information

Article

Improving Map Matching of Floating Car Data with Artificial Intelligence Techniques

Georgia Ayfantopoulou [1], Marios Nikolaos Militsis [1], Josep Maria Salanova Grau [1] and Socrates Basbas [2,*]

[1] Center for Research and Technology Hellas, Hellenic Institute of Transport, 57 001 Thessaloniki, Greece
[2] Laboratory of Transportation Planning, Transportation Engineering & Highway Engineering, Department of Transportation & Hydraulic Engineering, School of Rural & Surveying Engineering, Aristotle University of Thessaloniki, 541 24 Thessaloniki, Greece
* Correspondence: smpasmpa@auth.gr

Abstract: Map matching is a crucial data processing task for transferring measurements from the dynamic sensor location to the relevant road segment. It is especially important when estimating road network speed by using probe vehicles (floating car data) as speed measurement sensors. Most common approaches rely on finding the closet road segment, but road network geometry (e.g., dense areas, two-way streets, and superposition of road segments due to different heights) and inaccuracy in the GNSS location (up to decades of meters in urban areas) can wrongly allocate up to 30% of the measurements. More advanced methods rely on taking the topology of the network into account, significantly improving the accuracy at a higher computational cost, especially when the accuracy of the GNSS location is low. In order to both improve the accuracy of the "closet road segment" methods and reduce the processing time of the topology-based methods, the data can be pre-processed using AI techniques to reduce noise created by the inaccuracy of the GNSS location and improve the overall accuracy of the map-matching task. This paper applies AI to correct GNSS locations and improve the map-matching results, achieving a matching accuracy of 76%. The proposed methodology is demonstrated to the floating car data generated by a fleet of 1200 taxi vehicles in Thessaloniki used to estimate road network speed in real time for information services and for supporting traffic management in the city.

Keywords: map matching; GNSS trajectory; floating car data; artificial intelligence; deep neural networks

check for updates

Citation: Ayfantopoulou, G.; Militsis, M.N.; Grau, J.M.S.; Basbas, S. Improving Map Matching of Floating Car Data with Artificial Intelligence Techniques. *Information* **2022**, *13*, 508. https://doi.org/10.3390/info13110508

Academic Editors: Miguel Félix Mata Rivera and Roberto Zagal Flores

Received: 10 September 2022
Accepted: 21 October 2022
Published: 24 October 2022

1. Introduction

Route optimization [1], traffic scheduling [2], fleet management [3], travel time estimation [4], and other location-based services are significantly dependent on the accurate mapping of raw GNSS trajectories onto the segments of road networks. However, satellite visibility limitations, attenuated satellite signals, and GNSS device malfunctions may all result in up to 20 m of accuracy degradation [5]. Dual frequency capability overcomes the outlined noise sources; however, it is not widely available. Different GNSS device settings and unpredictable communication failures can result in unintended sampling times due to limited energy and transmission bandwidth. Currently available commercial digital maps are prone to deficiencies, resulting in additional matching errors ranging from 5 to 20 m [6]. Further noise can be induced by complicated road network geometry (e.g., roundabouts, bidirectional streets, and superposition of road segments due to different heights) [7], dense areas in the urban environment commonly referred to as urban canyons [8], and varying weather conditions [9]. Therefore, map matching is indispensable to accurately identify the road segments that a vehicle travelled by mitigating the prevalent discrepancies between the recorded raw GNSS trajectory and the ground truth one [10]. As a result, map matching facilitates the effectual and robust functionality of intelligent transportation systems by capturing the vehicle routes on the urban road network online [11,12].

A lot of effort has been put into exploring map-matching algorithms to support various trajectory-based applications over the last two decades [13–15]. A map-matching algorithm can be classified as online or offline based on the way it processes GNSS data. Online methods deal with real-time data streams, while offline methods process historical trajectory data [16].

Depending on the network and trip features taken into account, map-matching algorithms can be categorized into four categories [10]. Geometric map-matching algorithms exclusively consider the geometry of the road network. Topology based map-matching algorithms consider the topology of the network, which consists of features such as intersection turning restrictions, road segments direction, and the connectivity between consecutive segments. From a stochastic perspective, probabilistic map-matching algorithms assign probabilities to multiple road segments in a confidence region surrounding a GNSS position. Finally, algorithms that are considered to be more precise and robust to the noise sources in complex urban road networks are called advanced map-matching algorithms. The last class of algorithms consists of more refined conceptual models, such as Hidden Markov Models (HMMs) [17], which are often used for benchmarking.

Most of the existing algorithms dwell a great deal on the geometric and topological features of the road network while not taking advantage of the enormous vehicle trajectory data and the internal dependencies between consecutive positions. Cumulative travel patterns embedded in massive amounts of GNSS trajectory data urged by the proliferation of smart devices can only be captured by shifting from rule-based approaches to data-driven perspectives. Although methods, such as the weight-based method [5], Kalman filter [18], Hidden Markov Model (HMM) [1,7], and fuzzy control theory [19], are fast and intuitive, applying pre-defined rules for independently processing each sequence of positions potentially leads to poor map matching. As the methods discussed above do not utilize historical trajectories, they omit important information regarding certain users' mobility patterns. Moreover, they do not consider historical trajectories of other vehicles, making it impossible to reveal important features of traffic distribution and road network, which would significantly improve the matching performance. Rule-based algorithms, and especially the topology-based ones, fail to address the very frequent violations of road traffic rules made by individual drivers, either deliberately or inadvertently. It is therefore impossible to snap such recorded trajectories on real road segments. Rule-based algorithms perform significantly worse when their rules are not updated according to occasional or permanent network topology modifications.

In order to address the aforementioned challenges and drawbacks of rule-based algorithms on the map-matching task, this paper applies AI and especially deep neural networks. Effectively capturing internal dependencies between GNSS sequences, as well as external correlations between recorded trajectories and road network paths, can be achieved by leveraging mechanisms such as the self and multi head attention modules of a prominent deep learning architecture in the field of language modeling and machine translation, namely transformer [20]. Transformer, one of the most important recent breakthroughs of AI, is expected to capture geometric and topological rules of the road network as well as their violations and leverage information in big trajectory data with outstanding performance while achieving parallel computing due to the multi-head scheme of the attention module. The proposed methodology is demonstrated on the floating car data generated by a fleet of 1200 taxi vehicles in Thessaloniki used to estimate road network speed in real time for information services and for supporting traffic management in the city. It can be regarded as a hybrid methodology, both offline and online, in the sense that during the training phase it utilizes historical trajectory data, while during the inference, real-time stream data are accepted as the input.

The remainder of the paper is structured as follows. The state-of-the-art method is presented in Section 2. The methodology of the proposed model is outlined in Section 3. Section 3.1 defines the framework in which the model is demonstrated. Section 3.2 introduces the data used. A detailed description of the model architecture as well as its input and output are presented in Section 3.3. In Section 4, the evaluation results are presented based on the proposed metrics. The methodology and results are summarized in Section 5, whereas limitations, challenges, and future directions are outlined in Section 6.

2. State-of-the-Art Method

With the latest advancements in AI, a shift to data-driven methodologies has been witnessed to overcome the inherent limitations of rule-based algorithms. In that end deep learning architectures have been harnessed to enhance the performance of map matching. A state-of-the-art method, among others, is the transformer-based map matching in conjunction with data augmentation and the transfer learning approach proposed by Jin et al. [21]. Jen et al. model the map-matching task as a classification task where the model assigns probabilities to each one of the road segments mapping the recorded GNSS location to the segment with the highest probability. In addition, they adopt the widely used auto-regressive decoder, which consumes the previous output to determine the current one. Transformer is one of the most influential advances in AI over the last five years. It was introduced by Vaswani et al. [20] as a novel architecture for neural machine translation based on the attention mechanism. In this paper, we approach the map-matching task as a regression problem with parallel decoding. That is, the model outputs the denoised coordinates for each recorded GNSS location, achieving an optimal trade-off between the computational cost and global computation. Transformer allows for capturing the internal dependencies between GNSS sequences as well as the external correlations between recorded trajectories and road network paths, mainly due to its encoder–decoder structure and the use of multi-head self-attention mechanism. The self-attention mechanism proposed by Vaswani et al. [20] is described below.

Attention Mechanism

Attention facilitates each specific encoder and decoder layer to generate output sequences by focusing on different parts of the input sequence. The transformer model can attend to information from various representation subspaces simultaneously at different positions due to its multi-head self-attention mechanism [20]. For each specific head of the multi-head attention scheme, give a single embedding of GNSS points in the latent space, $z_i \in R^d$, and three learned vectors $k_i \in R^{d'}$, $q_i \in R^{d'}$, and $v_i \in R^{d'}$, where $d' = \frac{d}{H}$ and H represents the total number of heads utilized. The elements of these vectors, known as the key, query, and value vectors, are learned in parallel with the rest of the model parameters through the training procedure. To determine how much each position in the embedded sequence attends to all the other positions, the attention weights $\alpha_{i,\,j}$ are computed, where i is a query index and j is a key/value index. The attention weights are computed through the SoftMax of all the pairwise dot products between the query and key vectors through (1).

$$\alpha_{i,\,j} = \frac{e^{\frac{1}{\sqrt{d'}}\,q_i^T k_j}}{\sum_{j=1}^{n} e^{\frac{1}{\sqrt{d'}}\,q_i^T k_j}} \tag{1}$$

For a specific position on the sequence of embeddings, the final output of a single head in the multi-head self-attention mechanism is computed through the is the aggregation of value vectors weighted by the corresponding attention weights according to (2).

$$z_i = \sum_{j=1}^{n} \alpha_{i,\,j} v_j \tag{2}$$

The outputs of the H different heads are concatenated and projected linearly to a different subspace through a projection matrix with learned parameters, leading to the final output of the multi-head self-attention layer.

The extra attention module lying in each one of the N decoder layers, functions similarly to multi-headed self-attention layer except that it depends on the layer below it for the query vectors and on the output of encoder stack for the key and value vectors. As a result, each token in the decoder can attend to every token in the input sequence. This module is similar to the regular attention mechanisms harnessed by encoder–decoder structures.

3. Materials and Methods

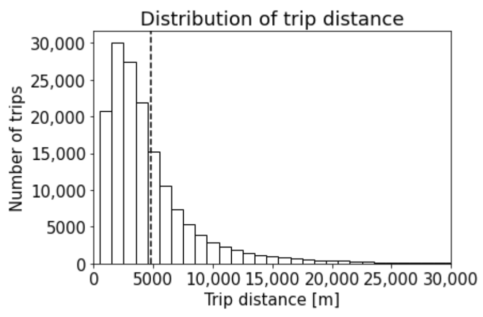
The dominant sequence transduction model, transformer, is utilized to address bottlenecks in rule-based algorithms. Transformer is one of the most influential advances in AI over the last five years. It was introduced by Vaswani et al. [20] as a new building block for neural machine translation based on the attention mechanism. The term "attention" is very well known in neuroscience, where individuals with processing bottlenecks focus selectively on some components of information and ignore others. Mapping the same concept to sequential data, the attention mechanism refers to the process of focusing on certain parts of sequences or regions during learning and blurring the remaining ones [22]. The main advantage of models utilizing attention is their ability to effectively capture long-term correlations. Transformer follows the encoder–decoder structure of the most competitive neural sequence transduction models [23–25]. In the proposed map-matching model, transformer is expected to capture the internal dependencies between GNSS sequences as well as the external correlations between recorded trajectories and road network paths. Attention modules in both encoders and decoders are primarily responsible for capturing correlations [26].

3.1. Case Study

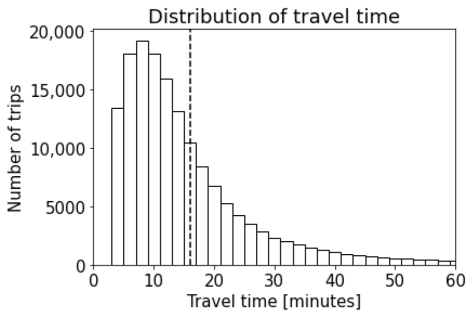
The Hellenic Institute of Transport (CERTH-H.I.T.) [27], in cooperation with the largest taxi association in Thessaloniki, Greece, "Taxiway" [28], has access in real time to the floating car data (FCD) produced by the taxi fleet. The taxi fleet consists of more than one thousand vehicles moving almost constantly in the region of Thessaloniki. Every moving vehicle produces and transmits 1 GNSS record per 10–12 s or every 100 m of movement. Each received record contains information about the moving speed, the vehicle's exact location, the altitude, the orientation, and additional information, such as whether the taxi transfers a customer or if it currently waits for a customer in a taxi stand. In average in H.I.T.'s Portal database servers nearly two thousand new FCD records per minute are being stored. The H.I.T. Portal provides accurate real-time traffic information in Thessaloniki, Greece, by estimating the average moving speed of the vehicles on the road network [29,30]. Currently, the speed estimations are produced every 15 min. The procedure to produce such value from the raw data can be described briefly as follows. Initially, all the FCD records are processed appropriately to remove any data that would allow for unauthorized or voluntary user identification. Records are then filtered to remove any erroneous entries with extraordinary speeds or unaligned coordinates, or entries generated by faulty GNSS receivers. By harnessing rule-based map-matching algorithms taking into account the network topology, each single record is mapped to the segment to which it is most likely to belong with the highest degree of certainty. Then, for each segment, proper statistical analysis is performed to provide a safe estimation of the average speed at which vehicle traffic is conducted on it. The FCD records are combined with detections from various Bluetooth sensors distributed across the city, generating accurate predictions regarding travel times on paths of the network and road traffic estimation [31].

3.2. Data

The dataset consists of 158,315 raw trajectories collected between 1 November and 16 November 2021. Since it includes both business days and weekends and the trips are sufficiently distributed over the course of a day, the dataset is expected to provide insight on the complex traffic patterns governing the road network. The records were manually annotated and snapped to the ground truth road segments. Regarding the noise on the GNSS signals, the Pearson's correlation coefficient [32] indicates that there is no correlation between the noise along latitude and noise along longitude, as it is equal to 0.0643 with a *p*-value very close to zero. The means and standard deviations of the latitude and longitude noise are 0.483–14.745 m and 0.285–15.256 m, respectively. Based on these findings, it can be stated that the noise for both coordinates follows the Gaussian distribution. The average trip length is approximately 5 km with a standard deviation of around 4 km, while the average travel time is 16 min with standard deviation of 14.54 min. The distributions of both measures are highlighted in Figure 1. Overall, 60% of the total recorded trips were traveled with one or more passengers, while the 40% were traveled with no passenger in the vehicle. As indicated in Figure 2, the recorded GNSS points (represented by the semitransparent grey area) are uniformly distributed over the road network of the metropolitan area of Thessaloniki. Additionally, 80% of the dataset is utilized for the training, 10 % is used for validation, and 10% for testing.



(**a**)



(**b**)

Figure 1. The distribution of trip distance (**a**) and travel time (**b**). The average value is represented by the vertical dashed line.

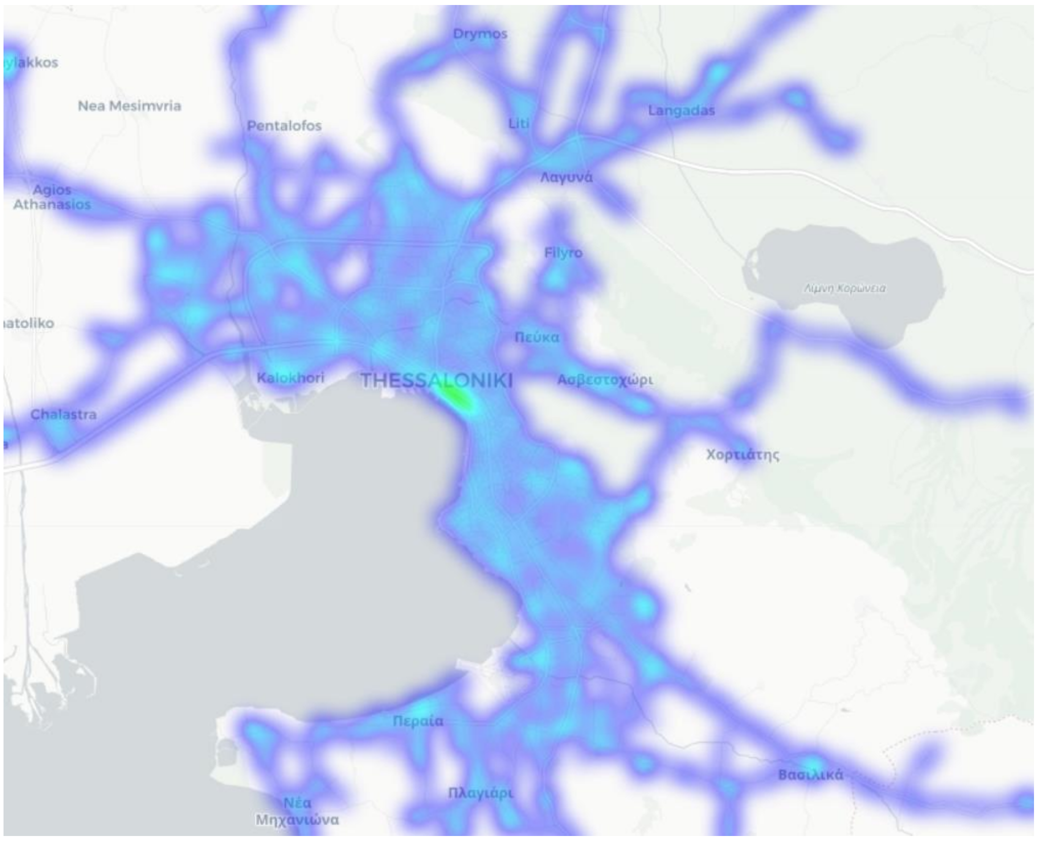


Figure 2. Spatial distribution of recorded GNSS points on metropolitan area of Thessaloniki.

3.3. Model Architecture

The overall matching procedure can be described briefly as follows. The model accepts as input the raw GNSS trajectory, which is a sequence of chronologically ordered GNSS points in vector format, denoted as $x = \{x_1, x_2, \ldots, x_n\}$, where n represents the length of the sequence. Each vector x_i contains information regarding the latitude and longitude, the sampling timestamp, and the speed, as well as the movement angle with respect to the north. The encoder maps the raw trajectory into a sequence of continuous representations $z = \{z_1, z_2, \ldots, z_n\}$ in the latent space. Each vector $z_i \in R^d$ is the embedding of each GNSS point in the latent space. Latent space sequences are of the same length as physical space raw sequences. Originally, transformer was used in auto-regressive fashion [25,33], generating one element of the output sequence at a time by consuming the previously generated output. This scheme is also followed by Jin et al. [21]. Auto-regression is, however, prohibitively expensive (proportional to the output length, and difficult to batch) to infer. As a result, parallel sequence generation has been harnessed in many fields such as audio [34], machine translation [35], word representation learning [36], speech recognition [37], and object detection [38]. Parallel decoding is also utilized through this paper, due to its optimal trade-off between computational cost and global computation. To this end, the decoder accepts as input a set of n learned positional embeddings of dimension d, where n is the length of the input sequence to the encoder. The final feed forward neural network on top of the decoder stack outputs a sequence of denoised GNSS points $y = \{y_1, y_2, \ldots, y_n\}$, which are further processed through a refinement and calibration procedure to generate trajectories in compliance with network geometry and topology. The architecture of the transformer is depicted on Figure 3. The demonstrated algorithm is implemented in Python [39], and for the transformer, Pytorch [40] is leveraged.

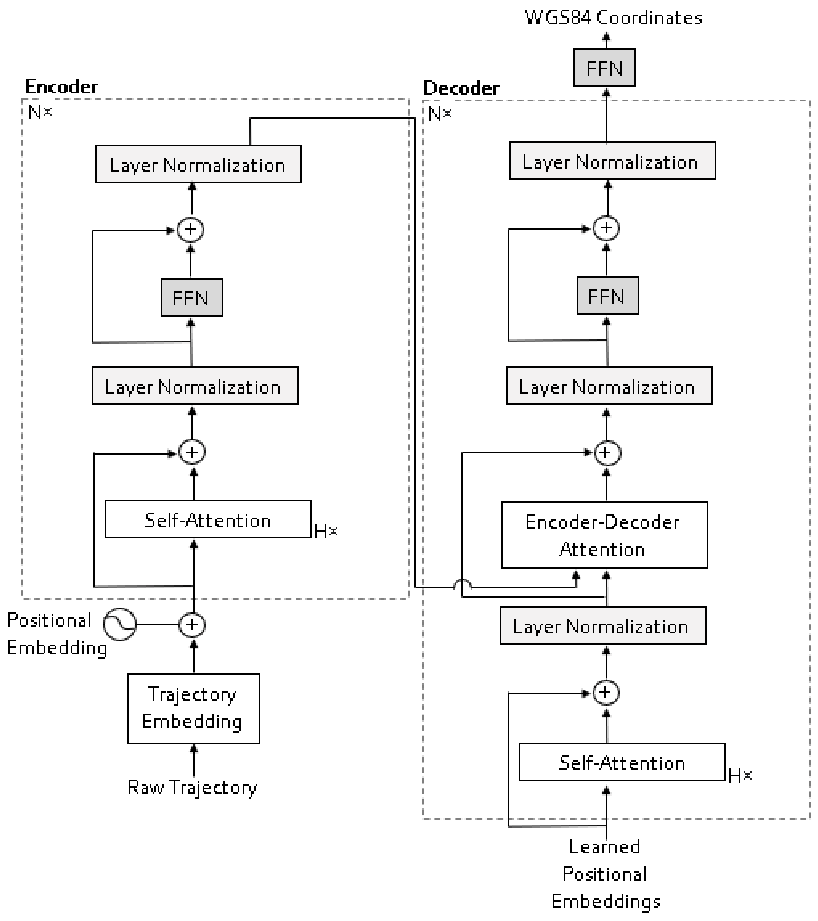


Figure 3. Architecture of transformer. Denoising of recorded GNSS coordinates.

3.3.1. Input Embedding

A linear projection is utilized to map the discrete GNSS location of each point (latitude, longitude) into a dense vector to latent space of dimension d capturing the semantics of the raw trajectory. Such vectors are called embeddings. GNSS points that are topologically similar to the physical space are projected geometrically close together in the embedding space. Since the transformer does not utilize recurrence or convolutional operations, the order of the sequence must be preserved by injecting information to model regarding the relative and absolute position of each embedding in the sequence. This is of high significance for the model to capture the temporal dependencies of the consecutive positions composing the GNSS trajectory. To this end additional vectors of dimension d are added to the input embeddings at the bottom layer of the encoder. There is a variety of positional embeddings that are either fixed or learned [41]. The sinusoid formula used by Vaswani et al. [20] is also utilized for this paper. The positional encodings are defined by sine and cosine functions of different frequencies, as shown in (3).

$$\begin{aligned} P_{(pos,2i)} &= \sin\left(\frac{pos}{10^{4^{2i}/d}}\right) \\ P_{(pos,2i+1)} &= \cos\left(\frac{pos}{10^{4^{2i}/d}}\right) \end{aligned} \tag{3}$$

3.3.2. Encoder–Decoder

The encoder consists of a stack of N identical layers. Each of these layers has a standard architecture and is composed of a multi-head self-attention module and a simple position-wise feed forward neural network with two linear transformations separated by a ReLU activation function. Around each of the two sublayers, there is a residual connection [42] followed by layer normalization [43]. The output of each sublayer is of dimension d.

The decoder also consists of a stack of N identical layers. Each of these layers follows the architecture of the corresponding layers in the encoded, supplemented by a third sublayer, which attends to the output generated by the last encoder layer. The bottom layer accepts as input n the learned positional embeddings of dimension d. As a result, the decoder, in parallel, decodes the n recorded GNSS points into their ground truth coordinates utilizing a regression head. The model can reason about all GNSS points together exploiting their dependencies, while considering the whole trajectory as context utilizing self-attention and encoder–decoder attention over the learnt positional embeddings.

3.3.3. Regression Head

The decoder stack outputs a vector of floats of dimension d for each GNSS point of the recorded trajectory. On top of the encoder, there is a regression head, which is a position-wise feed forward neural network composed of three hidden layers with ReLU activation function and a linear projection layer. This head predicts the ground truth latitude and longitude of each GNSS point as well as the angle of the vehicle's heading with respect to the north. By calculating the Euclidian distance and the deviation of the vehicle's heading and the segment orientation with respect to the north, the inferred coordinates are finally assigned to the closest road segment. In order to ensure geometry and topology compliance and continuity, further refinement and calibration is needed. Based on this post processing step, the final inferred trajectory consists of the inferred GNSS points, the nodes linking consecutive inferred segments, and all the intermediate points of each segment. This unscrambles complex geometrical patterns of road network such as roundabouts, bidirectional streets, superposition of road segments due to different heights, etc.

3.4. Technical Details

The model demonstrated in this paper is a transformer consisting of an encoder and a decoder stack, each one consisting of N = 6 identical layers. Both the multi-head self-attention and the encoder–decoder attention is composed by H = 8 heads. The dimension of each embedded vector, accepted as input to encoder and decoder layers, is of dimension

d = 256. The dimension of key, query, and value vectors is $d^{'}$ = 32. The hidden layers of all the feed forward neural networks consist of 256 neurons, expect the last layer of the regression head, which has three units. The models accept batched sequences with a batch size of 256.

The transformer is trained by minimizing the mean squared error between the model's predicted coordinates and the ground truth coordinates of the dataset through backpropagation. The Adam optimizer [44] is utilized with a learning rate of 10^{-5} and weight decay to 10^{-4}. The network parameters are initialized with Xavier init. [45]. Batch implementation is harnessed for taking advantage of the parallel computation of the GPU.

4. Results

The performance of the map-matching procedure is evaluated through the matching accuracy metric proposed in previous studies [46–49]. The matching accuracy is computed through (4):

$$Acc = \frac{len(\hat{T} \cap T)}{\max(len(\hat{T}),\ len(T))} \tag{4}$$

where $\hat{T}$ is the inferred trajectory, T is the ground truth trajectory, *len()* refers to the total length of the corresponding trajectory, and $\hat{T} \cap T$ represents the correctly matched segments. The operator max() acts as regularizer since it penalizes long inferred trajectories dealing with the fact that longer inferred trajectories are much more probable to contain the ground truth trajectory.

Table 1 highlights the matching performance achieved by a simple rule-based algorithm based on closest distance and orientation, the transformer network as well as their merge.

Table 1. Comparison of the matching accuracy achieved by rule-based algorithm, transformer, and their combination.

Algorithm	Accuracy
Rule Based	43.00%
Transformer	71.00%
Transformer and Rule-Based	76.00%

Based on the obtained results, it is obvious that the rule-based algorithm achieves poor performance, managing to map correctly less than the half road segments of each recorded trajectory. This comes with no surprise since the algorithm takes into account only specific geometrical features such as the distance and the orientation. To that end, complex geometrical schemes like roundabouts and superposition of road segments as well as topological rules, such as segment continuity and permitted turns, are omitted from the matching procedure. On the other hand, the performance obtained by the transformer network is much higher, mapping correctly around the 70% of each recorded trajectory length. This significant increase on the matching performance is because of the high learning capacity of the used deep neural architecture. The depth, the width, the memory, and the parallel mapping into different subspaces through the multi-head attention scheme seem to efficiently capture long-term dependencies among specific trajectories and complex traffic patterns of the road network. The matching accuracy of the transformer network is further increased by harnessing the rule-based algorithm as post processing step for refinement and calibration. This combination adds five more units on the matching performance.

As depicted in Figure 4, the performance of the suggested algorithm increases with increasing trajectory density in a given area. The noise imposed to the recorded GNSS trajectories due to the complex geometry of specific road segments is obviously reflected on the transformer's matching performance, as shown in Table 2.

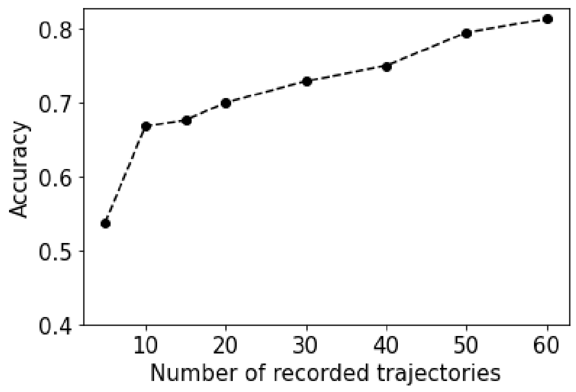


Figure 4. Local matching accuracy of the transformer network with regards to the density of trajectories of a given area.

Table 2. Comparison of the matching accuracy with regards to the geometry of road segment.

Algorithm	Accuracy
Intersections at Grade	68.00%
Grade-Separated Intersections	53.00%
Roundabouts	62.00%
Straight Links—1 Way	83.00%
Straight Links—2 Ways	79.00%

Table 3 highlights the fluctuation of the proposed algorithm's matching performance depending on the level of the spatial aggregation, which is related to the road network density.

Table 3. Comparison of the matching accuracy with regards to the level of spatial aggregation.

Area	Accuracy
City Centre	69.00%
Urban Area	73.00%
Metropolitan Area	74.00%
Regional Area	76.00%

Figure 5 depicts the effect of various hyperparameters on the matching performance achieved by the transformer network. The model reaches the highest value of its potential for embedding size d = 256, *batch size* = 256, number of identical layers on both the encoder and the decoder side N = 6, and number of heads on the multi-head attention module H = 8. It can be stated that the projection of the recorded GNSS points into a latent hyperspace of 256 dimensions, is a semantically meaningful representation capturing the correlation between different GNSS points. Points that are close to each other on the physical space in terms of semantics are geometrically close in the latent space. The batch size is the number of samples used for a single update on the model parameters during optimization, which is a training step. It is an important hyperparameter since it affects the stability of the optimization, that is, the convergence on a local optimum. The number of encoder and decoder layers is related to the capacity of the model to capture the complex patterns governing the map-matching process, and it is clear that the deeper the model is, the higher performance it can achieve. However, given a dataset, a network with too much capacity is prone to overfitting. That is, it achieves high performance on the samples used while training, but it cannot perform well on unknown samples; thus, it does not have generalization ability. On the other hand, given a dataset, a model of low capacity is prone to underfitting, which means it cannot achieve acceptable performance either in training on inference. The demonstrated model achieves around 70% matching accuracy with no lack of generalization for N = 6 layers. As a result of the multi-head self-attention mechanism, the model can simultaneously focus on information from different representation subspaces at different positions. The highest performance is achieved by utilizing $H = 8$ heads. That is, each GNSS point is projected on eight different subspaces on a single forward pass through the model.

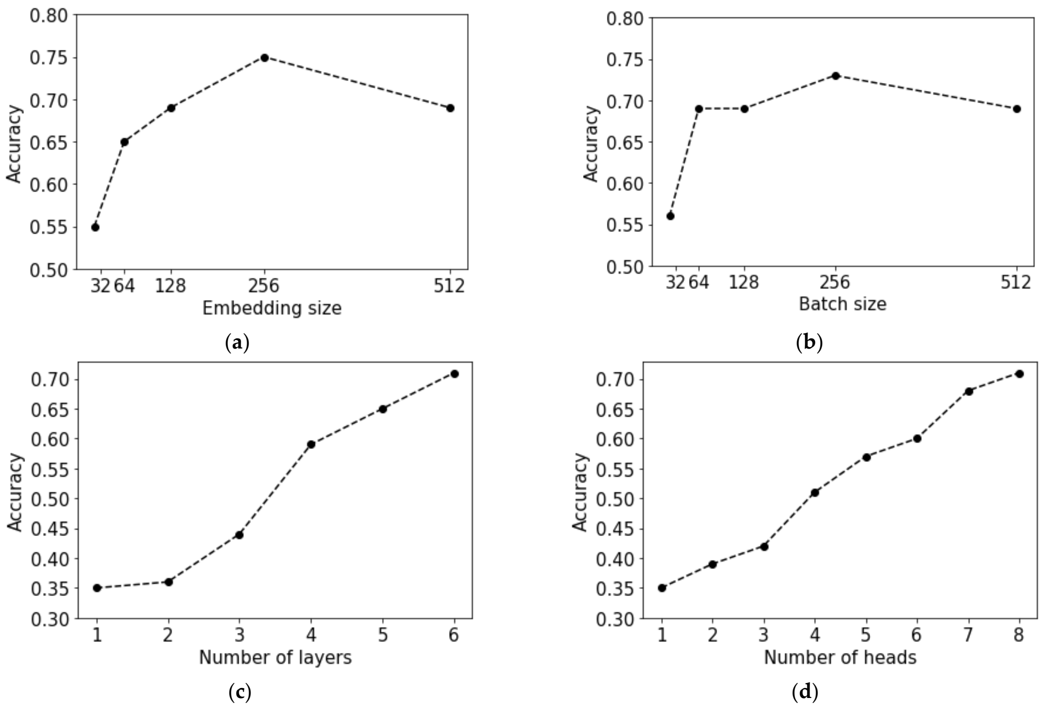


Figure 5. Matching accuracy of the transformer network with regards to the embedding size (**a**), the batch size (**b**), the number of layers both in the encoder and decoder (**c**), and the number of attention heads (**d**).

5. Discussion

In this paper, the map-matching procedure was investigated under the data-driven point of view utilizing AI techniques, and especially the high capacity of deep neural networks, in capturing complex patterns governing physical systems such as an urban road network. The case study of Thessaloniki indicates that rule-based algorithms that dwell a great deal on the geometric and topological features of the road network do not take advantage of the enormous vehicle trajectory data and the internal dependencies between consecutive positions. Collective travel patterns embedded in massive amounts of GNSS trajectory data urged by the proliferation of smart devices can only be captured by shifting from rule-based approaches to data-driven perspectives. Effectively capturing internal dependencies between GNSS sequences, as well as external correlations between recorded trajectories and road network paths, can be achieved by leveraging mechanisms, such as the multi-head self-attention modules of the prominent deep learning architecture in the field of language modeling and machine translation, namely transformer. The transformer network is an encoder–decoder architecture utilizing multi-head attention modules, for projecting each recorded GNSS point into many different latent subspaces, allowing us to jointly attend to information from different representation subspaces at different positions, expanding the model's ability to focus on different positions. The demonstrated algorithm is a transformer network with parallel decoding followed by a refinement and calibration post processing step. It is a hybrid algorithm, in terms of offline training and online inference. In that end, it provides accurate real-time estimations by revealing complex traffic and topological patterns based on enormous historical trajectory data. Based solely on the transformer network's capacity, the matching performance is as high as 71%, while the final refinement and calibration step adds 5%. In the case of Thessaloniki, the demonstrated algorithm is expected to be extended beyond the scope of motorized traffic for micromobility and decision support tools [50,51]. Feng et al. [46]

with a deep learning-based methodology, achieved a matching performance of 66% based on trajectories recorded in Beijing. Jiang et al. [47], also harnessing a deep learning model inspired by variational autoencoders, achieved a matching performance of 80% in the city of Porto. Jin et al. [21], using a transformer-based algorithm with autoregressive decoder, achieved a performance of around 90% based on taxi trajectories in Gangnam District in Seoul. Currently the direct comparison of the aforementioned methodologies with the proposed methodology of this paper is not feasible due to the fact that they are evaluated on different datasets with different evaluation metrics. However, all four methodologies could be evaluated under the same context in the future.

6. Conclusions

The obtained results are valid, for the floating car data collected by CERTH-H.I.T. [26] from probe sensors on taxi vehicles of Taxiway [27]. To that end, the matching accuracy of 76% may be affected by the features of the specific GNSS devices used for the collection of the data, which implies a specific noise distribution on the recorded locations. Additionally, the achieved performance refers to a sampling frequency of 10–12 s for the GNSS points. This sampling interval can be considered as high one, so the further investigation of the model's performance on data of lower sampling intervals is needed. The results may also be biased by the digital road network utilized through the matching procedure.

Based on the high matching performance achieved in Thessaloniki, the transformer architecture seems promising for unscrambling the very challenging map-matching task; however, further investigation is needed under different configurations. To consider the demonstrated model as a uniform machine dealing with the challenges of map matching, it needs to be fine-tuned and tested on different cities, with different GNSS devices and various sampling rates.

An important aspect of the demonstrated method is the data collection and annotation, which is a very laborious and time-consuming procedure. The transformer network achieves an accuracy of 71% trained approximately on 127,000 samples with no overfitting. Deep neural networks indeed are data hungry. As a result, the more data they consume during training, the more their generalization ability will be increased, and the higher their performance will be on both training and inference. The prohibitive time and labor needed for the manual annotation of more data could be avoided by utilizing data generated through simulations on digital environments such as digital twins [52]. Digital twins are cyber replicas of physical systems, created from traffic data collected from sensors, connected vehicles, traffic signals, and traffic monitoring cameras in real time, leveraging the embedded sensor systems of physical transportation systems to provide real-time and time-sensitive transportation services.

The matching performance of the demonstrated model, and of AI techniques in general, could be amplified by one or more complementary networks providing accurate real-time predictions regarding traffic measures of road segments such as the average speed and travel time as well as the traffic flow. Utilizing such estimations could drastically enhance the mapping of each single noisy GNSS location to the ground truth road segment. The accuracy of such complementary estimations as well as their compliance to the topology of the road network must be considered.

Harnessing the demonstrated algorithm for solving the challenging map-matching task requires a system with computational power and powerful GPUs to exploit parallelization. Computational cost and inference time can be significantly reduced by utilizing a lightweight version of the current "cumbersome" model through techniques such as knowledge distillation [53] and parameter pruning [54].

The demonstrated algorithm is currently effective for estimating road network speed in real time for information services and for supporting traffic management in the case of Thessaloniki. Addressing the limitations and challenges mentioned above would allow for the algorithm's transferability to various urban environments and its universality so that it can enhance the quality of location-based services of intelligent transportation systems, such as route optimization, traffic scheduling, fleet management, travel time estimation, and allocation. The effectiveness of the proposed map-matching algorithm could be enhanced by its evaluation on disaggregation level by leveraging the integrated procedures of traditional transport models and big data proposed by Croce et al. [55,56]. In order to further improve the matching accuracy, the inferred trajectories could be further calibrated through models regarding route and path choices [57].

Author Contributions: G.A.: conceptualization, data curation, project administration, resources, supervision, and validation, J.M.S.G.: data curation, project administration, supervision, validation, and writing—review and editing, M.N.M.: methodology, formal analysis, investigation, visualization, writing—review and editing, and writing—original draft, S.B.: data curation, project administration, supervision, validation, and writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Restrictions apply to the availability of these data. Data were obtained from Taxiway and are available at CERTH-H.I.T. with the permission of Taxiway.

Acknowledgments: The authors would like to thank Taxiway company for data sharing.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Yuan, J.; Zheng, Y.; Xie, X.; Sun, G. T-Drive: Enhancing Driving Directions with Taxi Drivers' Intelligence. *IEEE Trans. Knowl. Data Eng.* **2013**, *25*, 220–232. [CrossRef]
2. Stenneth, L.; Wolfson, O.; Yu, P.S.; Xu, B. Transportation mode detection using mobile phones and GIS information. In Proceedings of the 19th SIGSPATIAL International Conference on Advances in Geographic Information Systems, Chicago, IL, USA, 1–4 November 2011. [CrossRef]
3. Nair, R.; Miller-Hooks, E. Fleet Management for Vehicle Sharing Operations. *Transp. Sci.* **2011**, *45*, 524–540. [CrossRef]
4. Rahmani, M.; Koutsopoulos, H.N.; Jenelius, E. Travel time estimation from sparse floating car data with consistent path inference: A fixed point approach. *Transp. Res. Part C Emerg. Technol.* **2017**, *85*, 628–643. [CrossRef]
5. Sharath, M.; Velaga, N.R.; Quddus, M.A. A dynamic two-dimensional (D2D) weight-based map-matching algorithm. *Transp. Res. Part C Emerg. Technol.* **2019**, *98*, 409–432. [CrossRef]
6. Toledo-Moreo, R.; Betaille, D.; Peyret, F. Lane-Level Integrity Provision for Navigation and Map Matching with GNSS, Dead Reckoning, and Enhanced Maps. *IEEE Trans. Intell. Transp. Syst.* **2010**, *11*, 100–112. [CrossRef]
7. Merry, K.; Bettinger, P. Smartphone GPS accuracy study in an urban environment. *PLoS ONE* **2019**, *14*, e0219890. [CrossRef]
8. Mohamed, R.; Aly, H.; Youssef, M. Youssef, Accurate Real-time Map Matching for Challenging Environments. *IEEE Trans. Intell. Transp.* **2017**, *18*, 847–857. [CrossRef]
9. Kos, S.; Brčić, D.; Musulin, I. 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, Ljubljana, Slovenia, 25–26 March 2013. Available online: https://www.bib.irb.hr/623174 (accessed on 30 August 2022).
10. Quddus, M.A.; Ochieng, W.Y.; Noland, R.B. Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Transp. Res. Part C Emerg. Technol.* **2007**, *15*, 312–328. [CrossRef]
11. Lü, W.; Zhu, T.; Wu, D.; Dai, H.; Huang, J. A heuristic path-estimating algorithm for large-scale real-time traffic information calculating. *Sci. China Ser. E Technol. Sci.* **2008**, *51*, 165–174. [CrossRef]
12. Teng, W.; Wang, Y. Real-Time Map Matching: A New Algorithm Integrating Spatio-Temporal Proximity and Improved Weighted Circle. *Open Geosci.* **2019**, *11*, 288–297. [CrossRef]
13. Kubicka, M.; Cela, A.; Mounier, H.; Niculescu, S.-I. Comparative Study and Application-Oriented Classification of Vehicular Map-Matching Methods. *IEEE Intell. Transp. Syst. Mag.* **2018**, *10*, 150–166. [CrossRef]
14. Chao, P.; Xu, Y.; Hua, W.; Zhou, X. A Survey on Map-Matching Algorithms. In Proceedings of the ADC 2020: Databases Theory and Applications, Melbourne, Australia, 3–7 February 2020. [CrossRef]
15. Hashemi, M.; Karimi, H.A. A critical review of real-time map-matching algorithms: Current issues and future directions. *Comput. Environ. Urban Syst.* **2014**, *48*, 153–165. [CrossRef]
16. Gong, Y.-J.; Chen, E.; Zhang, X.; Ni, L.M.; Zhang, J. AntMapper: An Ant Colony-Based Map Matching Approach for Trajectory-Based Applications. *IEEE Trans. Intell. Transp. Syst.* **2018**, *19*, 390–401. [CrossRef]
17. Newson, P.; Krumm, J. Hidden Markov map matching through noise and sparseness. In Proceedings of the GIS'09: 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Seattle, WA, USA, 4–6 November 2009. [CrossRef]

18. Jo, T.; Haseyama, M.; Kitajima, H. A map matching method with the innovation of the kalman filtering. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **1996**, *79*, 1853–1855.
19. Kim, S.; Kim, J.; Hyun, I. Development of a map matching algorithm for car navigation system using fuzzy q-factor algorithm. In Proceedings of the 5th World Congress on Intelligent Transport Systems, Seoul, Korea, 12–16 October 1998. Available online: http://worldcat.org/isbn/899500732X (accessed on 30 August 2022).
20. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is all you need. In Proceedings of the A 31st International Conference on Neural Information Processing Systems—NIPS, Long Beach, CA, USA, 4–9 December 2017. Available online: https://dl.acm.org/doi/proceedings/10.5555/3295222 (accessed on 31 August 2022).
21. Jin, Z.; Kim, J.; Yeo, H.; Choi, S. Transformer-based map-matching model with limited labeled data using transfer-learning approach. *Transp. Res. Part C Emerg. Technol.* **2022**, *140*, 103668. [CrossRef]
22. Kamath, U.; Liu, J.; Whitaker, J. *Deep Learning for NLP and Speech Recognition*; Springer: Cham, Switzerland, 2019; pp. 407–419. [CrossRef]
23. Bahdanau, D.; Cho, K.; Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. In Proceedings of the ICLR 2015: International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015.
24. Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In Proceedings of the EMNLP 2014: Conference on Empirical Methods in Natural Language Processing, Doha, Qatar, 25–29 October 2014. [CrossRef]
25. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to Sequence Learning with Neural Networks. In Proceedings of the NIPS'14: 27th International Conference on Neural Information Processing Systems, Montreal, Canada, 8–13 December 2014. [CrossRef]
26. Lu, K.; Grover, A.; Abbeel, P.; Mordatch, I. Pretrained Transformers as Universal Computation Engines. *arXiv* **2021**, arXiv:2103.05247.
27. CERTH-HIT. Available online: https://www.imet.gr/index.php/en/ (accessed on 31 August 2022).
28. Taxiway. Available online: https://taxiway.gr/ (accessed on 31 August 2022).
29. Bratsas, C.; Koupidis, K.; Salanova, J.-M.; Giannakopoulos, K.; Kaloudis, A.; Ayfantopoulou, G. A Comparison of Machine Learning Methods for the Prediction of Traffic Speed in Urban Places. *Sustainability* **2019**, *12*, 142. [CrossRef]
30. Grau, J.M.S.; Maciejewski, M.; Bischoff, J.; Estrada, M.; Tzenos, P.; Stamos, I. Use of probe data generated by taxis. In *Big Data for Regional Science*; Routledge Advances in Regional Economics, Science and Policy; Taylor & Francis Group: Abingdon, UK; ISBN 1138282189/9781138282186.
31. Grau, J.M.S.; Mitsakis, E.; Tzenos, P.; Stamos, I.; Selmi, L.G. Ayfantopoulou, Multisource Data Framework for Road Traffic State Estimation. *J. Adv. Transp.* **2018**, *2018*, 9078547. [CrossRef]
32. Ahlgren, P.; Jarneving, B.; Rousseau, R. Requirements for a cocitation similarity measure, with special reference to Pearson's correlation coefficient. *J. Am. Soc. Inf. Sci. Technol.* **2003**, *54*, 550–560. [CrossRef]
33. Graves, A. Generating Sequences with Recurrent Neural Networks. *arXiv* **2013**, arXiv:1308.0850.
34. Oord, A.v.d.; Li, Y.; Babuschkin, I.; Simonyan, K.; Vinyals, O.; Kavukcuoglu, K.; Driessche, G.v.d.; Lockhart, E.; Cobo, L.C.; Stimberg, F.; et al. Parallel WaveNet: Fast High-Fidelity Speech Synthesis. In Proceedings of the Machine Learning Research—MLR, Sydney, Australia, 6–11 August 2017.
35. Ghazvininejad, M.; Levy, O.; Liu, Y.; Zettlemoyer, L. Mask-Predict: Parallel Decoding of Conditional Masked Language Models. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Hong Kong, China, 3–7 November 2019.
36. Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, MN, USA, 2–7 June 2019.
37. Chan, W.; Saharia, C.; Hinton, G.; Norouzi, M.; Jaitly, N. Imputer: Sequence Modelling via Imputation and Dynamic Programming. In Proceedings of the 37th International Conference on Machine Learning, Virtual Event, 13–18 July 2020.
38. Carion, N.; Kirillov, A.; Massa, F.; Synnaeve, G.; Usunier, N.; Zagoruyko, S. End-to-end object detection with Transformers. In Proceedings of the 16th European Conference on Computer Vision—ECCV 2020, Glasgow, UK, 23–28 August 2020.
39. Guido, V.R.; Drake, F.L. *Python 3 Reference Manual*; CreateSpace: Scotts Valley, CA, USA, 2009.
40. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Proceedings of the 33rd Conference on Neural Information Processing Systems—NeurIPS2019, Vancouver, Canada, 8–14 December 2019.
41. Gehring, J.; Auli, M.; Grangier, D.; Yarats, D.; Dauphin, Y.N. Convolutional Sequence to Sequence Learning. In ICML'17: Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017.
42. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition—CVPR, Las Vegas, NV, USA, 27–30 June 2016. [CrossRef]
43. Ba, J.L.; Kiros, J.R.; Hinton, G.E. Layer Normalization. *arXiv* **2016**, arXiv:1607.06450.
44. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference for Learning Representations, San Diego, CA, USA, 7–9 May 2015.
45. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the Artificial Intelligence and Statistics—AISTATS, Sardinia, Italy, 13–15 May 2010. Available online: http://proceedings.mlr.press/v9/glorot10a.html (accessed on 29 August 2022).

46. Feng, J.; Li, Y.; Zhao, K.; Xu, Z.; Xia, T.; Zhang, J.; Jin, D. DeepMM: Deep Learning Based Map Matching with Data Augmentation. *IEEE Trans. Mob. Comput.* **2020**, *21*, 2372–2384. [CrossRef]
47. Jiang, L.; Chen, C.-X. L2MM: Learning to Map Matching with Deep Models for Low-Quality GPS Trajectory Data. *ACM Trans. Knowl. Discov. Data* **2022**. [CrossRef]
48. Wu, H.; Mao, J.; Sun, W.; Zheng, B.; Zhang, H.; Chen, Z.; Wang, W. Probabilistic Robust Route Recovery with Spatio-Temporal Dynamics. In Proceedings of the KDD'16: 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016. [CrossRef]
49. Zheng, K.; Zheng, Y.; Xie, X.; Zhou, X. Reducing Uncertainty of Low-Sampling-Rate Trajectories. In Proceedings of the 2012 IEEE 28th International Conference on Data Engineering, Arlington, VA, USA, 1–5 April 2012. [CrossRef]
50. Ayfantopoulou, G.; Grau, J.M.S.; Maleas, Z.; Siomos, A. Micro-Mobility User Pattern Analysis and Station Location in Thessaloniki. *Sustainability* **2022**, *14*, 6715. [CrossRef]
51. Salanova, J.M.; Ayfantopoulou, G.; Magkos, E.; Mallidis, I.; Maleas, Z.; Narayanan, S.; Antoniou, C.; Tympakianaki, A.; Martin, I.; Fajardo-Calderin, J. Developing a Multilevel Decision Support Tool for Urban Mobility. *Sustainability* **2022**, *14*, 7764. [CrossRef]
52. Rudskoy, A.; Ilin, I.; Prokhorov, A. Digital Twins in the Intelligent Transport Systems. *Transp. Res. Procedia* **2021**, *54*, 927–935. [CrossRef]
53. Hinton, G.; Vinyals, O.; Dean, J. Distilling the Knowledge in a Neural Network. In Proceedings of the Advances of Neural Information Processing Systems 27—NIPS 2014, Montreal, Canada, 8–13 December 2014. [CrossRef]
54. Valdera, S.; Ameen, S. Methods for Pruning Deep Neural Networks. *arXiv* **2020**, arXiv:2011.00241.
55. Croce, A.I.; Musolino, G.; Rindone, C.; Vitetta, A. Vitetta, Transport System Models and Big Data: Zoning and Graph Building with Traditional Surveys, FCD and GIS. *ISPRS Int. J. Geo-Inf.* **2019**, *8*, 187. [CrossRef]
56. Croce, A.; Musolino, G.; Rindone, C.; Vitetta, A. Estimation of Travel Demand Models with Limited Information: Floating Car Data for Parameters' Calibration. *Sustainability* **2021**, *13*, 8838. [CrossRef]
57. Croce, A.I.; Musolino, G.; Rindone, C.; Vitetta, A. Route and Path Choices of Freight Vehicles: A Case Study with Floating Car Data. *Sustainability* **2020**, *12*, 8557. [CrossRef]