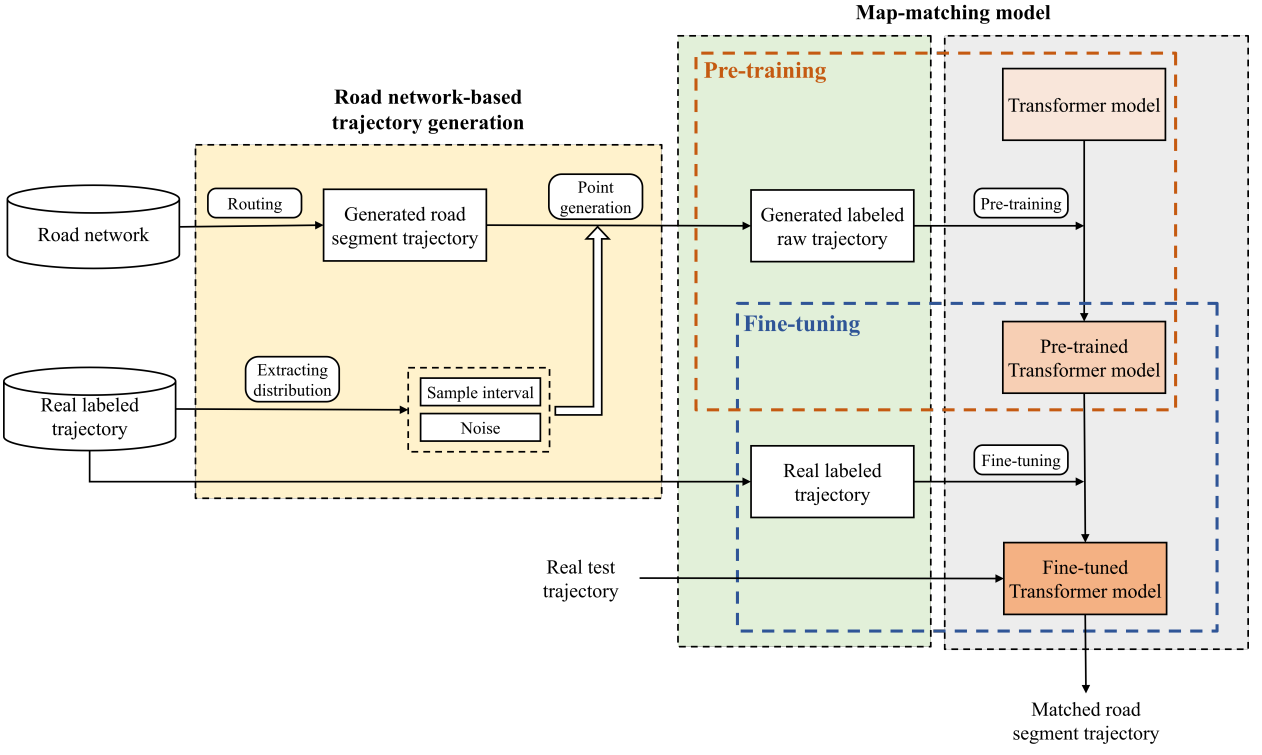
Graphical Abstract

Transformer-based Map Matching Model with Limited Labeled Data using Transfer-Learning Approach

Zhixiong Jin,Jiwon Kim,Hwasoo Yeo,Seongjin Choi



Highlights

Transformer-based Map Matching Model with Limited Labeled Data using Transfer-Learning Approach

Zhixiong Jin,Jiwon Kim,Hwasoo Yeo,Seongjin Choi

- We design a transfer learning approach to solve labeled data sparsity problem for deep learning-based map-matching model development.
- We develop a Transformer-based map-matching model that improves model performance in terms of model accuracy, computation time, design flexibility, and explainability.
- We evaluate the performance of map-matching models using three different metrics (Average Hamming Distance, F-score, and BLEU) at the point and segment level.
- We analyze the map-matching results at trajectory-level and attention mechanism-level to improve the explainability and interpretability of model performance.

Transformer-based Map Matching Model with Limited Labeled Data using Transfer-Learning Approach

Zhixiong Jin[a], Jiwon Kim[b], Hwasoo Yeo[a] and Seongjin Choi[c,*]

[a]Department of Civil and Environmental Engineering, Korea Advanced Institute of Science and Technology, 291 Daehak-ro, Yuseong-gu, Daejeon, Republic of Korea

[b]School of Civil Engineering, The University of Queensland, Brisbane St Lucia, Queensland, Australia

[c]Department of Civil Engineering, McGill University, 817 Sherbrooke Street West, Montreal, Quebec H3A 0C3, Canada

ARTICLE INFO

Keywords:
Map matching
Transformer
Transfer learning
Trajectory data
Limited labeled data

ABSTRACT

In many spatial trajectory-based applications, it is necessary to map raw trajectory data points onto road networks in digital maps, which is commonly referred to as a map-matching process. While most previous map-matching methods have focused on using rule-based algorithms to deal with the map-matching problems, in this paper, we consider the map-matching task from the data-driven perspective, proposing a deep learning-based map-matching model. We build a Transformer-based map-matching model with a transfer learning approach. We generate trajectory data to pre-train the Transformer model and then fine-tune the model with a limited number of labeled data to minimize the model development cost and reduce the real-to-virtual gaps. Three metrics (Average Hamming Distance, F-score, and BLEU) at two levels (point and segment level) are used to evaluate the model performance. The model is tested with real-world datasets, and the results show that the proposed map-matching model outperforms other existing map-matching models. We also analyze the matching mechanisms of the Transformer in the map-matching process, which helps to interpret the input data internal correlation and external relation between input data and matching results. In addition, the proposed model shows the possibility of using generated trajectories to solve the map-matching problems in the limited labeled data environment.

1. Introduction

The proliferation of mobile devices equipped with Global Positioning System (GPS) has promoted the generation of massive amounts of GPS trajectory data, which capture user-specific mobility characteristics and system-wide spatio-temporal traffic patterns (Kim and Mahmassani, 2015; Gong et al., 2017). The big trajectory data facilitate the emergence of many trajectory-based applications such as path discovery (Chen et al., 2011), location/destination prediction (Choi et al., 2018), movement pattern analysis (Renso et al., 2013; Choi et al., 2021), and urban planning (Huang et al., 2015). However, spatial discrepancies between recorded GPS locations and real object positions are prevalent, which raises the challenges of using GPS trajectory data in trajectory-based applications. For instance, different types of failures, such as limited satellite visibility, reflected satellite signals, and GPS receiver malfunctions, can add errors in position coordinates in the range of $5-20m$ (Sharath et al., 2019). Deficiencies in current commercial digital maps can add further matching errors ranging $5-20m$ (Toledo-Moreo et al., 2009). As a result, a preprocessing procedure known as *Map Matching* is necessary to correctly identify the true road segments that the moving object of a given raw GPS trajectory traveled on. (Quddus et al., 2007).

Map-matching algorithms have been studied for more than two decades to support various trajectory-based applications (Kubicka et al., 2018; Chao et al., 2020; Hashemi and Karimi, 2014). The map-matching methods can be divided into *online matching* that deals with streaming GPS data in real-time and *offline matching* that processes historical trajectory data in off-line settings (Gong et al., 2017). This paper focuses on offline map-matching, where our

*Corresponding author. Work done while at KAIST

Email addresses: iziz56@kaist.ac.kr (Z. Jin); jiwon.kim@uq.edu.au (J. Kim); hwasoo@kaist.ac.kr (H. Yeo); seongjin.choi@mcgill.ca (S. Choi)

ORCID(s): 0000-0002-1370-781X (Z. Jin); 0000-0001-6380-3001 (J. Kim); 0000-0002-2684-0978 (H. Yeo); 0000-0001-7140-537X (S. Choi)

[1]The authors appreciate Seoul Metropolitan Government and Dr. Min Ju Park for providing the taxi DTG data of Seoul city.

[2]This work was supported by Korea Institute of Police Technology (KIPoT) grant funded by the Korea government (KNPA) (092021C29S01000, Development of Traffic Congestion Management System for Urban Network)

goal is to improve the map-matching accuracy based on historical GPS vehicle trajectory data collected from urban road networks. Typical offline methods include geometric algorithm (Bernstein et al., 1996), weight-based method (Sharath et al., 2019), Kalman Filter (Jo et al., 1996), Hidden Markov Model (HMM) (Newson and Krumm, 2009), and fuzzy control theory (Kim et al., 1998). Most of the existing offline map-matching algorithms take rule-based approaches, where algorithms apply pre-defined rules to process each trajectory separately to find its best matching road sequence. While these methods are fast and intuitive, processing individual trajectories independently often leads to relatively lower map-matching performance as it cannot capture real-life, collective travel patterns embedded in a large amount of trajectory data. The big trajectory data contain valuable information which helps us to understand user mobility patterns and noise characteristics. On the one hand, it is possible to leverage mobility patterns in historical trajectories to improve matching performance. The reason for this is that travel patterns between certain locations are usually highly skewed and similar trajectories can often complement each other to make themselves more complete (Zheng et al., 2012; Lin et al., 2021). On the other hand, the big trajectory data can be used to extract the characteristics of GPS noise in order to further increase map-matching performance. A certain user's accumulated trajectory can disclose the position devices' noise characteristics (Feng et al., 2020; Wang et al., 2011). Also, the aggregated trajectories gathered from various vehicles driving through a similar road network can reveal its noise characteristics induced by dense urban canyons (Mohamed et al., 2016), complicated road geometries (Merry and Bettinger, 2019), and varying weather conditions (Kos et al., 2013).

In recent years, deep learning methods have gained popularity as powerful techniques for extracting information from big data and have achieved great successes in many fields such as natural language processing (Young et al., 2018), computer vision (Voulodimos et al., 2018), and speech recognition (Amodei et al., 2016). In transportation engineering, deep learning methods are widely used in analyzing spatio-temporal characteristics of traffic to support applications such as trajectory prediction (Choi et al., 2018, 2019; Sun and Kim, 2021), traffic flow prediction (Lv et al., 2014; Wu et al., 2018; Wang et al., 2016), and traffic signal control (Genders and Razavi, 2016; Yoon et al., 2020, 2021). In the context of map-matching, several studies have shown the possibility of developing deep learning-based map-matching models that can leverage information in big trajectory data to improve map-matching performance (Feng et al., 2020; Zhao et al., 2019).

However, there are two challenges for developing map-matching models based on deep learning. One challenge is the lack of *labeled data* for training the model. Collecting a great number of labeled data for model training is expensive and laborious (Kortylewski et al., 2018). One of the approaches to solving the problem is using data augmentation methods, which artificially inflate the dataset to generate invariant examples (Taylor and Nitschke, 2018). However, developing the model only with generated data is not sufficient enough due to the existence of real-to-virtual gaps. Therefore, we need an additional approach to complement the usage of the generated data in map-matching model development. Another challenge is related to the limited capability of the sequential learning models that are commonly adopted as map-matching models. Map-matching tasks can be considered solving sequence-to-sequence (seq2seq) problems, where input sequences (raw GPS trajectories) are converted into another domain of output sequences (road segments). The existing studies apply sequential learning structures based on Recurrent Neural Network (RNN) to solve map-matching problems(Liu et al., 2021). However, RNN has a limited capability to fully capture the intercorrelation among data points in input trajectory sequences and this can produce relatively lower map-matching performance (Bahdanau et al., 2014). To overcome the stated limitation, Feng et al. (2020) uses the attentional seq2seq model in the map-matching task. However, large computation time in model training and inflexibility of model structure problems exist due to the Recurrent Neural Networks (Vaswani et al., 2017).

In order to address these two challenges, this study develops a *Transformer*-based map-matching model combined with a training approach based on *transfer-learning*. The training approach based on transfer learning is used to solve data sparsity problems for model training. We first pre-train the deep learning model by using a large number of trajectories generated based on road network information. Then, a limited amount of available labeled data are used to fine-tune the model to reduce the real-to-virtual gaps. This kind of approach has been applied widely in computer vision to construct high-performance deep learning models when limited labeled data are available (Kortylewski et al., 2018; Tremblay et al., 2018; Namozov and Im Cho, 2018). Therefore, in this research, we can be benefited by using the stated approach to tackle the map-matching problem.

To solve the problems of RNN-based sequential learning models, this study uses *Transformer*, a prominent deep learning model that has been successfully applied in seq2seq problems (Vaswani et al., 2017). The Transformer is designed to consider the internal correlation of input data as well as the external relationship between input and output sequential data, and to achieve parallel processing by using self and multi-head *attention* mechanisms. Furthermore,

it is more ideal for applying the transfer learning approach to increase model performance than RNN-based models because of the flexible and modularized design (Lin et al., 2021; Lu et al., 2021). In addition, the Transformer model is more explainable than previous models, providing clear insights into the map-matching process (Clauwaert et al., 2020).

Our contributions can be summarized as follows:

- We design a transfer learning approach to solve labeled data sparsity problem for deep learning-based map-matching model development.
- We develop a Transformer-based map-matching model that improves model performance in terms of model accuracy, computation time, design flexibility, and explainability.
- We evaluate the performance of map-matching models using three different metrics (Average Hamming Distance, F-score, and BLEU) at the point and segment level.
- We analyze the map-matching results at trajectory-level and attention mechanism-level to improve the explainability and interpretability of model performance.

This paper is organized as follows. Section 2 presents literature review. The literature review consists of three sections: Section 2.1 presents the comprehensive review of map-matching algorithm; Section 2.3 reviews the Transformer and Section 2.2 introduces the transfer learning. Section 3 describes the methodology of the proposed model. In Section 3.1, the preliminary definitions and problems are described. Section 3.2 presents the model input, output and architecture. In Section 3.3, three metrics at different levels are introduced. Section 4 describes the performance comparison between proposed and baseline models. Section 4.1 introduces the data used in this paper, and in Section 4.2, baseline models are described for model performance comparison. In Section 4.3, the evaluation results are presented with proposed metrics. Section 4.4 presents discussion. Finally, in Section 5, conclusions and future works are presented.

2. Literature Review

2.1. Map-Matching Algorithms

According to different implementation principles and mechanisms defined in Quddus et al. (2007), the map-matching algorithms are categorized into four categories: geometric, topological, probabilistic, and advanced algorithms.

Geometric algorithms, which use geometric information from road network data to consider only the shape of the links and disregard how they are connected, become the most common approaches in the early stages of building map-matching models (Quddus et al., 2007). Bernstein et al. (1996) defined three simple geometric map-matching algorithms: point-to-point, point-to-curve, and point-to-point. Although these approaches are simple and intuitive, they are sensitive to noise, and connectivity problems exist during link transitions, resulting in unexpected results (Quddus et al., 2006; Singh et al., 2019; White et al., 2000).

Topological algorithms use topology information in the map-matching process. In GIS, topology is the term used to describe the relationship between entities (points, lines, and polygons). The stated relationship is defined by metrics like adjacency, connectivity, or containment (Quddus et al., 2007). In Greenfeld (2002), a weighted topological algorithm was proposed, which is based on topological analysis of road network. The stated method is sensitive to noise since it is dependent on observed users' coordinate information without considering any extra parameter (heading or speed). In Quddus et al. (2003), an enhanced topological map-matching algorithm that incorporates vehicle speed and heading information was developed, based on different criteria between the road network and various navigation data. In addition, a new weight-based method was developed in Hashemi and Karimi (2016), in which the weight of each GPS point was calculated based on positional accuracy, speed, and distance between observed points. Similarly, a global map-matching algorithm was proposed by considering road distance, direction, speed, and topology in Yuan et al. (2018).

The basic idea of a probabilistic algorithm is to define a confident custom error region around a vehicle trajectory to find multiple possible driving routes. It was firstly introduced in Honey et al. (1989) to match positions from a DR sensor by using a stored map database. In Bierlaire et al. (2013), a probabilistic map-matching method for smartphone

GPS data was proposed, which generates a set of potential paths, and calculates the possibility of each segment based on spatial (GPS coordinates) and temporal (speed and time) information. In Li et al. (2015), a spatial-linear cluster that clusters points based on the direction of movement was firstly performed, then for each group, ranked the possible path segments, and finally, searched for the appropriate path combination on the candidate data set. Also Zhang et al. (2016) used three possibility-related algorithms: probability-based path selection algorithm, the improved prim path selection algorithm, and the improved prim path selection algorithm based on probability to improve model performance. In the urban environment, the overall performance of probabilistic algorithms is better than geometrical algorithms and topological algorithms.

Advanced algorithms use advanced techniques such as Kalman filter (Jo et al., 1996; Kim et al., 2000), dynamic programming (Zhu et al., 2017) and Hidden Markov Model (HMM) (Newson and Krumm, 2009; Yang and Gidofalvi, 2018; Feng et al., 2020) in map-matching process. Jo et al. (1996) and Kim et al. (2000) combined the geometric method with the Kalman filter to improve the matching accuracy. In Zhu et al. (2017), trajectory segmentation map-matching with dynamic programming approach was proposed to be used in matching large-scale, high-resolution GPS data. Among the various advanced map-matching algorithms, hidden Markov model-based map-matching algorithms are widely accepted due to their applicability in sequential modeling and they consider road network connectivity. In the HMM-based map-matching models, each road segment is regarded as the state of the hidden Markov process, and noisy vehicle location measurements are considered as the state measurements (Newson and Krumm, 2009). Based on HMM, numerous developed methods are proposed. In Luo et al. (2017), an enhanced map-matching algorithm with a hidden Markov model was proposed based on the geometric data, tree topology matrix of road links, and refined quad-tree data structure. In Yang and Gidofalvi (2018), a Fast Map-Matching algorithm (FMM), which integrated the hidden Markov model with precomputation was proposed. In FMM, an upper-bounded origin-destination table is pre-computed first to store all pairs of shorted paths under a specified length, and then repeated routing queries are replaced with quick hash table searches. However, Feng et al. (2020) points out two shortcomings in HMM-based methods: they don't use historical trajectory in the map-matching process and they are susceptible to noisy data. Therefore, the researchers proposed deep learning-based map-matching models to solve the two shortcomings in HMM-based map-matching methods. The deep learning-based model (DeepMM) completes the map-matching process in the latent space, which provides a high tolerance to the noisy trajectory and improves matching accuracy with the information of historical mobile patterns.

2.2. Transfer Learning

The traditional deep learning technologies have already achieved great success in many areas and have been successfully applied in various transportation engineering (Noh et al., 2022; Choi et al., 2019, 2021). However, most deep learning models work well when abundant labeled data are used in model training. In many real-world applications, it is often expensive, time-consuming, or even impossible to collect the needed training data for model building. Semi-supervised learning can be one of the common solutions to solve the data sparsity problem in model development (Zhu, 2005). It is a method that uses massive abundant unlabeled data combined with a limited number of labeled data to train the model. Even though the semi-supervised learning methods can relax the dependence on labeled data in model training, most of them assume that the distributions of labeled and unlabeled data should be the same (Zhuang et al., 2020).

Another approach for developing a model under lack of labeled data scenario is to use Transfer learning. pTransfer learning is a machine learning approach in which a model generated for a task is reused as the basis for a model on a different task (Pan and Yang, 2009). The goal of transfer learning is to improve the target learners' performance on the target domains by transferring information from various but related source domains (Zhuang et al., 2020). In this way, the dependence on a large number of labeled training data can be reduced. In addition, the data distributions of source and target data are usually different, which increases the feasibility to use in various scenarios. In transfer learning, based on the target data size and similarity with the source dataset, there are four major scenarios - target dataset is large but different from the training dataset, target dataset is large and similar to the training dataset, target dataset is small and different from the training dataset, and target dataset is small but similar to the training dataset (Yosinski et al., 2014; Pan and Yang, 2009). However, if the relevance between the source and target domains is low, there will be negative effects on the target model (Zhuang et al., 2020).

Recently, to develop a deep learning model with a limited amount of labeled data, the transfer learning approach that pre-training the deep learning model with generated data that has high relevance with target data, and fine-tuning the pre-trained model with a limited amount of labeled data has recently gained popularity in a variety of fields (Grund-

kiewicz et al., 2019; Bird et al., 2020; Kortylewski et al., 2018; Tremblay et al., 2018). In Tremblay et al. (2018), the authors proposed a deep neural network for object detection using synthetic images. They showed that it is possible to produce a better performing model by combining a synthetic data-based pre-trained model and a real data-based fine-tuning method. In Kortylewski et al. (2018), the authors claim that synthetic data can be used to develop the deep learning model for solving simple face recognition problems, however, for complex face recognition problems, a significant real-to-virtual gap exists. To solve the gap problem, in Kortylewski et al. (2018), they demonstrate that the real-to-virtual gap is reduced by fine-tuning the synthetic data pre-trained model with real data.

2.3. Transformer

Developing a deep learning-based map-matching model can be treated as building a sequence to the sequence learning model. The sequence-to-sequence learning (Seq2Seq), which is mostly used in machine translation, is the process of training models to convert sequences from one domain to sequences in another domain (Sutskever et al., 2014; Neubig, 2017). In transportation engineering, it is used in traffic flow, next location/destination and vehicle trajectory prediction (Hao et al., 2019; Choi et al., 2018, 2019). Recurrent neural networks(RNN) are widely adopted because of their capacity to selectively pass information across sequence phases while processing sequential input one element at a time (Lipton et al., 2015). However, long-dependency problems exist in simple RNN-based sequential learning structures, which are known as encoder-decoder structures. The reason behind this is that the encoder-decoder approach requires a neural network to be able to compress all of the necessary information from a source sentence into a fixed-length vector. Also, it has limited capability to capture the intercorrelation among data points. In order to address the issue, an encoder-decoder model with attention mechanisms is proposed in Bahdanau et al. (2014). Even if the stated method solves the stated problems, it is difficult to train the model in parallel owing to the properties of RNN cells. In Vaswani et al. (2017), Transformer was introduced to overcome the long-dependency problems and the model makes it possible to train the model in parallel. The main characteristic of Transformer in machine translation is non-sequential. Transformer processes a sentence as a whole rather than word by word, reducing the risk of losing past information from the sentence. Therefore, the aforementioned problems that affect RNN-based seq2seq models are solved in the Transformer model. In Section 3.2, we will further introduce the key modules of the Transformer in detail.

Even though the Transformer was originally designed for machine translation, it has been widely used not only in Natural Language Processing (NLP) but also in various fields such as audio applications and Computer Vision (CV). In Natural Language Processing (NLP), a variety of Transformer variants have been applied, such as machine translation (Wang et al., 2019) and language modeling (Keskar et al., 2019; Vig and Belinkov, 2019). Especially, with the great success of Transformer-based pre-training models, the Transformer has become the state-of-the-art architecture in NLP (Lu et al., 2021; Vig and Belinkov, 2019). In Computer Vision (CV), the Transformer with its variants have been applied to solve various tasks, such as image classification (Chen et al., 2021; Liu et al., 2020), object detection (Carion et al., 2020; Zhu et al., 2020) and image generation (Parmar et al., 2018; Ramesh et al., 2021). In audio applications, various types of Transformers are used to solve the tasks such as speech recognition (Dong et al., 2018; Gulati et al., 2020), and speech synthesis (Li et al., 2019; Zheng et al., 2020). In addition, the Transformer is also capable to process multiple modalities and datasets. Therefore, it is also adopted in multimodal applications, e.g. text-to-video retrieval (Dzabraev et al., 2021; Gabeur et al., 2020) and speech-to-text translation (Di Gangi et al., 2019). Besides, the Transformer model can increase the model and result explainability due to the usage of attention mechanisms (Yang et al., 2020; Li et al., 2021).

In conclusion, the above methods demonstrated the feasibility of developing a Transformer-based map-matching model with limited labeled data. From the previous studies related to transfer-learning, we conclude the approach of pre-training with generated data and fine-tuning with a limited amount of labeled data can solve the labeled data sparsity problem in the development of deep learning models. Also, according to the previous studies on Transformer, the model can complement the limitation of seq2seq models and improve the performance in terms of model accuracy, computation time, design flexibility, and explainability.

3. Methodology

In this paper, the main research objective is to develop a high-performing deep learning-based map matching model with a limited labeled data environment. To solve the lack of labeled data for model development, we use the transfer-learning approach that consists of pre-training with generated data and fine-tuning with limited labeled data. As stated in Section 2.2, transfer learning is one of the efficient methods to solve the data sparsity problem in deep learning model development. In addition, the Transformer is adopted instead of the previous RNN-based seq2seq model in map-matching model development since it shows high accuracy with less computational costs. In addition, the Transformer has a flexible and explainable architecture, which is more suitable to apply the transfer learning approach to. To introduce our model in detail, in Section 3.1, we first introduce the preliminaries of our study. Then in Section 3.2, we will explain our map-matching model. In Section 3.3, the related evaluation metrics in this research are introduced.

3.1. Preliminaries

In this subsection, the terms, symbols and definitions used in this paper are introduced.

Definition 1 (GPS trajectory): A GPS trajectory Tr is a sequence of chronologically ordered GPS points $Tr : p_1 \rightarrow p_2 \rightarrow ... \rightarrow p_n$. Each point p_i has information on its GPS coordinates $(longitude, latitude)_i$ and timestamp t_i. Optionally, speed and heading information can be added. In this research, we only require chronologically ordered GPS coordinates without timestamp or other information.

Definition 2 (Road network): A road network (also called as map) is represented by a directed graph $G = (V, E)$, where a vertex $v = (x, y) \in V$ represents an intersection or a road end point, and edge $e = (id, start, end, l) \in E$ is a directed road that starts from vertex *start* to vertex *end* along polyline l with unique id.

Definition 3 (Point-level route): A point-level route R_P is a sequence of matched road segments, *i.e.*, $R_P : e_1 \rightarrow e_2 \rightarrow ... \rightarrow e_n$ where $e_i \in E, 1 \leq i \leq n$. The length of matched segment is same as the input trajectory length, *i.e.*, $length(Tr) = length(R_P)$.

Definition 4 (Segment-level route): A segment-level route R_S is expressed as a sequence of connected road segments, *i.e.*, $R : e_1 \rightarrow e_2 \rightarrow ... \rightarrow e_m$ where $e_i \in E, 1 \leq i \leq m, m \leq n$, and $e_i.end = e_{i+1}.start$. The length of route is less than input trajectory's (Tr) length, *i.e.*, $length(R_S) \leq length(Tr)$.

Definition 5 (Labeled data): A labeled data D_{Label} in map-matching is pairs of raw GPS points with their corresponding segments, *i.e.*, $D_{Label} = (Tr, R_P)$

Definition 6 (Map matching): Map-matching $MM_G : Tr \rightarrow R_{P/S}$ is the process of finding the point or segment-level route $R_{P/S}$ based on the GPS trajectory Tr in a given road network G. In other words, map-matching is the process of converting input GPS trajectories into the corresponding road segments.

3.2. Map-matching Model

We develop a Transformer-based map-matching model and propose a training approach based on transfer learning in this study. The proposed map-matching method consists of two main approaches: (1) Transfer-learning approach for model training with limited labeled data, and (2) Transformer-based map-matching model development. The related framework of our map-matching models is shown in Figure 1. In Section 3.2.1, we first generate various training datasets with different parameters for model pre-training. Then, the limited labeled data is used for fine-tuning the pre-trained map-matching model to improve matching performance and reduce the real-to-virtual gaps that exist between generated and real-world trajectories. In Section 3.2.2, we explain how the Transformer is used in our map-matching problem. In the following sections, we introduce each approach in detail.

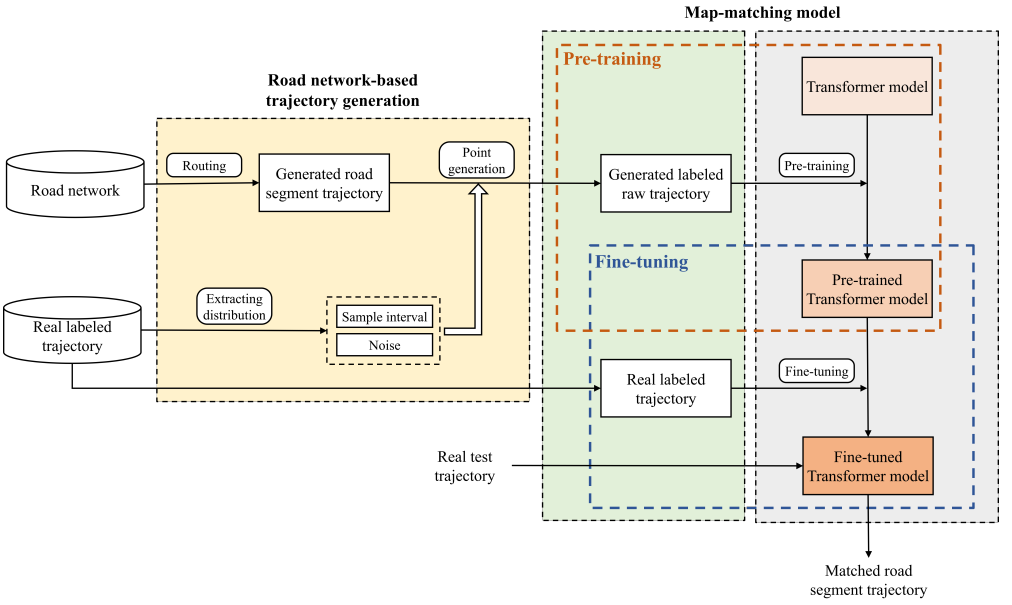


Figure 1: The framework of proposed map-matching model

3.2.1. Transfer Learning Approach for Model Training with Limited Labeled Data

As stated earlier, collecting a great number of labeled data for model training is often expensive and laborious (Kortylewski et al., 2018). Therefore, one of the biggest challenges in deep-learning based map-matching development is building a high-performing model with limited labeled data. In this research, we combine data augmentation (or data generation) method and the transfer-learning approach to overcome the challenge and develop a high-performing map-matching model. The data augmentation method is used to generate a great number of road network-based trajectory data for model pre-training, while the limited labeled data is used in model fine-tuning to reduce the real-to-virtual gaps. The concepts of pre-training and fine-tuning are from transfer learning. In computer vision (Kortylewski et al., 2018; Tremblay et al., 2018; Namozov and Im Cho, 2018), the researchers have successfully used stated methods and have demonstrated the potential to develop the high-performing model using generated data and limited labeled data. In this section, we introduce how we built our model via data augmentation (data generation) method and transfer-learning approach.

Pre-training with Generated Trajectory Data In map matching, data augmentation or data generation methods are used to solve the data sparsity problem. The data augmentation methods in trajectory generation are categorized as rule-based and data-based models. The rule-based augmentation methods are defined as generating trajectories based on pre-defined rules. Most researches focus on generating trajectories based on the shortest paths between two locations since they are simple, intuitive, and fast. However, people do not always choose the shortest path in reality, for example, drivers may choose a longer path due to short travel time or inevitable situations. Therefore, it is necessary to consider various scenarios to improve the matching performance of the deep learning model. Different from rule-based

trajectory augmentation methods, the data-based methods generate trajectories based on the characteristics of known data. Typical algorithms are data duplication (Travis and Bevly, 2008), Markov chain (Chen et al., 2011), Generative Adversarial Network (GAN) (Wang et al., 2021), and Generative Adversarial Imitation Learning (Choi et al., 2021). However, these methods need a sufficient number of labeled data to cover the various scenarios in the target area, which is hard to apply in our map-matching task.

In this study, our goal is to develop a high-performing map-matching model with limited labeled data. Therefore, rule-based trajectory algorithms are more suitable since the performance of trajectory generation does not strongly rely on the size of collected data. We propose a rule-based trajectory generation method based on the road network data, which is cost-effective and can generate various scenarios. The proposed trajectory generation method is divided into four steps: ***Route Generation, Point Generation, Point Selection, and GPS Trajectory Generation***.

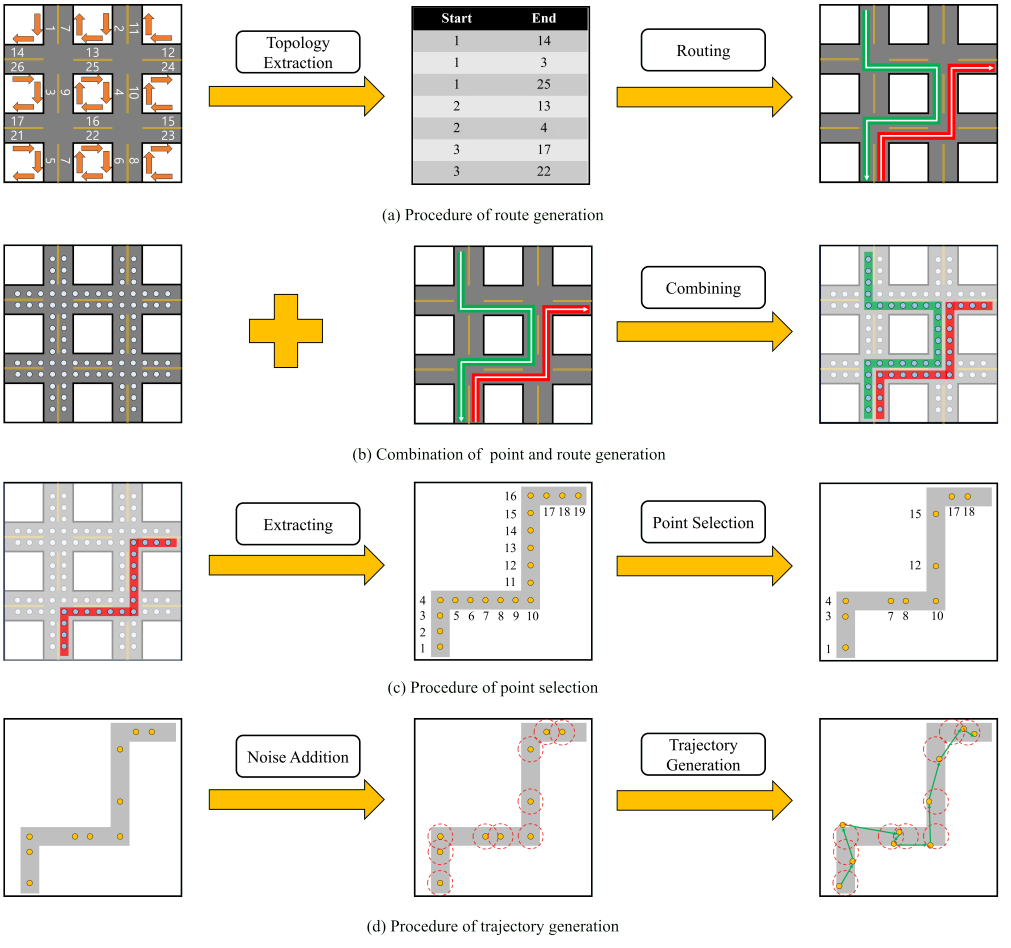


Figure 2: Overview of data generation architecture

- ***Route Generation***
  First, a segment connection table is defined using topological information from the road network. The table consists of *start* and *end* columns that the vehicles move from segment in *start* to segment in *end*. Then, all feasible routes are generated based on the constructed table. The length of the route is determined by the number of linked segments N. More complex routes can be generated when N increases. Figure 2 (a) shows the process of route generation.

- ***Point Generation***
  In the point generation step, we generate points with constant distance D on the road network. The generated

points are labeled with road segment ID and their original ID. Then we combine the information from point and route generation results to get an initial labeled GPS trajectory. The process is shown in Figure 2 (b) in detail.

- ***Point Selection***
  After generating the initial labeled trajectory, the number of points at each segment should be determined. The amount of points at each segment is mainly affected by labeled trajectory distribution. The number of points is different at each road segment due to various sampling intervals and segment lengths. Therefore, we estimate the point selection range $[r_1, r_2]$ and randomly choose points inside it to guarantee that the sampling intervals of produced points are close to real trajectories' sampling intervals. Furthermore, the order of the point selection is based on the road direction that the selected point ID in the previous step cannot be greater than the selected point ID in the present step. For example, in Figure 2 (c), if we choose a point with ID 4, we cannot choose points with IDs are 1,2, and 3 in the next step. The reason for this is that vehicles can only go ahead along the direction of the roads. After step 3, the final labeled trajectories are generated.

- ***GPS Trajectory Generation***
  In the final step, we generate raw trajectories for training the model based on the generated trajectories obtained in step 3. We add noise at each point on each trajectory to ensure that the generated raw trajectories are close to real-world trajectories. To clarify the characteristics of GPS noise, we first match each GPS point to its corresponding segment manually. Since we do not have true trajectory information, we try to find the closest point on the labeled road segment and assume it as the true position of the target point. Then, we calculate the distribution of the noise along the longitude and latitude from the differences between raw input trajectory and assigned true trajectory. To indicate the relationship between the GPS noise along longitude and latitude, Pearson's correlation coefficient (Ahlgren et al., 2003) is used and the corresponding value is 0.094 with 0 P-value. The result indicates that the noise along longitude and latitude are uncorrelated. We use Gaussian distribution to fit the noise and the related results are shown in Figure 3. The means and standard deviations of the longitude and latitude noise are $-0.427m$ and $15.653m$ and $-0.608m$ and $14.153m$, respectively. The previous research has demonstrated the point of view (Feng et al., 2020). Therefore, in this research, we also assume that the spatial noise for each coordinate follows zero-mean Gaussian distribution, which is shown as,

$$f(x_{long/lat}|\sigma^2_{noise}) = \frac{1}{\sqrt{2\pi\sigma^2_{noise}}} e^{\frac{-x^2_{long/lat}}{2\sigma^2_{noise}}} \tag{1}$$

  where $x_{long/lat}$ denotes the spatial coordination (longitude or latitude), σ_{noise} is the standard deviation of the Gaussian distribution. In this research, different generated raw trajectories with the same labeled trajectory are produced by using different σ values. Figure 2 (d) depicts the process of GPS trajectory generation.

  In conclusion, the raw trajectory data are generated by using information from the road network and labeled trajectory data. After the trajectory generation step, the generated trajectories are ready for pre-training the deep learning model.

We first pre-train our Transformer model by using generated trajectories with various sampling intervals and noise distribution. In this paper, the amount of generated trajectory data used for model training is 240,000. Additionally, the pre-trained Transformer model is composed of eight attention heads and six layers for each encoder and decoder block. When pre-training the model, the loss function is defined by cross-entropy loss between the predicted output point-level route $\hat{R_P}$ and the labeled route R_P. Via backpropagation with the Adam optimizer, we train the network with a learning rate of 0.0007 and the dropout value of 0.1.

Fine-tuning with Limited labeled Data Even if we try to generate data close to real trajectories as much as possible, there are still differences between the two datasets, which means that it is not enough to develop map-matching algorithms only depending on generated trajectory data. In trajectory generation, the datasets are generated by the specific noise distributions (σ_{noise}) and sampling intervals. However, these two factors can be different from real-world trajectories due to complex and changing communication environments. To fill the real-to-virtual gaps of two

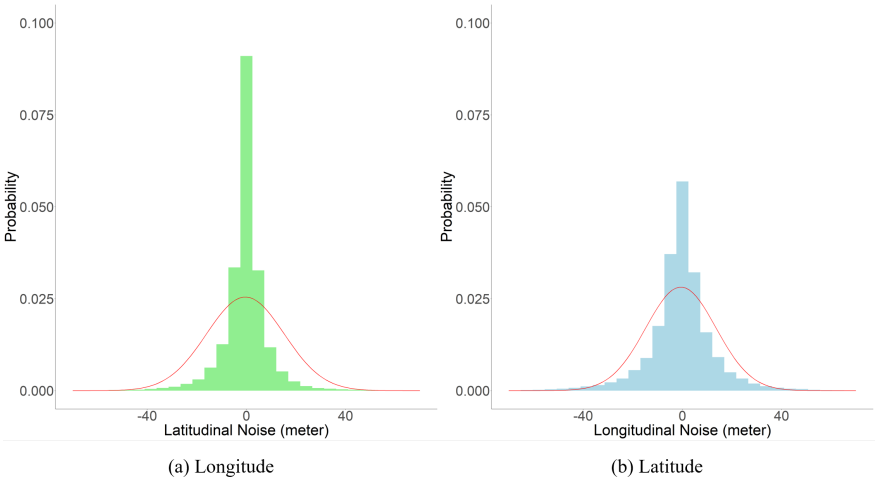


Figure 3: Spatial noise distribution of raw trajectories in labeled data

datasets, we choose to use fine-tuning method from transfer learning. In machine learning and deep learning, the term *fine-tuning* is often used to describe the optimization process of hyper-parameters during the validation step. However, in the context of "transfer learning," *fine-tuning* refers to the process of transforming a model trained on one domain (problem) to a new domain (problem). Depending on the fine-tuning dataset size and similarity with pre-trained dataset, there are four general scenarios to fine-tune the model (Yosinski et al., 2014; Pan and Yang, 2009).

(Scenario 1): For a large labeled dataset that is different from the pre-trained model's generated dataset It is preferable to fine-tune all of the model's layers since the fine-tuning dataset is large enough to train the model and significantly different from the pre-training dataset.

(Scenario 2): For a large labeled dataset that is similar to the pre-trained model's generated dataset It is feasible to freeze components of the layers to fine-tune the model since the fine-tuning dataset is similar to the pre-training dataset. Even if fine-tune the whole model, there is no risk of over-fitting since the dataset is sufficient to re-train the model.

(Scenario 3): For a small labeled dataset that is different from the pre-trained model's generated dataset It is the most difficult scenario to deal with and occurs frequently in the real fine-tuning problems. The reason is that there are significant differences between the pre-trained dataset and labeled dataset, which means that we cannot fine-tune for a small number of layers without taking risks of overfitting problems owing to the small amount of tuning dataset. In this case, it is necessary to fine-tune only an appropriate number of layers, which are difficult to control.

(Scenario 4): For a small labeled dataset that is similar to the pre-trained model's generated dataset It is one of the special cases of scenario 3. In this scenario, we can also use the fine-tuning technique from scenario 3 that choose the appropriate layer for fine-tuning. The main difference is the number of fine-tuning layers might be less than the previous scenario since the labeled dataset is similar to the pre-training dataset.

The best scenario for fine-tuning the model is second one that there is a large labeled dataset that is similar to the pre-training dataset. Even though the labeled dataset for fine-tuning differs from the pre-training dataset, we retrain the entire model to build a high-performing map-matching model if the dataset is large enough. However, it is challenging and laborious to collect a large amount of labeled data for model training in reality. Instead, we have to use small amount of labeled dataset to build a high-performing model to solve the problems. Therefore, scenario 3 and 4 are the two feasible scenarios in this study since the goal is to develop a high-performing map-matching model using limited labeled data.

Despite the fact that both scenarios 3 and 4 require discovering appropriate layers for fine-tuning the model, the difficulties of implementing fine-tuning are different. In other words, scenario 4 is considerably easier than scenario 3 since the real-world trajectories in the former situation are close to the pre-trained model's generated dataset. As a result, having prior information of the real-world trajectories helps us in the generation of more realistic trajectories and reduces fine-tuning challenges. However, there are situations when we are unable to obtain any information of real trajectory data, making it more difficult to build a high-performing map-matching model. In other words, it is hard to

generate a realistic training dataset, which increases the difficulties in the fine-tuning process. As a result, it is preferable to consider both situations and generate two different datasets for pre-training, one of which is similar to the real-world instance and the other completely different, and then use the limited labeled dataset for fine-tuning. Furthermore, throughout comparing the performance of two models that are pre-trained with different generated trajectory data and fine-tuned with the same labeled data, we can demonstrate the importance of prior knowledge of real-world trajectories and determine how much fine-tuning improves model performance.

In this study, we choose four fine-tuning components in Transformer: ① output module, ② normalization layers in encoder and decoder modules,③ full encoder modules, and ④ full decoder modules. The red circled numbers in Figure 5 depicts the fine-tunable components. To identify the appropriate fine-tuning layers for both cases, we fine-tune each module individually first, then gradually increase the number of tuning modules. The corresponding results are introduced in Section 4.3.2 in detail.

3.2.2. Transformer-based Map-matching Model

In this section, we explain the input and output structure of the proposed map-matching model. Then, spatio-temporal feature extraction method of input GPS trajectory is introduced. Finally, the architecture of map-matching model is presented. The Transformer-base map-matching process is shown in Figure 4

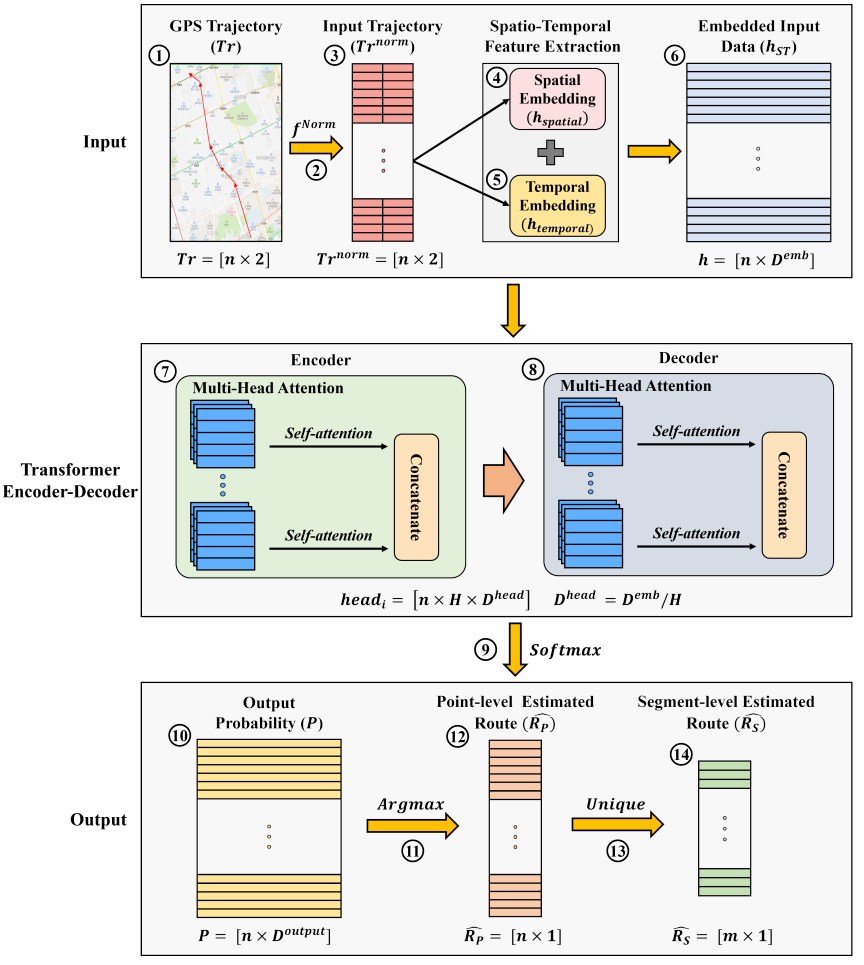


Figure 4: Transformer-based map-matching model process

Model Input and Output As discussed earlier, the input of map-matching model is the GPS trajectory (Tr) and the output is the point or segment-level route ($R_{P/S}$). Each GPS trajectory contains n GPS points.

$$Tr = [p_1, \cdots, p_n] = \left[\left(lat_1, long_1\right), \cdots, \left(lat_n, long_n\right)\right] \tag{2}$$

Instead of using raw GPS points, in this study, we use a normalized GPS trajectory to make the training faster and reduce the possibility to get stuck in local optimal solution (Sola and Sevilla, 1997). The normalized GPS trajectory is denoted as Tr^{norm} as shown in Eq.3.

$$Tr^{norm} = f^{norm}(Tr) = \left[\left(f^{norm}(lat_1), f^{norm}(long_1)\right), \cdots, \left(f^{norm}(lat_n), f^{norm}(long_n)\right)\right] \tag{3}$$

The normalization function is defined as

$$f^{norm}(X) = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{4}$$

where X represents the GPS coordinate longitude or latitude, $X_{max} and X_{min}$ are the maximum and minimum longitude or latitude values in the target network. In Figure 4, steps 1-3 present the normalization process from GPS trajectory (Tr) to input trajectory (Tr^{norm}).

The first output of the proposed model is the point-level estimated route $\hat{R_P}$, which contains n matched edges $\hat{e_n}$ for each GPS point. In Figure 4, steps 3-12 shows the map-matching process from input trajectory (Tr^{norm}) to point-level estimated route ($\hat{R_P}$).

$$\hat{R_P} = MM_G(Tr^{norm}) = [\hat{e_1}, \cdots, \hat{e_n}] \tag{5}$$

To further obtain the segment-level estimated route $\hat{R_S}$, the unique values $\hat{e_m}$ are chosen in the point-level estimated route $\hat{R_P}$ without sorting since the order of the value provides the vehicle's traveling direction information. In Figure 4, steps 12-14 show the process of obtaining $\hat{R_S}$ from $\hat{R_P}$.

$$\hat{R_S} = Unique(\hat{R_P}) = [\hat{e_1}, \cdots, \hat{e_m}] \tag{6}$$

Spatio-Temporal Feature Extraction The GPS trajectory is one of the spatio-temoral data since it includes both spatial and temporal information. In map matching, the spatial information of input trajectory is used to detect the location of the GPS points and find noise distribution, while the temporal information helps us comprehend the order of the GPS points, indicating the moving direction of the vehicle. As a result, it is crucial to extract both features properly in the map-matching process. In this research, learned positional embedding (Gehring et al., 2017) is applied for temporal information extraction, while a novel GPS embedding method is used to extract spatial features of the input trajectory.

- **Spatial Feature Extraction**
  In deep learning, it is common to use feature-extraction layers to convert input variables into feature vectors. In map matching, it is necessary to extract the spatial features from input trajectories to train the deep learning model. Previous researches that used deep learning for map-matching problems discretized the road networks into zones and used embedding or one-hot encoding to extract the spatial features (Zhao et al., 2019; Feng et al., 2020). However, using discretized road network is ineffective in map-matching tasks as this method can lead to information loss of our input data. In other words, the important features of trajectory such as the noise characteristics, moving direction, and distance between GPS points will be neglected if we choose the previous method. As a result, in this study, we use multiple fully connected layers to prevent data loss and extract spatial features of input data.

$$h_{spatial} = FC(Tr^{norm}) \tag{7}$$

- **Temporal Feature Extraction**
  In the Transformer, additional positional representation, also known as *positional embedding* is required to model the temporal features of input GPS data since the positional information of input data is ignorant. In this research, the positional embedding is defined as,

$$h_{temporal} = Embedding(Arrange(len(Tr^{norm})) \tag{8}$$

  where $Embedding$ is the dense representation that converts discrete position feature to continuous vector form, $Arrange(X)$ generates the integer number from 0 to X-1, and $len(Tr^{norm})$ represents the length of trajectory. Vaswani et al. (2017) demonstrates that the results are similar between cosine function-based positional encoding and learned positional embedding which is used in this research.

After extracting spatial features from Eq. 7 and temporal features from Eq. 8, we combine these two features to get a spatio-temporal feature representation of input data. The corresponding expression is shown as,

$$h_{ST} = h_{spatial} + h_{temporal} \tag{9}$$

where both of them have same dimension D^{emb}, which is dimension of embedding. In Figure 4, steps 3-6 present the process of spatio-temporal feature extraction of input trajectory(Tr^{norm}).

Transformer Architecture In the proposed map-matching model, there are two reasons for choosing Transformer to deal with map-matching problems. The Transformer can capture the internal correlation of the GPS trajectory and the external correlation between the GPS trajectory and the output route (or segment-level route R_S). The correlations are primarily captured by *attention modules* in both encoder and decoder (Lu et al., 2021). In the encoding stage, the attention modules capture the internal correlation of GPS trajectory, which helps to understand the relationship between each GPS points in the trajectory and reduces the effects of uncorrelated GPS points in map-matching processes. Similarly, the attention modules in the decoding stage extract the relationship between the input GPS trajectory and the output route to enhance matching performance and to analyze and interpret the matching result. Therefore, the matching performance at confusing regions such as the initial segment, last segment, and the transition area between two consecutive segments can improve due to stated characteristics.

The architecture of the Transformer is shown in Figure 5. It mainly consists of four components: input embedding, encoder, decoder, and output modules. The input embedding module includes spatial and temporal embedding. It extracts the spatio-temporal features of input trajectory, which has been introduced previously. The encoder and decoder modules are mainly composed of multi-head attention, normalization layers, and position-wise feed-forward networks. The encoder module is used to understand spatio-temporal features of GPS trajectory data, and generate a representation (h_{enc}) for the observation sequence based on the extracted information of input data (h_{ST}). Conversely, the decoder module converts encoded information from the encoder module to the target output sequence. The output module also functions as a classification module, classifying the input GPS points into segments. The output module composes a fully connected layer. In the next, three main components in encoder and decoder modules are introduced.

- **Self and Multi-head Attention**
  In self-attention model, query matrix Q, key matrix K and value matrix V have dimensions $d = d_k = d_q = d_v$, respectively. The attention equation used in Transformer is shown as,

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = Softmax(\frac{\mathbf{QK}^T}{\sqrt{D_k}})\mathbf{V} = \mathbf{AV} \tag{10}$$

  where $\mathbf{A} = softmax(\frac{\mathbf{QK}^T}{\sqrt{D_k}})$ is attention matrix. To alleviate gradient vanishing problem of softmax function, the dot-products of queries and keys are divided by $\sqrt{D_k}$.

  Instead of simply applying a single attention function, it has been found that it is beneficial to map the queries, keys, and values for H times to learn different contextual information respectively. In other words, the dimension

of each head d is equal to $\frac{D^{emb}}{H}$, *i.e.*, $D^{head} = d = \frac{D^{emb}}{H}$. The self-attention function is performed on each projected version of queries, keys, and values in parallel. Then the results from each self-attention function are concatenated and projected again to obtain the weight of final values. The aforementioned process is defined as multi-head attention, which is shown as,

$$\begin{aligned} &MultiHeadAttn(\mathbf{Q},\mathbf{K},\mathbf{V}) = Concat(head_1, head_2 ... head_H)\mathbf{W}^O \\ &head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{11}$$

where W_i^Q, W_i^K, W_i^V, W^O are the projections of parameter matrices in queries, keys, values, and output, respectively.In Figure 4, steps 7-8 presents the multi-head attention process of encoder and decoder.

- **Normalization layer**
  In the encoder and decoder modules, the Transformer uses a residual connection (He et al., 2016) around multi-head attention and position-wise feed-forward network, followed by Layer Normalization (LN) (Xu et al., 2019; Ba et al., 2016). LN is defined as

$$\begin{aligned} &LayerNorm(\mathbf{X}) = \mathbf{g} \cdot N(\mathbf{X}) + \mathbf{b} \\ &N(\mathbf{X}) = \frac{\mathbf{X} - \mu}{\sigma} \\ &\mu = \frac{1}{D^{emb}} \sum_{i=1}^{H} x_i \\ &\sigma = \sqrt{\frac{1}{D^{emb}} \sum_{i=1}^{D^{emb}} (x_i - \mu)^2} \end{aligned} \tag{12}$$

  where $\mathbf{X} = (x_1, x_2 \cdots x_{emb})$ is the input vector with size D^{emb}. μ and σ are the mean and standard deviation of input. $\mathbf{b}$ and $\mathbf{g}$ are the trainable parameters with the same dimension D^{emb}. LN is considered as a mechanism, which is effective at stabilizing the hidden state dynamics in recurrent networks (Ba et al., 2016).

- **Position-wise Feed-Forward Network**
  The position-wise feed-forward network consists of two linear transformations separated by a RELU activation. Each network in the encoder and decoder modules operates separately and identically in each position. The equation is defined as,

$$FFN(x) = ReLU(W_1 x + b_1)W_2 + b_2 \tag{13}$$

  where x is the outputs of previous layers, and W_1, W_2, b_1, b_2 are trainable parameters. Even though the network is simple, it is important for the Transformer to achieve good performance since it can prevent rank collapse problems, which can occur when the self-attention are simply stacked (Lin et al., 2021).

After getting results from decoder modules, we use the output module to do the classification. The output module uses a fully connected to change the dimension from D^{emb} to D^{output}, where D^{emb} is the dimension of embedding and D^{output} is the target output dimension of our model. In this research, D^{output} = number of link +1 due to padding tokens in input trajectory. In addition, the $Softmax$ function is used to calculate the matching probability of each GPS point at each segment. The corresponding process is shown as,

$$P = Softmax(FC(h_{dec})) \tag{14}$$

where h_{dec} represents the result from decoder modules. FC is the fully connected layer, P output probability and $Softmax$ represents the softmax function. The dimension of P is same as input data Tr^{norm}. In Figure 4, steps 8-10 depict the process of calculating output probability (P) from model output.

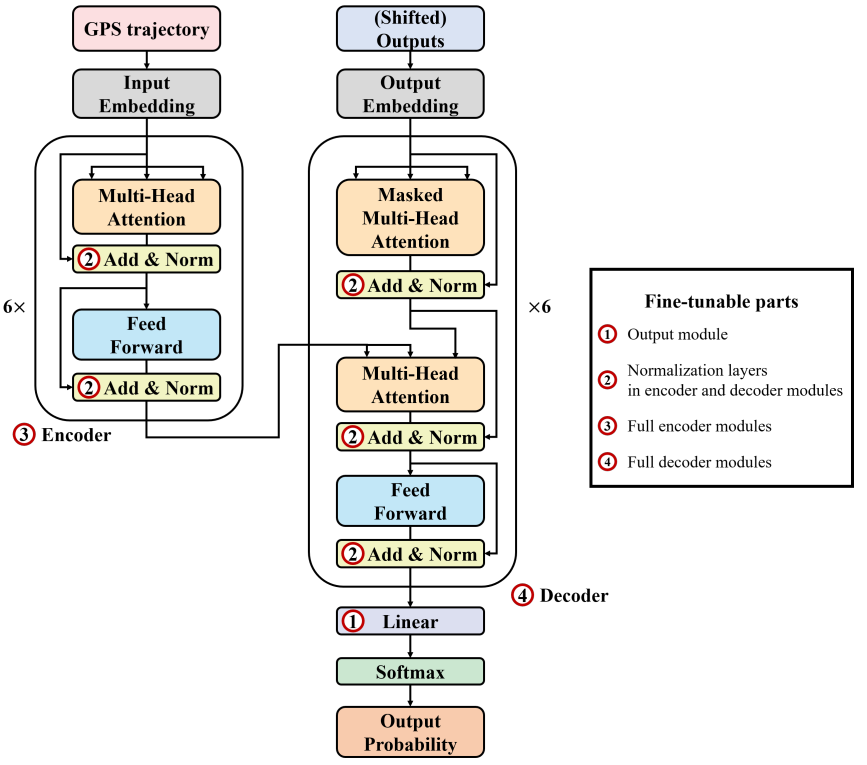


Figure 5: Overview of the Transformer architecture

After obtaining output probability P, we use function $argmax$ to get the point-level estimated route ($\hat{R_P}$). The corresponding expression is shown as,

$$\hat{R_P} = argmax(P) = [\hat{e_1}, \cdots, \hat{e_n}] \tag{15}$$

where $\hat{e_n}$ indicates the best matched segment for each GPS point. In Figure 4, steps 10-12 depict the process of calculating the point-level estimated route ($\hat{R_P}$) from output probability (P).

3.3. Evaluation Metrics

We use three metrics for evaluation - Average Hamming Distance (AHD), F-score, and BLEU at point and segment level to compare and evaluate the overall performance of map-matching models. The point-level matching analyzes the matching result at the point-level route(R_P). The point-level matching is beneficial to find how the map-matching model matches each point accurately to its corresponding segment. Most previous researches, particularly in the field of online map-matching models, have used various point-level approaches to evaluate their map-matching model. Even though it is crucial to properly match each point, route integrity is also an essential factor to consider in map-matching tasks, especially in offline map-matching models. The route integrity refers to how completely the map-matching models match the GPS points to road segments at trajectory level. For example, suppose the labeled result at point-level route (R_P) is [98, 98, 98, 97, 97, 97, 1, 1, 3, 3], and the candidates are [98, 98, 97, 97, 97, 1, 1, 1, 3, 3] and [98, 98, 107, 97, 97, 1, 1, 183, 3, 3]. Although both results have a matching accuracy of 0.8, the first one is better in route integrity than the second. As a result, not only map accuracy but also route integrity should be taken into account. We define segment-level matching, which focuses on matching results at the segment-level route (R_S). We select the unique value in the matching result without changing the order. In the previous example, the segment-level labeled result is transformed into [98, 97, 1, 3]. Segment-level matching is beneficial for minimizing the impact of confusing points in map-matching results. In practice, it is difficult to annotate the points manually at some regions where there are multiple segments close to the target matching points on the actual path (Taguchi et al., 2018). Therefore, we need to

analyze the result in segment-level matching. As previously stated, the labeled result at segment level is [98, 97, 1, 3] and the two candidates are converted as [98, 97, 1, 3] and [98, 107, 97, 1, 183, 3], respectively. In this situation, the former's matching accuracy is 1 and the latter's should be transformed first by using the sequence alignment method for accuracy-calculation.

Therefore, we evaluate the model performances with three metrics at point and segment level to consider the point matching accuracy and route integrity. The Average Hamming Distance (AHD) and F-score are used to calculate the accuracy and the Bilingual Evaluation Understudy (BLEU) score is useful to consider the order of result sequence and its integrity.

Average Hamming Distance (AHD) In information theory, the Hamming distance measures the number of positions in which the corresponding symbols are different between two equal-length strings (Hamming, 1986; Bookstein et al., 2002). Average Hamming Distance is defined as the total number of different symbols divided by the length of string (Shutao and Fu, 1998; Zhang, 2001). In map-matching evaluation, we transform the Average Hamming Distance to calculate the model accuracy. The equation is shown as,

$$AHD_{point/segment} = \frac{\sum \#\,of\ matched\ points/segments}{\sum \#\,of\ points/segments} \tag{16}$$

However, in segment-level matching, the length of the result and labeled sequences are not always identical. For example, the result sequence is [7, 8, 9], whereas the true sequence can be [7, 8, 9, 10]. Therefore, to calculate the accuracy at the segment level, it is necessary to implement sequence alignment algorithms. In this paper, the Needleman-Wunsch algorithm, which is an algorithm used in bioinformatics to align protein or nucleotide sequences is adopted to balance the length of the result and labeled segment sequences Likic (2008).

F-Score The second performance evaluation metric is the F-score, which is one of the most commonly used measurements of a model's accuracy Sokolova et al. (2006). It is used to evaluate binary classification systems, which classify examples into 'positive' or 'negative'. The traditional F-score (or $F_1 score$) uses the harmonic mean of precision and recall of the model. The related equation is shown as,

$$F_{score} = \frac{2}{\frac{1}{recall} \cdot \frac{1}{precision}} = 2 \times \frac{precision \times recall}{precision + recall} \tag{17}$$

BLEU score BLEU (Bilingual Evaluation Understudy) score (Papineni et al., 2002) is one of the most commonly used metrics in machine translation and sequence to sequence learning problems. Recently, BLEU is applied as a measurement of effectiveness in trajectory-based researches (Choi et al., 2021; Sun and Kim, 2021). In machine translation, BLEU scans the reference sentences to see if the translated sentences include the same words or contiguous sequence of n elements. BLEU uses a modified form of precision to compare reference sequences and a candidate output sequence by using the clipping method. The number of each chunk is clipped to a maximum count (m_{max}) to avoid generating the same chunks to get a high score in the output sequence. The equation of modified precision is shown as,

$$P_n = \frac{\sum\limits_{i \in C} min(m_i\, m_{i,\max})}{w_t} \tag{18}$$

where n is the number of elements considered as chunk; C is a set of unique chunks in the output sequence; m_i is the number of occurrences of chunk I in the output; $m_(i, max)$ is the maximum number of occurrences of chunk i in one of the reference sequences; and w_t is the total number of chunks in the output sequence.

$BLEU_n$ score consists of the geometric mean of P_n and a term of brevity penalty. The brevity penalty is used to prevent short candidates from getting high scores. The $BLEU_n$ is shown as,

$$BLEU_n = min(1, \frac{length_{gen}}{length_{ref,close}})(\prod_{i=1}^{n} P_i)^{\frac{1}{n}} \tag{19}$$

where $length_{gen}$ represents the length of the output sequence; $length_{ref,close}$ is the length of a reference sequence that has the closest length to the output sequence. We use $n = 3$ for evaluation because the minimum length of the sequence is 3 in the results.

4. Performance Evaluation

4.1. Data

There are two types of trajectory datasets used in model training and evaluation. Specifically, the generated dataset is used in the Transformer model pre-training, while the labeled dataset is utilized for model fine-tuning and model performance evaluation. The details of these two datasets are described as follows.

Generated data As discussed in Section 3.2.1, we use a rule-based trajectory generation method based on the road network information since it is cost-effective and can generate various scenarios. The proposed trajectory generation method is divided into four steps: Route Generation, Point Generation, Point Selection, and GPS Trajectory Generation. In trajectory generation, it is critical to make the trajectories close to the real ones since the model performance can be affected by the quality of training data. Therefore, we use the noise distribution and sample interval information from the labeled dataset to generate the trajectory. As discussed in GPS Trajectory Generation, the noise along longitude and latitude is are uncorrelated and each of them follows the zero-mean Gaussian distribution. The related figure is shown in Figure 3. Therefore, from the results, we determine the distribution of GPS noise along longitude and latitude as zero-mean Gaussian distribution with a standard deviation of 15m. To determine the sampling intervals, the number of points is different at each road segment due to various sampling intervals and segment lengths. Therefore, as we discussed in Point Selection in Section 3.2.1, instead of determining the specific sampling interval, we estimate the point selection range $[r_1, r_2]$ and randomly choose the points inside it. For example, if the point selection range is $[1, 9]$, it means that the number of points on the segments can be randomly selected from 1 to 9. This kind of approach can ensure the sampling intervals of generated trajectories are close to the real situation since the real situation. In this experiment, we set the range as $[1, 14]$, which is close to the situation of our dataset. Here, 14 is the maximum number of points that can exist on a segment in our labeled dataset. Finally, in this research, a total of 240,000 labeled trajectories were generated for model pre-training.

Labeled data A case study is designed to evaluate the performance of the proposed map-matching model, using data collected by digital tachographs (DTG) installed at taxis operating in Gangnam District in Seoul, South Korea. The DTG collects information such as driving position (longitude and latitude), speed, and passenger occupancy. The passenger occupancy information shows if there are passengers on the taxi or not, and it is used in data preprocessing. The collected data are converted into a taxi trajectory dataset by chronologically linking the data points of the same taxi ID. In this study, we choose the taxi trajectories that traveled in the Gangnam district where GPS errors occur frequently due to the complex urban environment situation. The Gangnam district consists of 228 major road segments in a grid structure as shown in Figure 6. We further divide the trajectory with the same ID into several sub-trajectories, which cover the trip for each passenger, because each taxi trajectory comprises several trips associated with various passengers. We choose the passenger-level sub-trajectories, which lengths are greater than two kilometers and average sample intervals are around 20 seconds. After the data preprocessing step, we manually label each GPS point to its corresponding segment, obtaining 1,331 vehicle trajectories with 35,127 GPS points. In our dataset, various scenarios are included such as congestion, detouring, and U-turn. In this research, we randomly select 931 trajectories from labeled labeled taxi trajectory data (70% of the dataset) for training and fine-tuning the model, while the left 400 labeled passenger-level trajectories (30% of the dataset) are used as the test dataset for evaluating models' performances.

4.2. Baseline Models

We compare the performance of our model with four baseline models.

- ***FMM*** (Yang and Gidofalvi, 2018): It provides a fast map-matching technique that is both efficient and scalable. FMM uses two techniques to incorporate the hidden Markov model: 1) pre-computing the upper bound origin-destination table to store all pairs of shortest routes; 2) using fast hash table search to replace repetitive routing queries.

- ***ST-Matching*** (Lou et al., 2009): It's widely used in GPS trajectories with low-sampled cases. It creates a candidate graph to identify the best-matched route by taking into account the road network's spatial and topological

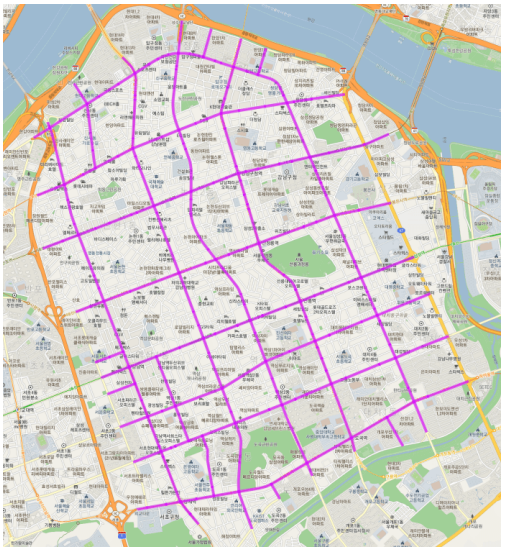


Figure 6: Road network in Gangnam district (Background: Kakao map (https://map.kakao.com))

characteristics, as well as the temporal aspects of trajectories.

- ***LSTM-based seq2seq model*** (Zhao et al., 2019): It is less affected by dense noise and it is also powerful in the urban area. In the seq2seq model, the input trajectories are compressed to context vectors in encoding states, and the road segment trajectories are translated at decoding states. The RNN cells used in both encoding and decoding states are LSTM.

- ***LSTM-based attentional seq2seq model*** (Feng et al., 2020): It is more efficient in the map-matching process since it solves the long-dependency problems existing in RNN-based seq2seq models. Specifically, if a fixed-size context vector is used, there may be diminishing problems when translating long raw trajectories, leading to the accuracy drop. To solve the problem, a dynamic context vector generated from encoder hidden states for each translation step is used to make an attention layer in the attentional seq2seq model. The LSTM cells are used in both encoding and decoding states.

4.3. Result

The results of performance evaluations for the proposed map-matching model are discussed in four different perspectives: ***1) Performance evaluation at pre-training stage, 2) Performance evaluation at fine-tuning stage, 3) Analysis on trajectory-level performance improvement, and 4) Analysis on attention mechanism in Transformer.*** In Section 4.3.1, we use generated trajectory data to pre-train the deep learning models and evaluate the matching performance using labeled testing data. We generate two trajectory datasets that have different noise standard deviations(σ_{noise}) to evaluate the noise effects on model performance. The first dataset is built without noise, while the second dataset is constructed with Gaussian distributed noise with the $15m$ standard deviation for each coordinate. In Section 4.3.2, we fine-tune two pre-trained Transformer models to see how much the method improves matching performances. As stated earlier, we randomly select 931 trajectories (70% of labeled data) for fine-tuning the model and the last 400 trajectories are used in model evaluation. Figure 5 shows four components, which are represented by red circles. The full encoder modules, for example, are used to extract features of input data, whereas the full decoder modules are used to convert the extracted information from the encoder modules to the target domain. We gradually increase the number of components for fine-tuning in order to find the best scenarios for our proposed map-matching model. At analyzing stage, we primary analyze the result at the trajectory level in Section 4.3.3 to reveal how the fine-tuning approach improves the map-matching performances and reduces the real-to-virtual gaps when compared to the pre-training model at trajectory level. Finally, in Section 4.3.4 attention mechanisms of fine-tuned Transformer model are analyzed in order to better understand the map-matching mechanisms of our proposed model.

Table 1
Pre-trained model performance at point and segment level (bar graph scaled from 0.3 to 1)

Type	Model	Data	AHD	F-score	BLEU
Point	ST-Matching	-	0.8719	0.7151	0.8548
	FMM	-	0.8573	0.7184	0.8517
	LSTM seq2seq	labeled training data	0.4829	0.3480	0.5679
		generated data (σ_{noise} = 0m)	0.6340	0.4933	0.7032
		generated data (σ_{noise} = 15m)	0.8960	0.8386	0.9029
	LSTM Attn seq2seq	labeled training data	0.6049	0.4475	0.7150
		generated data (σ_{noise} = 0m)	0.8018	0.7367	0.9004
		generated data (σ_{noise} = 15m)	0.9563	0.9372	0.9557
	Transformer	labeled training data	0.7644	0.6799	0.6585
		generated data (σ_{noise} = 0m)	0.8623	0.7860	0.8905
		generated data (σ_{noise} = 15m)	0.9737	0.9558	0.9724
Segment	ST-Matching	-	0.7408	0.6915	0.6668
	FMM	-	0.7477	0.6950	0.6775
	LSTM seq2seq	labeled training data	0.5714	0.4739	0.4546
		generated data (σ_{noise} = 0m)	0.7127	0.6004	0.6417
		generated data (σ_{noise} = 15m)	0.9156	0.8852	0.9209
	LSTM Attn seq2seq	labeled training data	0.7007	0.5947	0.6272
		generated data (σ_{noise} = 0m)	0.9027	0.8478	0.8743
		generated data (σ_{noise} = 15m)	0.9741	0.9601	0.9660
	Transformer	labeled training data	0.7068	0.5816	0.7423
		generated data (σ_{noise} = 0m)	0.9150	0.8633	0.9020
		generated data (σ_{noise} = 15m)	0.9784	0.9643	0.9751

4.3.1. Performance Evaluation at Pre-training Stage

In trajectory generation for model pre-training, we assume the spatial noise at longitude and latitude follows zero-mean Gaussian distribution, as stated in GPS trajectory generation in 3.2.1, To make more realistic trajectories, we choose $15m$ as the noise standard deviation (σ_{noise}) since the standard deviations of the longitude and latitude noise $15.653m$ and $14.153m$, respectively. We evaluate our model with four baseline models on the labeled test dataset and corresponding results are shown in Table 1. FMM and ST-Matching models do not require training procedures since they are rule-based models, and the performance of the model is evaluated based on the test dataset with 400 trajectories. Three deep learning models (LSTM-based seq2seq, LSTM-based attentional seq2seq and Transformer models) are trained on three different datasets: 1) labeled training dataset with 931 trajectories (D_{label}), 2) generated trajectory without noise ($\sigma_{noise} = 0m$) ($D_{Gen,0m}$) and generated trajectory with Gaussian distributed noise ($\sigma_{noise} = 15m$) ($D_{Gen,15m}$). Then, the performance of each model is evaluated based on the test dataset with 400 trajectories.

From the perspective of training data, the results reveal that D_{label} is not enough to build a high-performing map-matching model because it doesn't cover the whole target area and various scenarios. In addition, we found that deep learning-based map-matching models show better performances than rule-based models when they are trained with proper datasets. In other words, the quality of the training datasets affects the performance of these models. Specifically, these models show lower performances than rule-based models if the training datasets are different from the target dataset, such as $D_{Gen,0m}$. In our experiment, the quality of the dataset is highly affected by the existence of Gaussian distributed noise. The $D_{Gen,15m}$ is more close to real-world trajectory since the noise distribution used in data generation is based on real labeled data. On the contrary, $D_{Gen,0m}$ doesn't add noise, which has significant differences. The result indicates the potential to use generated data to overcome the lack of labeled data problems at the training stage. From the perspective map-matching model, two rule-based models (FMM and ST-matching) achieve similar performance in three metrics with two levels. The three deep learning models trained by generated data $D_{Gen,15m}$ show better matching performances than two rule-based models. Among three deep learning models, the LSTM-based seq2seq model shows the lower performance. Conversely, the Transformer-based map-matching model outperforms the other deep learning models. Even though the LSTM-based attentional seq2seq model has comparable performance when the training dataset is $D_{Gen,15m}$, we can show the superiority of the Transformer-based model in the other perspectives. We will discuss it in Section 4.4. Therefore, we can conclude that our proposed Transformer-

based map-matching model trained by generated data $D_{Gen,15m}$ outperforms baseline models in the map-matching process. From the perspective of model evaluation, the rule-based map-matching models perform better at point-level matching than segment-level matching. As mentioned in Section 3.3, the point-level matching result focuses on how the model matches each point accurately to its corresponding segment. Conversely, the route integrity is illustrated by the segment-level results, which indicates how well the models match the linked segments that reflect the entire trajectory. The stated rule-based models can not efficiently match the points which are located at confusing regions such as the initial segment, last segment, and the transition area between two consecutive segments. They often mismatch these points onto the wrong segments that are not related to representing the target trajectory. As a result, the segment-level results are highly affected by these wrong points, which leads to lower accuracy than point-level results. On the contrary, in deep learning-based models, the overall performances are better in segment-level results, indicating that deep-learning models are better to extract objects' routes and are ideal for developing map-matching models.

4.3.2. Performance Evaluation at Fine-tuning Stage

In the following series of experiments, we study how the real-to-virtual gaps between generated trajectories and the real labeled trajectories are reduced by using the fine-tuning method. From the previous section, we conclude that Transformer-based map-matching models show better performances among three deep learning models with three different datasets. As a result, we choose to fine-tune the Transformer-based models to get high-performing map-matching models.

In this experiment, two pre-trained Transformer models, which are trained with $D_{Gen,0m}$ and $D_{Gen,15m}$, are chosen to do fine-tuning with D_{label}.The components of Transformer for fine-tuning are depicted in Figure 5: ① output module, ② normalization layers in encoder and decoder modules,③ full encoder modules, and ④ full decoder modules. We gradually increase the number of tuning components from the output module to the entire model to analyze the effects of fine-tuning on model performances. We fine-tune the models 30 times and the averaged results are shown in Table 2 and 3.

Table 2 shows the fine-tuning results at $D_{Gen,0m}$-based pre-trained Transformer model. From the perspective of model performance, overall performance is improved after fine-tuning at both point and segment levels. To be more specific, the pre-trained model performs poorly when compared to the pre-trained model trained with $D_{Gen,15m}$ since the characteristics of the pre-trained dataset are different from the real labeled dataset. From the perspective of fine-tuning components, there are no significant performance differences except tuning the output layers. This case can be treated as scenario 3, where the size of the labeled dataset is small and different from the pre-trained dataset as stated in Section 3.2.1. In this case, it is not enough to only fine-tuning the output module to get a high-performing model. Instead, We must find appropriate fine-tuning modules to improve model performance. In this instance, fine-tuning is used to improve model performance at both point and segment levels by reducing real-to-virtual gaps.

Table 3 represents the fine-tuning results of the pre-trained model trained with $D_{Gen,15m}$. From the perspective of model performance, in the point-level result, there are minor improvements. In the segment-level result, although the improvements are small, they are noticeable when compare to the point-level findings. Also, there are no significant variations between the results of each fine-tuned component. One of the key explanations of these findings is that the pre-trained dataset exhibits similar characteristics to the real labeled dataset. Specifically, the pre-trained model shows high performance because the generated trajectories used in model training are close to real-world trajectories. Therefore, the fine-tuned results are not obvious at each component. We can consider this case as scenario 4, where the size of the labeled dataset is small but similar to the pre-trained dataset, as stated in Section 3.2.1. According to the results, the fine-tuning method in this case is prominent in improving segment-level matching performance. In the next Section 4.3.3 we will discuss the findings in detail.

There are two main conclusions throughout the results. First, the fine-tuning method is beneficial in improving model performance. Even though there are performance differences between two fine-tuned pre-trained models, both of them show high performance in the map-matching task. Especially, the pre-trained model used in Table 2 becomes a high-performing model after using the fine-tuning method. This finding shows the potential of the fine-tuning method in building a more general map-matching model with simply generated trajectories. In practice, it is difficult to generate trajectories that are close to target real-world trajectories precisely without prior information. As a result, it is important to develop a more general map-matching model that not only shows high performance but also can be built without any prior information. Second, the prior information of real-world trajectories is efficient in model development. The efficiency is demonstrated in fine-tuning process. As previously stated, $D_{Gen,15m}$ are close to real-world trajectories so that the pre-trained model outperforms the others. As a result, it is considerably easier to find suitable components

for fine-tuning the model when compared to the pre-trained model, which is shown in Table 3. In other words, if we generate datasets using prior information, we can reduce the costs of finding appropriate fine-tunable components.

Table 2
Matching results of fine-tuned pre-trained model with $D_{Gen,0m}$ (bar graph scaled from 0.75 to 1)

Type	Fine Tuned Parts	AHD	F-score	BLEU
Point	None	0.8623	0.7860	0.8905
	①	0.9182	0.8720	0.9256
	②	0.9522	0.9252	0.9540
	③	0.9489	0.9222	0.9508
	④	0.9521	0.9268	0.9544
	①+②	0.9521	0.9268	0.9538
	①+③	0.9494	0.9232	0.9512
	①+④	0.9492	0.9237	0.9511
	①+③+④	0.9449	0.9224	0.9541
	Full	0.9492	0.9237	0.9511
Segment	None	0.9150	0.8633	0.9020
	①	0.9533	0.9223	0.9437
	②	0.9742	0.9568	0.9683
	③	0.9763	0.9602	0.9708
	④	0.9762	0.9603	0.9694
	①+②	0.9717	0.9528	0.9644
	①+③	0.9761	0.9600	0.9701
	①+④	0.9757	0.9596	0.9686
	①+③+④	0.9758	0.9594	0.9688
	Full	0.9763	0.9602	0.9695

Table 3
Matching results of fine-tuned pre-trained model with $D_{Gen,15m}$ (bar graph scaled from 0.9 to 1)

Type	Fine Tuned Parts	AHD	F-score	BLEU
Point	None	0.9737	0.9558	0.9724
	①	0.9713	0.9553	0.9706
	②	0.9789	0.9660	0.9779
	③	0.9737	0.9599	0.9730
	④	0.9748	0.9621	0.9740
	①+②	0.9789	0.9660	0.9779
	①+③	0.9730	0.9596	0.9724
	①+④	0.9747	0.9622	0.9740
	①+③+④	0.9737	0.9615	0.9731
	Full	0.9737	0.9612	0.9731
Segment	None	0.9784	0.9643	0.9751
	①	0.9818	0.9696	0.9772
	②	0.9867	0.9773	0.9814
	③	0.9859	0.9761	0.9812
	④	0.9874	0.9786	0.9813
	①+②	0.9867	0.9772	0.9813
	①+③	0.9864	0.9769	0.9822
	①+④	0.9877	0.9790	0.9815
	①+③+④	0.9884	0.9801	0.9841
	Full	0.9880	0.9795	0.9837

4.3.3. Analysis on Trajectory-level Performance Improvement

In this section, we analyze the result at the trajectory level and investigate how the fine-tuning approach improves the matching performance and reduces the real-to-virtual gaps between generated and real-world trajectories. Figure 7 depicts one of the matching results of the fine-tuned map-matching model which shows the best performance from the previous section. This figure is divided into two parts: the upper portion shows road network scenarios, while the lower part of the figure presents the probability of GPS points matching at each segment. In the upper part of the figure, the red dots represent a vehicle's moving positions on the road, which are randomly distributed. The colored link represents the labeled segment, where each one has its color. The labeled segment-level route is $[64, 20, 21, 22, 23, 24]$. We calculate the probability of each moving point matching on the labeled segment and plot the associated result on the lower part of Figure 7. From the figure, we find that the fine-tuned model matches all the points and segments appropriately. In the figure, the matching probability of each GPS point on the corresponding segment is close to 1 instead of the confusing region such as transition area between two segments, first and last segment. Furthermore, our fine-tuned model performs well in the transition region. For example, the fifth GPS point is recorded between segments 20 and 21. The summation of probability matching on two segments is close to 1, indicating that the map-matching model performs well.

However, if we test the same scenario on the pre-trained Transformer model, the model can not match the first two points on segment 64. To compare the matching outcomes from pre-trained and fine-tuned models in detail, the probability results for the first two GPS points are shown in Figure 8. In Figure 8, the labeled point-level route is $[64, 64, 20]$ for the first three points. In contrast, the matching result is $[20, 20, 20]$ with a high matching probability in the pre-trained model. There is a more than 50% chance of matching the first two points at segment 20. One of the reasons behind this is the pre-trained model considers the first two points as the noise points which are generated at segment 64. To calibrate the aforementioned problems, the fine-tuning method is applied and the result sequence is turned to $[64, 64, 20]$. The probability for matching the first two points at segment 64 is improved 25.6% and 56.2%, respectively. Therefore, we can conclude that the fine-tuning method effectively reduces the real-to-virtual gaps between generated and real-world trajectories. This method the model more robust to complex real-world scenarios and produces much more realistic results.

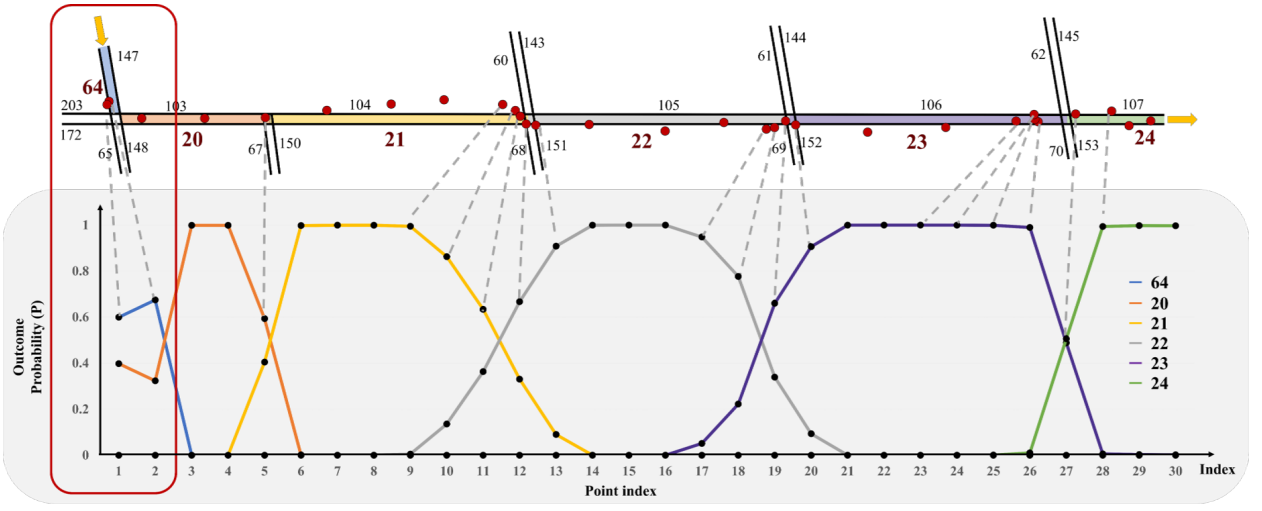


Figure 7: The matching result for fine-tuned Transformer model

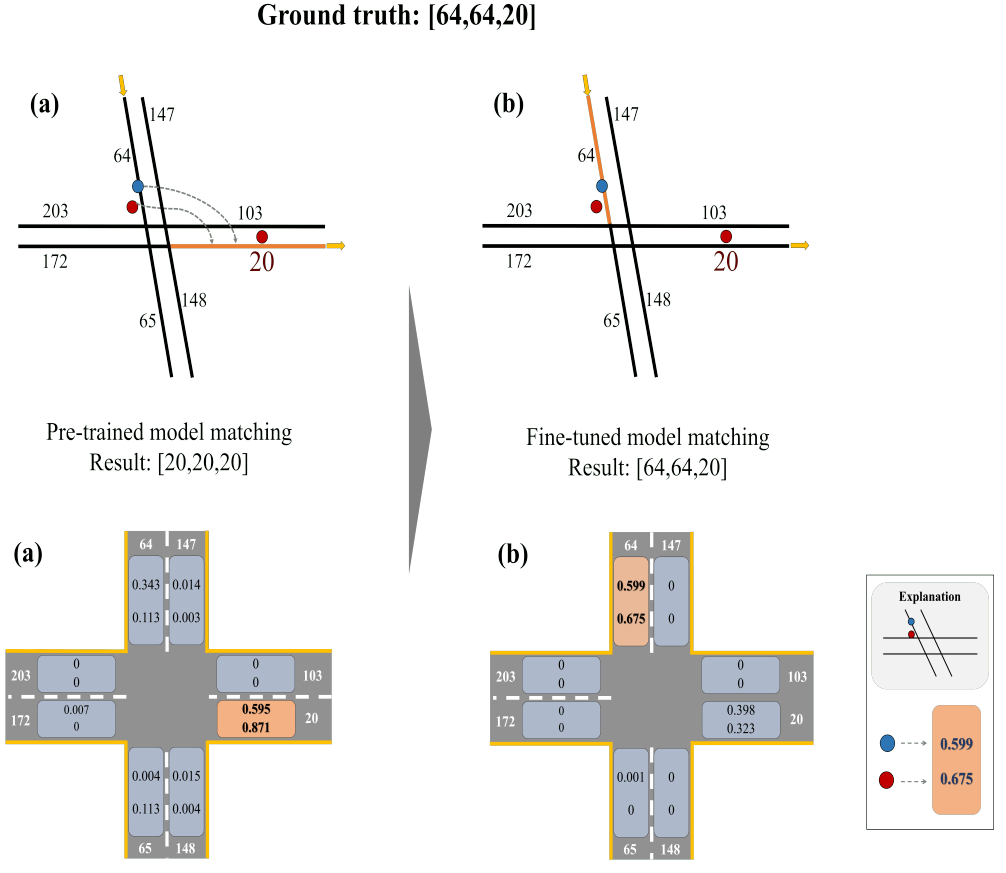


Figure 8: Amplified results for the result figure

4.3.4. Analysis on Attention Mechanism in Transformer

In this section, we use attention mechanisms to analyze the map-matching process, which shows how the model considers the internal correlation of GPS points and matches the road segments throughout the input GPS trajectories. We extract the attention weights from the encoder and decoder layers to investigate the correlation between GPS points and the relationship between GPS points and road segments, respectively.

In encoder attention weight analysis, We use the visualization tool provided by Vig (2019) to clearly see the correlations, and the result is shown in Figure 9. The upper part of the figure depicts the internal correlation of GPS points in the trajectory. We choose GPS 6, GPS 16, and GPS 26 as test points and see how the related points are changed in different GPS points. There are eight colors that represent the specific weight of each head. Here, the darker hue means a stronger correlation with the target point. The result shows that there is a certain range of GPS points that have a strong correlation with target points. Furthermore, each GPS point has its own related range. We determine the three related ranges from the upper figure and plot them in the lower figure to find the position of correlated points on the road segment in detail. In Figure, the colored boxes represent the related ranges. According to the figure, the GPS points, which are on the current and adjacent road segment, have a strong correlation with the target point. For example, in Figure 9 (b), the GPS16 has a correlation with the points from GPS11 to GPS26 and the green box stretches from the end of segment 21 to the end of segment 23.

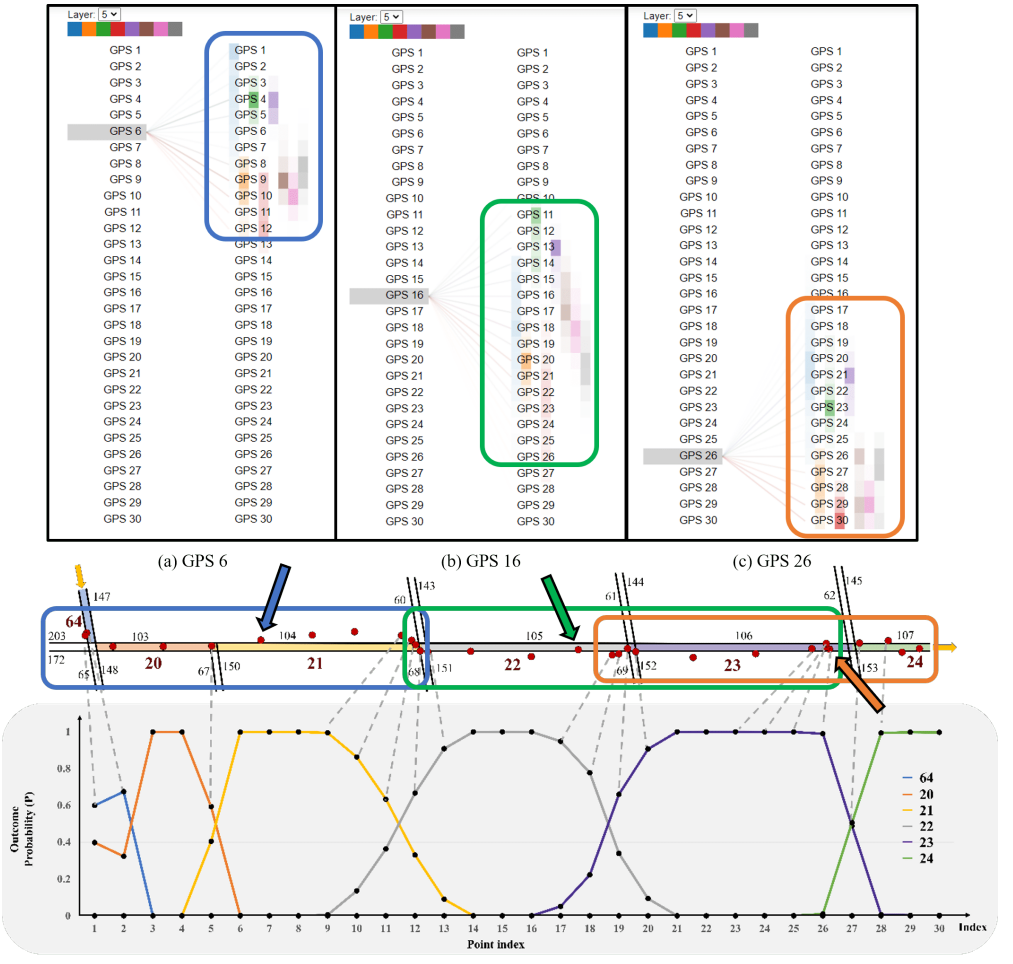


Figure 9: Internal correlation of GPS points in the trajectory (a) correlated points of GPS6 (b) correlated points of GPS16 (c) correlated points of GPS26

In decoder attention analysis, the logarithm transformation of the weights is used to plot the figure since the value of the weights are too small to depict in detail. The bright part in the figure denotes large attention weights, which can

also be interpreted as a high correlation with decoding results. From Figure 10, we find that specific ranges of points have high attention weights. To determine the range, we use the threshold value of -3.15, which is the average mean and median of total log weights. We determine the range of GPS points that affect the matching results in decoder modules and depict them in Figure 10. The stated ranges are represented as colored boxes. According to the determined ranges, we discover that in order to match the current road segment, not only GPS points on the current road segment but also GPS points on neighboring segments are involved in the matching process, which has a similar result in encoder weight analysis. For example, the yellow box in Figure 10 illustrates the range which influences the determination of segment 21. The box stretches from third to nineteenth points where they are on the segment 20,21 and 22. The result shows that while identifying segment 22, points on segments 20 and 22 have a substantial influence on the decoding process. Furthermore, since there is only one segment adjacent to the first and last segments, the points from one road segment are used in the decoding process, implying that there is less information in the matching process compared to other road segment matching. As a result, there are several matching errors in the first and end segment matching process.

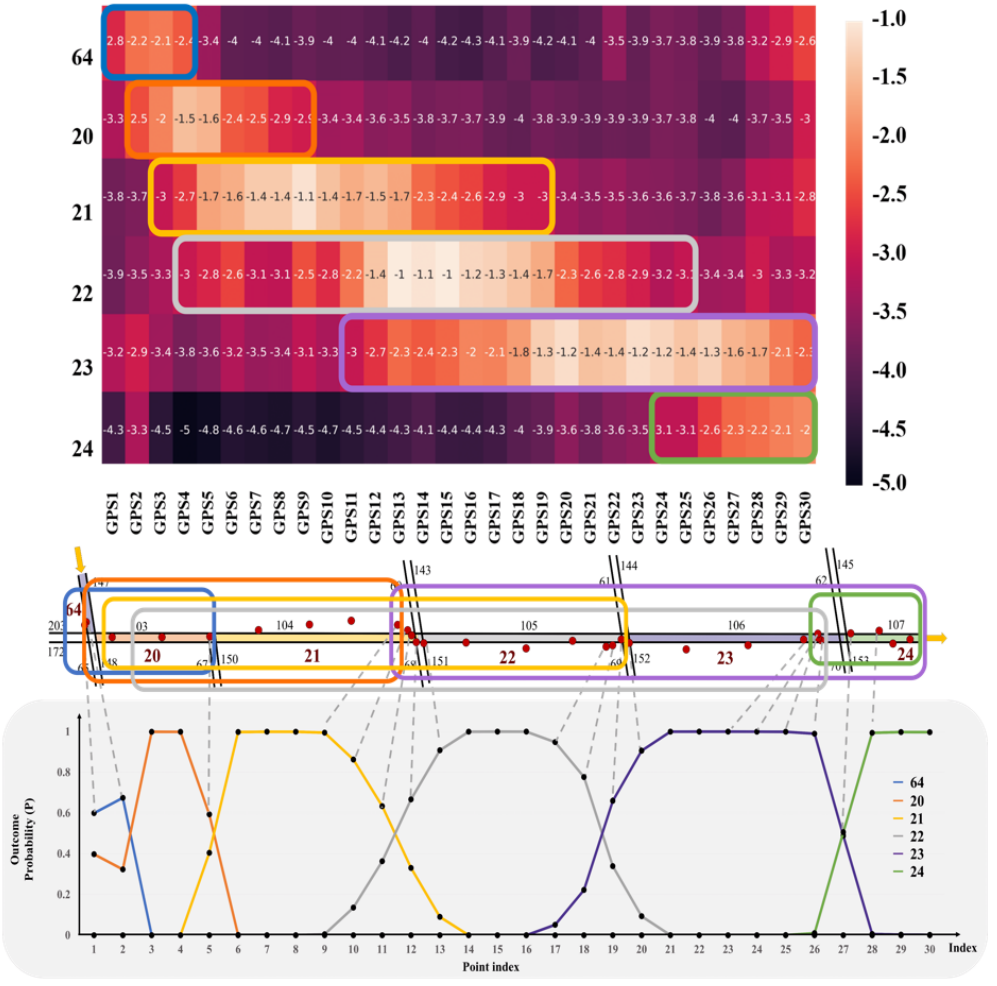


Figure 10: Plot of the decoder attention mechanisms

Table 4
The training time and number of parameters of the attnetional LSTM-based seq2seq model and Transformer

Model	Time per epoch (s)	Number of Parameters
LSTM Attn seq2seq	588.6	9,339,081
Transformer	270.4	11,281,895

4.4. Discussion

4.4.1. Performance of the Model

Among the three deep learning models, the Transformer-based map-matching model shows the best performance. The superiority of the Transformer is shown in two perspectives - robustness of the model and computational efficiency. First, the robustness characteristic is shown when the training dataset is different. Even though, the Transformer and the LSTM-based attentional seq2seq model trained with $D_{Gen,15m}$ show the comparable performance, there are significant performance differences when they are trained by the another generated dataset $D_{Gen,0m}$. Transformer outperforms the LSTM-based attentional seq2seq model in this scenario and shows more robust performance. In the real world, if we cannot generate the training dataset properly, it is suggested to use the Transformer since it is more efficient to match the GPS trajectories onto their corresponding segments accurately.

Second, we compare the training time to find the superiority of the Transformer model. The pre-training result shows that the LSTM-based attentional seq2seq model has a comparable performance with Transformer when the training dataset is generated trajectory with Gaussian distributed noise ($\sigma_{noise} = 15m$) ($D_{Gen,15m}$). To find the superiority of the Transformer model, we analyze the training time with the trainable number of parameters in the model and the result is shown in Table 4. The whole process is done with Intel(R) Core(TM) i7-6800K CPU @ 3.40GHZ, 128GB RAM, and NVIDIA TITAN Xp. In the table, "Time per epoch" shows the average time for one entire transit of the training data through the model, and "Number of Parameters" indicates the number of trainable parameters in the deep learning model. The results indicate that despite having a larger number of trainable parameters, the Transformer model spends less time in model training compared to the LSTM-based attentional seq2seq model (LSTM Attn seq2seq). The main reason is due to the parallel computation in Transformer model training. To be more specific, the LSTM-based models are restricted to compute in a series way due to the RNN cells, which are the main difference with the Transformer model. Therefore, the Transformer outperforms the LSTM-based attentional seq2seq model (LSTM Attn seq2seq) from the two stated perspectives. To clarify the mechanisms of the Transformer model, we further analyze the attention modules in the Transformer. We found that the GPS points, which are on the current and adjacent road segment, have a strong correlation with the target GPS point. Similarly, to match a segment, not only GPS points on the target segment but also GPS points on the neighboring segments are related to the matching process.

4.4.2. The Effectiveness of Fine-Tuning

From the performance evaluation results at fine-tuning stage, we conclude that the fine-tuning method can reduce the real-to-virtual gaps caused by the differences between generated and real data, which is beneficial in model performance improvement. In the fine-tuning results shown in Section 4.3.2, two pre-trained Transformer models show high performances after using the fine-tuning method. Especially, the Transformer model trained by $D_{Gen,0m}$ becomes a high-performing map-matching model. Therefore, even though the prior information of the trajectories such as GPS noise distributions helps in training data generation and model performance improvement, the fine-tuning method with limited labeled data can be one of the useful approaches to improve model performance when we cannot get any prior information related to the target trajectories. To further find the reason for performance improvement after fine-tuning, we analyzed the result at the trajectory level. We found that the fine-tuned Transformer model can match the first several GPS points onto their corresponding segments. In other words, the fine-tuning method can effectively reduce the real-to-virtual gaps. The gaps are mainly caused due to different sampling rates of GPS points, noise distribution of GPS points between generated and real data.

4.4.3. Real-World Implementation

One of the limitations of deep learning-based map-matching models is that they construct models using a great number of labeled data, which is different from rule-based models. In contrast to previous deep learning-based models, our method only needs a limited number of labeled data in model development. We used the transfer learning approach that included pre-training with generated data and fine-tuning with limited labeled data to develop the deep learning-

based map-matching models. In practice, there is no need to feed the labeled data into the model development if we can generate trajectories that are close to the real trajectories. In other words, when we neglect the Fine-Tuning step of our methodology, our approach is similar to the rule-based algorithms which does not require labeled data. To be more specific, we found that if we pre-train the model with trajectories which has similar distribution ($\sigma_{noise} = 15m$) with target data to do the map-matching, both LSTM-based attentional seq2seq model (LSTM Attn seq2seq) and Transformer show high performance. Conversely, if the pre-trained datasets are different from the target trajectory, both models show relatively low performance on the map-matching task. In practice, it is difficult to generate trajectories that are close to target trajectories precisely without prior information, which means that we cannot generate ideal trajectories for model pre-training.

Therefore, to reduce the real-to-virtual gaps, we need to feed the limited number of labeled data as input to improve model performance. If we can develop the high-performing map-matching model with only using a small amount of labeled data, it is still can be considered a cost-effective approach. Nowadays, we can also find that some companies or groups label GPS data for trajectory applications. However, the amount of labeled data is limited due to the cost problem. Therefore, if we can fully utilize the limited labeled data, which can be collected in the current stage, for developing high-performing map-matching data, the proposed model can be highly accepted in preprocessing step in the trajectory-based application.

5. Conclusion

This study proposes a framework for developing a novel deep learning-based map-matching model in the limited labeled data environment. To solve the data sparsity problem in model training, a *Transfer-learning approach*, which pre-trains the model with generated data and fine-tunes the pre-trained model with real labeled data is applied. Besides, an advanced deep-learning model *Transformer*,is used to improve model accuracy with less computation time by capturing the internal correlation of input GPS points and the external relationship between input and output.

In terms of contributions, this study has made several improvements in the field of deep learning-based map-matching models. To solve the data sparsity problem for developing a high-performing map-matching model, the transfer learning approach is adopted in this study. Specifically, a large number of trajectories are generated based on road network information to pre-train the model, and then a limited amount of available labeled data is used to fine-tune the model to reduce the real-to-virtual gaps. The proposed model shows the possibility of using generated trajectories to solve the map-matching problems in the urban environment. The proposed model improves matching performance by using the Transformer model, which considers the internal correlation of GPS points and the external relationship between input trajectory and output segments. This overcomes the disadvantages of the previous deep learning-based map-matching model that they can not take into account the data relationships and need larger computation time in model training. We also analyze the matching mechanisms of the Transformer in the map-matching process, which helps to interpret the input data internal correlation and external relation between input data and matching results. Finally, we consider the map-matching task from the data perspective and propose three related metrics at point and segment levels, which help in developing more high-performing map-matching models.

The model's performance is evaluated on the Gangnam DTG dataset, which contains moving patterns of taxis in the Gangnam district. The performance evaluation is divided into two levels: point-level evaluation and segment-level evaluation. The point-level evaluation mainly focuses on how the model matches each point to its corresponding segment correctly. Conversely, in the segment-level evaluation, the main concern is how the model matches the integrated route (or segment-level route) correctly. Both levels are evaluated in terms of AHM, F-score, and BLEU, which are widely used metrics in sequence modeling. At the pre-training stage, the results show that the generated data-based pre-trained models show better performance than rule-based models (FMM and ST-matching). In addition, the Transformer-based map-matching model outperforms other deep learning-based models (LSTM-based seq2seq and LSTM-based attentional seq2seq models) in three different datasets. At the fine-tuning stage, we fine-tune the two pre-trained Transformer-based map-matching models trained by different generated datasets. The results indicate that fine-tuning method can reduce the real-to-virtual gaps in both models. Specifically, in the pre-training model which shows better performance, fine-tuning is principal to make the results more realistic. In the other pre-training model, fine-tuning is mainly used to improve model performance. To further analyze the results that how the fine-tuning method makes the result more realistic, we choose one noisy trajectory as an example. The results show that the pre-trained model can match the first two points correctly after fine-tuning. In addition, we also analyze the attention mechanisms to find the internal correlation of GPS points and the external relationship between input trajectory and

output results. The results show that the points which are on the current and adjacent road segment have a strong correlation with the target point. In addition, similar to the previous findings, while identifying a certain segment, not only the points on the target segment but also the points on the neighboring segments have correlations.

There are several directions in which the current study can be extended to further improve the map-matching performance. First one is to improve the quality of generated data for model pre-training. Currently, we use a simple rule to generate the trajectories for pre-training. There are, however, other variables that can help improve matching performance in addition to the proposed method. For instance, traffic volumes for each road segment, traffic signal, and road geometry information can all provide additional information to further improve the quality of generated data. Also, the complexity of the network, GPS positioning errors will highly affect the quality of the generated dataset for model training. In this case, there are two general solutions to solve this problem. The first one is to increase the number of labeled trajectories. If the amount of labeled trajectories is enough to train the model, it is preferable to fine-tune all the model's layers even if the dataset is different from the generated dataset. However, this kind of approach is laborious and costs a lot. The other solution is to fine-tune a suitable amount of layers in the model. Although it is cost-effective, the model accuracy can be relatively lower than the previous solution and it is hard to find an appropriate number of layers to do fine-tuning. Therefore, in future work, we will further analyze the requirements of labeled data with more complex scenarios and test our model in different scenarios. In addition, it is difficult to identify the relationship between the model performance and the number of real labeled data used in fine-tuning due to the lack of real labeled trajectories. Thus, we will increase the number of labeled data to find the stated relationship. Furthermore, we will test the model performance under different types of data such as buses, trucks, and normal vehicles. In this research, we manually label preprocessed Taxi trajectories onto their corresponding sequence of segments. Therefore, there are no restrictions on test data to develop the map-matching model. To be more specific, the deep learning-based map-matching models are efficient to extract and use the knowledge from the historical data to solve the problems. As a result, if we have other types of data like vehicle or bus labeled trajectories, we can also develop the corresponding model and test their performance.

CRediT authorship contribution statement

Zhixiong Jin: Conceptualization of this study, Methodology, Software, Validation, Formal analysis, Writing - original draft. **Jiwon Kim:** Conceptualization of this study, Writing - review and editing. **Hwasoo Yeo:** Data curation, Methodology, Resources, Supervision, Funding acquisition, Project administration, Writing - review and editing. **Seongjin Choi:** Conceptualization of this study, Methodology, Formal analysis, Supervision, Writing - original draft, Writing - review and editing.

References

Ahlgren, P., Jarneving, B., Rousseau, R., 2003. Requirements for a cocitation similarity measure, with special reference to pearson's correlation coefficient. Journal of the American Society for Information Science and Technology 54, 550–560.

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G., et al., 2016. Deep speech 2: End-to-end speech recognition in english and mandarin, in: International conference on machine learning, PMLR. pp. 173–182.

Ba, J.L., Kiros, J.R., Hinton, G.E., 2016. Layer normalization. arXiv preprint arXiv:1607.06450 .

Bahdanau, D., Cho, K., Bengio, Y., 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 .

Bernstein, D., Kornhauser, A., et al., 1996. An introduction to map matching for personal navigation assistants .

Bierlaire, M., Chen, J., Newman, J., 2013. A probabilistic map matching method for smartphone gps data. Transportation Research Part C: Emerging Technologies 26, 78–98.

Bird, J.J., Faria, D.R., Premebida, C., Ekárt, A., Ayrosa, P.P., 2020. Overcoming data scarcity in speaker identification: Dataset augmentation with synthetic mfccs via character-level rnn, in: 2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), IEEE. pp. 146–151.

Bookstein, A., Kulyukin, V.A., Raita, T., 2002. Generalized hamming distance. Information Retrieval 5, 353–375.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S., 2020. End-to-end object detection with transformers, in: European Conference on Computer Vision, Springer. pp. 213–229.

Chao, P., Xu, Y., Hua, W., Zhou, X., 2020. A survey on map-matching algorithms, in: Australasian Database Conference, Springer. pp. 121–133.

Chen, C.F., Fan, Q., Panda, R., 2021. Crossvit: Cross-attention multi-scale vision transformer for image classification. arXiv preprint arXiv:2103.14899 .

Chen, Z., Shen, H.T., Zhou, X., 2011. Discovering popular routes from trajectories, in: 2011 IEEE 27th International Conference on Data Engineering, IEEE. pp. 900–911.

Choi, S., Kim, J., Yeo, H., 2019. Attention-based recurrent neural network for urban vehicle trajectory prediction. Procedia Computer Science 151, 327–334.

Choi, S., Kim, J., Yeo, H., 2021. Trajgail: Generating urban vehicle trajectories using generative adversarial imitation learning. Transportation Research Part C: Emerging Technologies 128, 103091.

Choi, S., Yeo, H., Kim, J., 2018. Network-wide vehicle trajectory prediction in urban traffic networks using deep learning. Transportation Research Record 2672, 173–184.

Clauwaert, J., Menschaert, G., Waegeman, W., 2020. Explainable transformer models for functional genomics in prokaryotes. bioRxiv .

Di Gangi, M.A., Negri, M., Cattoni, R., Dessi, R., Turchi, M., 2019. Enhancing transformer for end-to-end speech-to-text translation, in: Proceedings of Machine Translation Summit XVII: Research Track, pp. 21–31.

Dong, L., Xu, S., Xu, B., 2018. Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition, in: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE. pp. 5884–5888.

Dzabraev, M., Kalashnikov, M., Komkov, S., Petiushko, A., 2021. Mdmmt: Multidomain multimodal transformer for video retrieval, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 3354–3363.

Feng, J., Li, Y., Zhao, K., Xu, Z., Xia, T., Zhang, J., Jin, D., 2020. Deepmm: Deep learning based map matching with data augmentation. IEEE Transactions on Mobile Computing .

Gabeur, V., Sun, C., Alahari, K., Schmid, C., 2020. Multi-modal transformer for video retrieval, in: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IV 16, Springer. pp. 214–229.

Gehring, J., Auli, M., Grangier, D., Yarats, D., Dauphin, Y.N., 2017. Convolutional sequence to sequence learning, in: International Conference on Machine Learning, PMLR. pp. 1243–1252.

Genders, W., Razavi, S., 2016. Using a deep reinforcement learning agent for traffic signal control. arXiv preprint arXiv:1611.01142 .

Gong, Y.J., Chen, E., Zhang, X., Ni, L.M., Zhang, J., 2017. Antmapper: An ant colony-based map matching approach for trajectory-based applications. IEEE Transactions on Intelligent Transportation Systems 19, 390–401.

Greenfeld, J.S., 2002. Matching gps observations to locations on a digital map, in: Transportation Research Board 81st Annual Meeting.

Grundkiewicz, R., Junczys-Dowmunt, M., Heafield, K., 2019. Neural grammatical error correction systems with unsupervised pre-training on synthetic data, in: Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications, pp. 252–263.

Gulati, A., Qin, J., Chiu, C.C., Parmar, N., Zhang, Y., Yu, J., Han, W., Wang, S., Zhang, Z., Wu, Y., et al., 2020. Conformer: Convolution-augmented transformer for speech recognition. arXiv preprint arXiv:2005.08100 .

Hamming, R.W., 1986. Coding and information theory. Prentice-Hall, Inc.

Hao, S., Lee, D.H., Zhao, D., 2019. Sequence to sequence learning with attention mechanism for short-term passenger flow prediction in large-scale metro system. Transportation Research Part C: Emerging Technologies 107, 287–300.

Hashemi, M., Karimi, H.A., 2014. A critical review of real-time map-matching algorithms: Current issues and future directions. Computers, Environment and Urban Systems 48, 153–165.

Hashemi, M., Karimi, H.A., 2016. A weight-based map-matching algorithm for vehicle navigation in complex urban networks. Journal of Intelligent Transportation Systems 20, 573–590.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778.

Honey, S.K., Zavoli, W.B., Milnes, K.A., Phillips, A.C., White Jr, M.S., Loughmiller Jr, G.E., 1989. Vehicle navigational system and method. US Patent 4,796,191.

Huang, X., Zhao, Y., Ma, C., Yang, J., Ye, X., Zhang, C., 2015. Trajgraph: A graph-based visual analytics approach to studying urban network centralities using taxi trajectory data. IEEE transactions on visualization and computer graphics 22, 160–169.

Jo, T., Haseyama, M., Kitajima, H., 1996. A map matching method with the innovation of the kalman filtering. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences 79, 1853–1855.

Keskar, N.S., McCann, B., Varshney, L.R., Xiong, C., Socher, R., 2019. Ctrl: A conditional transformer language model for controllable generation. arXiv preprint arXiv:1909.05858 .

Kim, J., Mahmassani, H.S., 2015. Spatial and temporal characterization of travel patterns in a traffic network using vehicle trajectories. Transportation Research Procedia 9, 164–184.

Kim, S., Kim, J., Hyun, I., 1998. Development of a map matching algorithm for car navigation system using fuzzy q-factor algorithm, in: TOWARDS THE NEW HORIZON TOGETHER. PROCEEDINGS OF THE 5TH WORLD CONGRESS ON INTELLIGENT TRANSPORT SYSTEMS, HELD 12-16 OCTOBER 1998, SEOUL, KOREA. PAPER NO. 4020.

Kim, W., Jee, G.I., Lee, J., 2000. Efficient use of digital road map in various positioning for its, in: IEEE 2000. Position Location and Navigation Symposium (Cat. No. 00CH37062), IEEE. pp. 170–176.

Kortylewski, A., Schneider, A., Gerig, T., Egger, B., Morel-Forster, A., Vetter, T., 2018. Training deep face recognition systems with synthetic data. arXiv preprint arXiv:1802.05891 .

Kos, S., Brcic, D., Musulin, I., 2013. Smartphone application gps performance during various space weather conditions: a preliminary study, in: Proceedings of the 21st International Symposium on Electronics in Transport (ISEP 2013), pp. 1–4.

Kubicka, M., Cela, A., Mounier, H., Niculescu, S.I., 2018. Comparative study and application-oriented classification of vehicular map-matching methods. IEEE Intelligent Transportation Systems Magazine 10, 150–166.

Li, H., Kulik, L., Ramamohanarao, K., 2015. Robust inferences of travel paths from gps trajectories. International Journal of Geographical Information Science 29, 2194–2222.

Li, L., Zhang, Y., Chen, L., 2021. Personalized transformer for explainable recommendation. arXiv preprint arXiv:2105.11601 .

Li, N., Liu, S., Liu, Y., Zhao, S., Liu, M., 2019. Neural speech synthesis with transformer network, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 6706–6713.

Likic, V., 2008. The needleman-wunsch algorithm for sequence alignment. Lecture given at the 7th Melbourne Bioinformatics Course, Bi021 Molecular Science and Biotechnology Institute, University of Melbourne , 1–46.

Lin, T., Wang, Y., Liu, X., Qiu, X., 2021. A survey of transformers. arXiv preprint arXiv:2106.04554 .

Lipton, Z.C., Berkowitz, J., Elkan, C., 2015. A critical review of recurrent neural networks for sequence learning. arXiv preprint arXiv:1506.00019 .

Liu, L., Hamilton, W., Long, G., Jiang, J., Larochelle, H., 2020. A universal representation transformer layer for few-shot image classification. arXiv preprint arXiv:2006.11702 .

Liu, Z., Fang, J., Tong, Y., Xu, M., 2021. Deep learning enabled vehicle trajectory map-matching method with advanced spatial–temporal analysis. IET Intelligent Transport Systems 14, 2052–2063.

Lou, Y., Zhang, C., Zheng, Y., Xie, X., Wang, W., Huang, Y., 2009. Map-matching for low-sampling-rate gps trajectories, in: Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems, pp. 352–361.

Lu, K., Grover, A., Abbeel, P., Mordatch, I., 2021. Pretrained transformers as universal computation engines. arXiv preprint arXiv:2103.05247 .

Luo, A., Chen, S., Xv, B., 2017. Enhanced map-matching algorithm with a hidden markov model for mobile phone positioning. ISPRS International Journal of Geo-Information 6, 327.

Lv, Y., Duan, Y., Kang, W., Li, Z., Wang, F.Y., 2014. Traffic flow prediction with big data: a deep learning approach. IEEE Transactions on Intelligent Transportation Systems 16, 865–873.

Merry, K., Bettinger, P., 2019. Smartphone gps accuracy study in an urban environment. PloS one 14, e0219890.

Mohamed, R., Aly, H., Youssef, M., 2016. Accurate real-time map matching for challenging environments. IEEE Transactions on Intelligent Transportation Systems 18, 847–857.

Namozov, A., Im Cho, Y., 2018. An efficient deep learning algorithm for fire and smoke detection with limited data. Advances in Electrical and Computer Engineering 18, 121–128.

Neubig, G., 2017. Neural machine translation and sequence-to-sequence models: A tutorial. arXiv preprint arXiv:1703.01619 .

Newson, P., Krumm, J., 2009. Hidden markov map matching through noise and sparseness, in: Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems, pp. 336–343.

Noh, B., Park, H., Yeo, H., 2022. Analyzing vehicle–pedestrian interactions: Combining data cube structure and predictive collision risk estimation model. Accident Analysis & Prevention 165, 106539.

Pan, S.J., Yang, Q., 2009. A survey on transfer learning. IEEE Transactions on knowledge and data engineering 22, 1345–1359.

Papineni, K., Roukos, S., Ward, T., Zhu, W.J., 2002. Bleu: a method for automatic evaluation of machine translation, in: Proceedings of the 40th annual meeting of the Association for Computational Linguistics, pp. 311–318.

Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A., Tran, D., 2018. Image transformer, in: International Conference on Machine Learning, PMLR. pp. 4055–4064.

Quddus, M.A., Noland, R.B., Ochieng, W.Y., 2006. A high accuracy fuzzy logic based map matching algorithm for road transport. Journal of Intelligent Transportation Systems 10, 103–115.

Quddus, M.A., Ochieng, W.Y., Noland, R.B., 2007. Current map-matching algorithms for transport applications: State-of-the art and future research directions. Transportation research part c: Emerging technologies 15, 312–328.

Quddus, M.A., Ochieng, W.Y., Zhao, L., Noland, R.B., 2003. A general map matching algorithm for transport telematics applications. GPS solutions 7, 157–167.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., Sutskever, I., 2021. Zero-shot text-to-image generation. arXiv preprint arXiv:2102.12092 .

Renso, C., Baglioni, M., de Macedo, J.A.F., Trasarti, R., Wachowicz, M., 2013. How you move reveals who you are: understanding human behavior by analyzing trajectory data. Knowledge and information systems 37, 331–362.

Sharath, M., Velaga, N.R., Quddus, M.A., 2019. A dynamic two-dimensional (d2d) weight-based map-matching algorithm. Transportation Research Part C: Emerging Technologies 98, 409–432.

Shutao, X., Fu, F., 1998. On the average hamming distance for binary codes. Discrete applied mathematics 89, 269–276.
Singh, J., Singh, S., Singh, S., Singh, H., 2019. Evaluating the performance of map matching algorithms for navigation systems: an empirical study. Spatial Information Research 27, 63–74.
Sokolova, M., Japkowicz, N., Szpakowicz, S., 2006. Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation, in: Australasian joint conference on artificial intelligence, Springer. pp. 1015–1021.
Sola, J., Sevilla, J., 1997. Importance of input data normalization for the application of neural networks to complex industrial problems. IEEE Transactions on nuclear science 44, 1464–1468.
Sun, J., Kim, J., 2021. Joint prediction of next location and travel time from urban vehicle trajectories using long short-term memory neural networks. Transportation Research Part C: Emerging Technologies 128, 103114.
Sutskever, I., Vinyals, O., Le, Q.V., 2014. Sequence to sequence learning with neural networks, in: Advances in neural information processing systems, pp. 3104–3112.
Taguchi, S., Koide, S., Yoshimura, T., 2018. Online map matching with route prediction. IEEE Transactions on Intelligent Transportation Systems 20, 338–347.
Taylor, L., Nitschke, G., 2018. Improving deep learning with generic data augmentation, in: 2018 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE. pp. 1542–1547.
Toledo-Moreo, R., Bétaille, D., Peyret, F., 2009. Lane-level integrity provision for navigation and map matching with gnss, dead reckoning, and enhanced maps. IEEE Transactions on Intelligent Transportation Systems 11, 100–112.
Travis, W., Bevly, D.M., 2008. Trajectory duplication using relative position information for automated ground vehicle convoys, in: 2008 IEEE/ION Position, Location and Navigation Symposium, IEEE. pp. 1022–1032.
Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., To, T., Cameracci, E., Boochoon, S., Birchfield, S., 2018. Training deep networks with synthetic data: Bridging the reality gap by domain randomization, in: Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp. 969–977.
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need, in: Advances in neural information processing systems, pp. 5998–6008.
Vig, J., 2019. A multiscale visualization of attention in the transformer model, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, Association for Computational Linguistics, Florence, Italy. pp. 37–42. URL: https://www.aclweb.org/anthology/P19-3007, doi:10.18653/v1/P19-3007.
Vig, J., Belinkov, Y., 2019. Analyzing the structure of attention in a transformer language model. arXiv preprint arXiv:1906.04284 .
Voulodimos, A., Doulamis, N., Doulamis, A., Protopapadakis, E., 2018. Deep learning for computer vision: A brief review. Computational intelligence and neuroscience 2018.
Wang, J., Gu, Q., Wu, J., Liu, G., Xiong, Z., 2016. Traffic speed prediction and congestion source exploration: A deep learning method, in: 2016 IEEE 16th international conference on data mining (ICDM), IEEE. pp. 499–508.
Wang, Q., Li, B., Xiao, T., Zhu, J., Li, C., Wong, D.F., Chao, L.S., 2019. Learning deep transformer models for machine translation. arXiv preprint arXiv:1906.01787 .
Wang, X., Liu, X., Lu, Z., Yang, H., 2021. Large scale gps trajectory generation using map based on two stage gan. Journal of Data Science 19, 126–141.
Wang, Y., Zhu, Y., He, Z., Yue, Y., Li, Q., 2011. Challenges and opportunities in exploiting large-scale gps probe data. HP Laboratories, Technical Report HPL-2011-109 21.
White, C.E., Bernstein, D., Kornhauser, A.L., 2000. Some map matching algorithms for personal navigation assistants. Transportation research part c: emerging technologies 8, 91–108.
Wu, Y., Tan, H., Qin, L., Ran, B., Jiang, Z., 2018. A hybrid deep learning based traffic flow prediction method and its understanding. Transportation Research Part C: Emerging Technologies 90, 166–180.
Xu, J., Sun, X., Zhang, Z., Zhao, G., Lin, J., 2019. Understanding and improving layer normalization. arXiv preprint arXiv:1911.07013 .
Yang, C., Gidofalvi, G., 2018. Fast map matching, an algorithm integrating hidden markov model with precomputation. International Journal of Geographical Information Science 32, 547–570.
Yang, S., Quan, Z., Nie, M., Yang, W., 2020. Transpose: Towards explainable human pose estimation by transformer. arXiv preprint arXiv:2012.14214 .
Yoon, J., Ahn, K., Park, J., Yeo, H., 2021. Transferable traffic signal control: Reinforcement learning with graph centric state representation. Transportation Research Part C: Emerging Technologies 130, 103321.
Yoon, J., Kim, S., Byon, Y.J., Yeo, H., 2020. Design of reinforcement learning for perimeter control using network transmission model based macroscopic traffic simulation. Plos one 15, e0236655.
Yosinski, J., Clune, J., Bengio, Y., Lipson, H., 2014. How transferable are features in deep neural networks? arXiv preprint arXiv:1411.1792 .
Young, T., Hazarika, D., Poria, S., Cambria, E., 2018. Recent trends in deep learning based natural language processing. ieee Computational intelligenCe magazine 13, 55–75.
Yuan, L., Li, D., Hu, S., 2018. A map-matching algorithm with low-frequency floating car data based on matching path. EURASIP Journal on Wireless Communications and Networking 2018, 1–14.
Zhang, J., Meng, W., Liu, Q., Jiang, H., Feng, Y., Wang, G., 2016. Efficient vehicles path planning algorithm based on taxi gps big data. Optik 127, 2579–2585.
Zhang, Z.Z., 2001. A relation between the average hamming distance and the average hamming weight of binary codes. Journal of statistical planning and inference 94, 413–419.
Zhao, K., Feng, J., Xu, Z., Xia, T., Chen, L., Sun, F., Guo, D., Jin, D., Li, Y., 2019. Deepmm: Deep learning based map matching with data augmentation, in: Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 452–455.

Zheng, K., Zheng, Y., Xie, X., Zhou, X., 2012. Reducing uncertainty of low-sampling-rate trajectories, in: 2012 IEEE 28th international conference on data engineering, IEEE. pp. 1144–1155.

Zheng, Y., Li, X., Xie, F., Lu, L., 2020. Improving end-to-end speech synthesis with local recurrent neural network enhanced transformer, in: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE. pp. 6734–6738.

Zhu, L., Holden, J.R., Gonder, J.D., 2017. Trajectory segmentation map-matching approach for large-scale, high-resolution gps data. Transportation Research Record 2645, 67–75.

Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J., 2020. Deformable detr: Deformable transformers for end-to-end object detection. arXiv preprint arXiv:2010.04159 .

Zhu, X.J., 2005. Semi-supervised learning literature survey .

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., He, Q., 2020. A comprehensive survey on transfer learning. Proceedings of the IEEE 109, 43–76.