer containing $2k$ neurons; an output layer containing 1 neuron; and two hidden layers containing 100 and 20 neurons, respectively, where k is the embedding dimension, and we set $k = 50$. The four parameters of ensemble learning, $d_{max} > \lambda_4 > \lambda_3 > \lambda_2 > \lambda_1 > 0$, were randomly initialized and updated with each epoch of training. The model was trained for a total of 30 epochs. All $n(n-1)$ sample pairs were trained in the first epoch, and in the remaining 29 epochs, $l = 10\%n - 15\%n$ landmarks were randomly re-generated for each epoch; i.e., $l(n-l)$ sample pairs were trained for each epoch, and all epochs were trained at a learning rate of 0.01.

We used two types of metrics to measure the accuracy and speed of our method. Firstly, we utilized the Mean Absolute Error (MAE) and the Mean Relative Error (MRE)

to measure the difference between our predicted value $\hat{d}_{ij}$ and the real value d_{ij}. Their definitions are

$$\mathrm{MAE} = \frac{1}{N}\sum |d_{ij} - \hat{d}_{ij}| \tag{2}$$

and

$$\mathrm{MRE} = \frac{1}{N}\sum \frac{|d_{ij} - \hat{d}_{ij}|}{d_{ij}} \tag{3}$$

respectively, and the smaller the value, the higher the accuracy. Secondly, we used the training time (PT) and the average distance prediction time (AT) to measure the speed of the model. In our experiment, our prediction set consists of $n(n-1)$ pairs of all nodes.

4.2. Overall Results

Table 2 shows the experimental results of the prediction error and time cost of ndist2vec, vdist2vec-S and node2vec-Sg on the four road network datasets. In terms of the prediction error, it can be seen that vdist2vec-S has the smallest MAE and MRE for each dataset because it learns all node pair information and retains the distance information of the node pairs through node embedding (Bolded numbers indicate best performance). Our method ndist2vec has a larger MAE and MRE than those of vdist2vec-S, but the size is limited. We used the landmark-based method for learning, and we did not learn all the node pair information but retained the node embedding information in the vector embedding matrix. Node2vec-Sg is an approximate method for predicting the shortest path distance of social networks. We changed its weight to the real distance and carried out experiments on road networks, but we did not obtain good results. The reason is that node2vec-Sg sometimes oversamples and undersamples in the process of generating training samples; that is, the samples are not evenly divided, and not all node pair information is learned. At the same time, the embedding technology node2vec is not suitable for road network nodes; this is explained in our subsequent experiments.

Table 2. Experimental results of three models.

Datasets	Models	MAE	MRE	PT (h)	AT (μs)
SU	ndist2vec	99.74	0.034	**0.10**	**7.79**
	vdist2vec-S	**87.22**	**0.028**	0.41	8.11
	node2vec-Sg	642.09	0.171	0.35	8.02
DG	ndist2vec	278.41	0.046	**0.48**	**16.95**
	vdist2vec-S	**144.32**	**0.030**	2.40	17.12
	node2vec-Sg	2412.34	0.203	1.37	18.06
AH	ndist2vec	146.32	0.033	**2.50**	**25.56**
	vdist2vec-S	**100.98**	**0.021**	12.00	40.37
	node2vec-Sg	3711.44	0.258	9.74	35.69
DH	ndist2vec	109.73	0.029	**3.10**	**30.20**
	vdist2vec-S	**90.77**	**0.020**	18.50	45.40
	node2vec-Sg	4103.29	0.273	15.63	39.69

In terms of the training time cost PT, our method ndist2vec has the best performance. The advantage of our model is that it sacrifices some accuracy to greatly reduce the training time, and this accuracy sacrifice is worthwhile. Specifically, for the datasets SU, AH and DH, the MAEs of ndist2vec are 1.14, 1.44 and 1.20 times those of vdist2vec-s, respectively, but the training time PT is at least one-quarter of that of vdis2vec-S. Moreover, with an increase in the number of nodes (n), the time gap will become increasingly larger. Although node2vec-Sg is also a landmark-based method, it is very time consuming because of the different ways of selecting training samples.

The average prediction time AT is calculated by dividing the total prediction time S of all node pairs of samples by the number of sample pairs $n(n-1)$ of all the nodes, and

the unit is microseconds (μs). The three models need to link the embedded vectors of two nodes in the prediction distance and input them into the corresponding neural network (the neural network structure of the three methods is similar) for forward propagation to obtain the prediction results. It can be seen that the AT of ndist2vec for the four datasets is the smallest. This is because the vector embedding dimension k of ndist2vec is always 50, while the embedding dimension k of vdist2vec-S is $0.02n$, and the dimension of node2vec is $k = 128$; more dimensions will increase the training time PT and the average prediction time AT.

Ndist2vec performs the worst in terms of prediction error in the Dongguan dataset. This is because although the Dongguan dataset only has 7658 nodes, a total of about 59 million sample pairs, its average distance between nodes is the highest, $d_{mean} = 35,096$ m. It is conceivable that the node distribution in the Dongguan dataset is dominated by a large distance. In addition, our method is based on landmarks. We did not learn all the sample pairs during training, so we lack the learning of large-distance sample pairs. Therefore, our method may not be suitable for datasets with large distances between nodes, and this will be our next breakthrough direction.

Our training strategy is to train 30 epochs. The first epoch trains all $n(n-1)$ sample pairs, the remaining 29 epochs train $l(n-l)$ landmark sample pairs, and landmarks are randomly selected again in each epoch. In addition, our vector embedding matrix **H** is updated according to the prediction results. Using the control variable method, we summarized the four models in Table 3 and compared them in Table 4 to verify the effectiveness of our model training strategy.

Table 3. Model settings.

Models	*Embedding*	*Epoch*	*Landmark*
ndist2vec	*L*	*Epoch1*	*S*
ndist2vec-1	*N*	*Epoch1*	*S*
ndist2vec-2	*L*	*Epoch2*	*S*
ndist2vec-3	*L*	*Epoch1*	*F*

Table 4. Comparison results of ndist2vec and three variant models.

Datasets	Models	MAE	MRE	PT (h)	AT (μs)
SU	ndist2vec	**99.74**	**0.034**	0.10	7.79
	ndist2vec-1	632.11	0.166	0.19	13.56
	ndist2vec-2	805.76	0.287	0.07	8.58
	ndist2vec-3	601.29	0.161	0.10	7.81
DG	ndist2vec	**278.41**	**0.046**	0.48	16.95
	ndist2vec-1	2212.16	0.199	1.37	20.60
	ndist2vec-2	1663.10	0.140	0.36	15.07
	ndist2vec-3	1118.09	0.084	0.61	16.76
AH	ndist2vec	**146.32**	**0.033**	2.50	25.56
	ndist2vec-1	3550.57	0.245	3.08	28.79
	ndist2vec-2	962.04	0.182	2.08	25.88
	ndist2vec-3	902.58	0.111	2.49	25.72
DH	ndist2vec	**109.73**	**0.029**	3.10	30.20
	ndist2vec-1	3956.17	0.251	4.09	34.94
	ndist2vec-2	448.66	0.088	2.33	29.31
	ndist2vec-3	331.61	0.045	3.10	29.28

Table 3 shows the training settings of the different models. *Embedding* represents the form of the vector embedding matrix, *L* indicates that the vector embedding matrix **H** can be updated through the prediction results (that is, our method), and *N* indicates that the vector embedding matrix **H** is learned in advance according to node2vec embedding

technology and will not change with the prediction results. *Epoch* has two choices. *Epoch1* indicates that the first epoch trains all sample pairs, and the remaining 29 epochs train landmark sample pairs; *Epoch2* indicates that 30 epochs train landmark sample pairs. The *S* in *Landmark* indicates that each epoch generates new landmark sample pairs to participate in training, and *F* indicates that the landmark is generated once and fixed to participate in all epochs training; that is, each epoch is trained with fixed landmark sample pairs.

Table 4 shows the experimental results of ndist2vec and the other variant models. Specifically, by comparing ndist2vec and ndist2vec-1, we can see that for the four datasets, the result of ndist2vec is better than that of ndist2vec-1, which shows that the method of updating the vector embedding matrix according to the prediction results is more suitable for the prediction of the shortest path distance of a road rather than directly using node2vec embedding technology. Node2vec embedding technology pays more attention to capturing the similarity between nodes, but the shortest path prediction of the road network pays more attention to the distance relationship between nodes. Therefore, the method of updating the vector embedding matrix according to the prediction results is more appropriate to capture the characteristics of the road network. It measures the distance between the nodes rather than the similarity.

Ndist2vec trained all $n(n-1)$ sample pairs in the first epoch, while ndist2vec-2 used the landmark-based method in the first epoch and only trained $l(n-l)$ sample pairs. As a result, the training time PT of ndist2vec was higher than that of ndist2vec-2 (only higher in the first epoch time of training), but the MAE and MRE were reduced. The reason for this is that, although the landmark-based method can reduce the training time, we used the random landmark selection method, which may not generate better sample pairs for training in the first epoch, and then we updated the vector embedding matrix **H**. However, training all sample pairs in the first epoch can better teach and update the vector embedding matrix **H** and provide a good foundation for the next 29 epochs of training.

Comparing the ndist2vec model and the ndist2vec-3 model, in the landmark-based training epoch (the remaining 29 epochs), BB repeats learning 29 times for $l(n-l)$ sample pairs, so it can only learn the information of $l(n-l)$ sample pairs. When the random landmark selection is not good, the performance of ndis2vec-3 deteriorates. However, in the epoch landmark-based training of the ndist2vec model, each epoch randomly selects new landmarks and generates new sample pairs for training; that is, the ndist2vec model can learn more information of $|\cup_{i=1}^{29} T_i| - |\cap_{i=1}^{29} T_i|$ sample pairs (where T_i is the set of sample pairs generated by the combination of V^{L_i} and V^{R_i}). For regression tasks, the more information used for learning means better fitting. Therefore, it can be seen in Table 4 that ndist2vec performs better than ndis2vec-3.

4.3. Discussion

In this paper, we studied undirected road networks. In fact, in a directed road network, whether all nodes are bidirectionally connected determines whether our model is feasible. When all nodes are bidirectionally connected, a feasible solution for our model in directed road networks is to change the connection order of the node embedding vectors and to train two prediction models in order to predict the bidirectional node distances separately. Currently, we cannot come up with a solution to apply this model to a case where only some nodes are bidirectionally connected. In addition, our model uses a randomly selected landmark method; i.e., l landmark nodes are randomly selected from the node set.

The method of randomly selecting landmarks does not seem to be the best choice, and a better landmark selection method may cause a large improvement in the results. We also tried some other methods of selecting landmarks, such as using the k-media algorithm to select l median nodes and using the concave hull algorithm to select all edge nodes, but the effect was not as good as directly selecting l nodes at random. We were not able to determine the specific reasons for this.

5. Conclusions and Future Work

This paper presents a model, nidst2vec, based on embedding and landmark technology, which uses multi-layer neural networks to obtain an approximate solution to the shortest path distance problem. Ndist2vec learns the distance information between nodes through embedding technology; i.e., it learns the updated vector embedding matrix **H** to maintain the accuracy of prediction, and only $O(50n)$ space is required to store the vector embedding matrix **H**. The landmark method is added to ndist2vec, which greatly reduces the training time. In particular, in each training round, the model selects new landmarks to learn more information about the node pairs without increasing the training time, which facilitates the updating of node embeddings. The experimental results show that, while the prediction error is elevated (by up to 20%), the training time is significantly reduced (by at least 75%) compared to that of the benchmark method.

In future work, we plan to adapt our method to a road network graph with a large distance between nodes and to extend it to other types of graph data. In addition, combining our method, studying more reasonable methods of landmark selection, and exploring the impact of different embedding techniques and embedding dimensions are all worthwhile research directions. We will use a geospatial big data computing framework [36,37] to improve the performance of the deep learning model considering large datasets in future work.

Author Contributions: Conceptualization, Shaohua Wang and Xu Chen; methodology, Shaohua Wang and Xu Chen; software, Xu Chen; validation, Xu Chen and Haojian Liang; formal analysis, Xu Chen and Huilai Li; investigation, Huilai Li and Fangzheng Lyu; resources, Shaohua Wang; data curation, Xu Chen; writing—original draft preparation, Xu Chen and Shaohua Wang; writing—review and editing, Xu Chen, Shaohua Wang, Huilai Li, Fangzheng Lyu, Haojian Liang, Xueyan Zhang and Yang Zhong; visualization, Xu Chen; supervision, Shaohua Wang; project administration, Shaohua Wang. All authors have read and agreed to the published version of the manuscript.

Funding: This research was financially supported by the Strategic Priority Research Program of the Chinese Academy of Sciences (Grant No. XDA28100500) and the Hundred Talents Program Youth Project (Category B) of the Chinese Academy of Sciences (E2Z10501).

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Demetrescu, C.; Goldberg, A.; Johnson, D. *9th DIMACS Implementation Challenge-Shortest Paths*; American Mathematical Society: Providence, RI, USA, 2006.
2. Karduni, A.; Kermanshah, A.; Derrible, S. A protocol to convert spatial polyline data to network formats and applications to world urban road networks. *Sci. Data* **2016**, *3*, 160046. [CrossRef] [PubMed]
3. Abraham, I.; Delling, D.; Goldberg, A.V.; Werneck, R.F. A hub-based labeling algorithm for shortest paths in road networks. In *International Symposium on Experimental Algorithms*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 230–241.
4. Bast, H.; Funke, S.; Sanders, P.; Schultes, D. Fast routing in road networks with transit nodes. *Science* **2007**, *316*, 566. [CrossRef] [PubMed]
5. Dijkstra, E.W. A note on two problems in connexion with graphs. *Numer. Math.* **1959**, *1*, 269–271. [CrossRef]
6. Floyd, R.W. Algorithm 97: Shortest path. *Commun. ACM* **1962**, *5*, 345. [CrossRef]
7. Chang, L.; Yu, J.X.; Qin, L.; Cheng, H.; Qiao, M. The exact distance to destination in undirected world. *VLDB J.* **2012**, *21*, 869–888. [CrossRef]
8. Akiba, T.; Iwata, Y.; Yoshida, Y. Fast exact shortest-path distance queries on large networks by pruned landmark labeling. In Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, New York, NY, USA, 22–27 June 2013; pp. 349–360.
9. Cohen, E.; Halperin, E.; Kaplan, H.; Zwick, U. Reachability and distance queries via 2-hop labels. *SIAM J. Comput.* **2003**, *32*, 1338–1355. [CrossRef]
10. Thorup, M.; Zwick, U. Approximate distance oracles. *JACM* **2005**, *52*, 1–24. [CrossRef]

11. Sankaranarayanan, J.; Samet, H. Distance oracles for spatial networks. In Proceedings of the 2009 IEEE 25th International Conference on Data Engineering, Shanghai, China, 29 March–2 April 2009; pp. 652–663.
12. Chechik, S. Approximate distance oracles with improved bounds. In Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing, Portland, OR, USA, 14 June 2015; pp. 1–10.
13. Rizi, F.S.; Schloetterer, J.; Granitzer, M. Shortest path distance approximation using deep learning techniques. In Proceedings of the 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), Barcelona, Spain, 28–31 August 2018; pp. 1007–1014.
14. Huang, S.; Wang, Y.; Zhao, T.; Li, G. A Learning-based Method for Computing Shortest Path Distances on Road Networks. In Proceedings of the 2021 IEEE 37th International Conference on Data Engineering (ICDE), Chania, Greece, 19–22 April 2021; pp. 360–371.
15. Chivers, I.; Sleightholme, J. An introduction to Algorithms and the Big O Notation. In *Introduction to Programming with Fortran*; Springer: Cham, Switzerland, 2015; pp. 359–364.
16. Potamias, M.; Bonchi, F.; Castillo, C.; Gionis, A. Fast shortest path distance estimation in large networks. In Proceedings of the 18th ACM Conference on Information and Knowledge Management, Turin, Italy, 22–26 October 2009; pp. 867–876.
17. Jin, R.; Ruan, N.; Xiang, Y.; Lee, V. A highway-centric labeling approach for answering distance queries on large sparse graphs. In Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, Scottsdale, AZ, USA, 20–24 May 2012; pp. 445–456.
18. Jiang, M.; Fu, A.W.C.; Wong, R.C.W.; Xu, Y. Hop doubling label indexing for point-to-point distance querying on scale-free networks. *arXiv* **2014**, arXiv:1403.0779. [CrossRef]
19. Tang, L.; Crovella, M. Virtual landmarks for the internet. In Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement, Miami Beach, FL, USA, 27–29 October 2003; pp. 143–152.
20. Zhao, X.; Zheng, H. Orion: Shortest path estimation for large social graphs. In Proceedings of the 3rd Workshop on Online Social Networks (WOSN 2010), Boston, MA, USA, 22 June 2010.
21. Gubichev, A.; Bedathur, S.; Seufert, S.; Weikum, G. Fast and accurate estimation of shortest paths in large graphs. In Proceedings of the 19th ACM International Conference on Information and Knowledge Management, Atlanta, GA, USA, 17–22 October 2010; pp. 499–508.
22. Kleinberg, J.; Slivkins, A.; Wexler, T. Triangulation and embedding using small sets of beacons. In Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science, Rome, Italy, 17–19 October 2004; pp. 444–453.
23. Zhao, X.; Sala, A.; Zheng, H.; Zhao, B.Y. Efficient shortest paths on massive social graphs. In Proceedings of the 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), Orlando, FL, USA, 15–18 October 2011; pp. 77–86.
24. Cao, S.; Lu, W.; Xu, Q. Deep neural networks for learning graph representations. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; Volume 30.
25. Qi, J.; Wang, W.; Zhang, R.; Zhao, Z. A learning based approach to predict shortest-path distances. In Proceedings of the 23rd International Conference on Extending Database Technology (EDBT), Copenhagen, Denmark, 30 March–2 April 2020.
26. Cao, S.; Lu, W.; Xu, Q. Grarep: Learning graph representations with global structural information. In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, Melbourne, Australia, 19–23 October 2015; pp. 891–900.
27. Grover, A.; Leskovec, J. node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 855–864.
28. Nickel, M.; Kiela, D. Poincaré embeddings for learning hierarchical representations. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 6341–6350.
29. Zhang, J.; Dong, Y.; Wang, Y.; Tang, J.; Ding, M. ProNE: Fast and Scalable Network Representation Learning. *IJCAI* **2019**, *19*, 4278–4284.
30. Darmochwał, A. The Euclidean space. *Formaliz. Math.* **1991**, *2*, 599–603.
31. Kleinberg, R. Geographic routing using hyperbolic space. In Proceedings of the IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications, Anchorage, AK, USA, 6–12 May 2007; pp. 1902–1909.
32. Bottou, L. Large-scale machine learning with stochastic gradient descent. In Proceedings of the COMPSTAT'2010, Paris, France, 22–27 August 2010; pp. 177–186.
33. Nair, V.; Hinton, G.E. Rectified linear units improve restricted boltzmann machines. In Proceedings of the ICML, Haifa, Israel, 21–24 June 2010.
34. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
35. Ndist2vec Project. Available online: https://doi.org/10.6084/m9.figshare.20238813.v1 (accessed on 10 July 2022).
36. Wang, S.; Zhong, Y.; Wang, E. An integrated GIS platform architecture for spatiotemporal big data. *Future Gener. Comput. Syst.* **2019**, *94*, 160–172. [CrossRef]
37. Heitzler, M.; Lam, J.C.; Hackl, J.; Adey, B.T.; Hurni, L. GPU-accelerated rendering methods to visually analyze large-scale disaster simulation data. *J. Geovis. Spat. Anal.* **2017**, *1*, 1–18. [CrossRef]