AI

MDPI

Article

GMTP: Enhanced Travel Time Prediction with Graph Attention Network and BERT Integration

Ting Liu and Yuan Liu *

School of Artificial Intelligence and Computer Science, Jiangnan University, Wuxi 102206, China; 6223152002@stu.jiangnan.edu.cn

* Correspondence: lyuan1800@jiangnan.edu.cn

Abstract: (1) Background: Existing Vehicle travel time prediction applications face challenges in modeling complex road network and handling irregular spatiotemporal traffic state propagation. (2) Methods: To address these issues, we propose a Graph Attention-based Multi-Spatiotemporal Features for Travel Time Prediction (GMTP) model, which integrates an enhanced graph attention network (GATv2) and Bidirectional Encoder Representations from Transformers (BERT) to analyze dynamic correlations across spatial and temporal dimensions. The pre-training process consists of two blocks: the Road Segment Interaction Pattern to Enhance GATv2, which generates road segment representation vectors, and a traffic congestion-aware trajectory encoder by incorporating a shared attention mechanism for high computational efficiency. Additionally, two self-supervised tasks are designed for improved model accuracy and robustness. (3) Results: The fine-tuned model had comparatively optimal performance metrics with significant reductions in Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and Root Mean Squared Error (RMSE). (4) Conclusions: Ultimately, the integration of this model into travel time prediction, based on two large-scale real-world trajectory datasets, demonstrates enhanced performance and computational efficiency.

Keywords: road network; GATv2; BERT; travel time preditction

Citation: Liu, T.; Liu, Y. GMTP: Enhanced Travel Time Prediction with Graph Attention Network and BERT Integration. *AI* **2024**, *5*, 2926–2944. https://doi.org/10.3390/ai5040141

Received: 10 November 2024
Revised: 3 December 2024
Accepted: 9 December 2024
Published: 13 December 2024

1. Introduction

With the continuous advancement of satellite positioning technology, the global positioning system (GPS) and Beidou high-precision positioning system are increasingly being applied in outdoor environments. These technologies not only enable real-time acquisition of users' precise location information but also provide a robust data foundation for various applications. The widespread adoption of positioning technology has generated amounts of spatiotemporal trajectory data, which typically include valuable information such as traffic flow, user behavior, and environmental factors [1,2]. As a result, effectively using these data for tasks such as trajectory-based prediction [3,4], traffic flow prediction [5,6], urban hazardous materials management [7], and trajectory similarity calculation [8] has become a hot topic in the field of data engineering.

Travel time prediction (TTP) is a critical component of many trajectory prediction tasks and an essential feature in numerous mobile map applications. For example, Baidu Maps, one of the world's largest mobile map platforms, serves more than 340 million active monthly users [9]. Accurate TTP is vital for practical applications, including navigation systems, urban planning, traffic management, ride-hailing services, logistics, and emergency response. Through TTP technology, the navigation system can provide the best route suggestions, avoid congestion, and provide drivers with visual information for safe and smooth driving based on road sign and traffic sign recognition [10]. Urban planners can leverage TTP data to assess traffic conditions across different city regions, informing decisions related to the spatial layout of commercial and residential areas [11]. Traffic managers can use TTP data to optimize signal timings and traffic flow, alleviating congestion

and improving overall traffic efficiency [12]. Ride-hailing platforms can optimize vehicle dispatch and order allocation based on predicted peak demand and hotspot locations, reducing passenger wait times and minimizing empty vehicle miles [13]. Logistics companies can predict package delivery times, optimize routes, manage inventory effectively, and monitor transportation processes in real time to improve service reliability [14]. Individual users can access real-time estimated arrival times through applications, plan trips more efficiently, and reduce waiting time. In case of disruptions, TTP data aid in quickly adjusting emergency measures and improving response times while offering users timely route suggestions to avoid accident-prone areas [15]. Thus, accurate and reliable TTP systems provide valuable data analysis and decision-making tools for urban planning and traffic management while advancing intelligent transportation systems and enhancing urban management capabilities.

Travel time prediction (TTP) is influenced by various factors, including traffic congestion, departure time, weather conditions, and individual driver preferences [10]. Traditional models often struggle to effectively integrate heterogeneous data and multidimensional spatiotemporal features, leading to limited prediction accuracy. However, the introduction of deep learning techniques has enabled TTP systems to capture complex nonlinear relationships and handle multiple variables. For example, real-time traffic data can be incorporated to optimize prediction models. Additionally, in regions with complex road network structures, it is crucial to consider the spatiotemporal correlations within the road network to improve the accuracy and reliability of travel time predictions.

In travel time prediction, mining spatiotemporal correlation features help models better capture the temporal irregularities and spatial dependencies in trajectory data, such as the contextual information between road segments and the temporal characteristics of traffic flow. Traditional road segment-based prediction methods [16,17] typically predict the travel time for each road segment independently and then sum these predictions to calculate the total travel time. Although this approach is computationally efficient, its accuracy decreases as the path length increases, and it often overlooks important contextual information like traffic lights and turns, limiting its overall precision. To address these shortcomings, end-to-end methods have been developed [18,19], which treat all road segments in a trajectory as a unified whole, capturing the contextual relationships between road segments using recurrent structures. However, the computational cost of these cyclic structures increases significantly as the number of road segments grows, restricting their real-time processing capabilities.

In response to these limitations, researchers have introduced deep learning models that combine various neural network architectures to handle complex data for travel time prediction. For example, the convolutional spatiotemporal graph attention network (STGAT) proposed in [20] transforms road networks into low-dimensional vectors and captures spatial information by setting an appropriate convolutional network size, thereby enhancing travel time prediction accuracy. Additionally, Wang [21] proposed a neural network-based model that integrates spatiotemporal features, traffic characteristics, and personalized factors. This model, comprising a shallow linear layer, a deep neural network, and a recurrent neural network, outperforms the WD (wide and deep) [22] model by more effectively mining local road segment information.

However, while these methods address the correlation between road segments to some extent, they still fail to fully explore the complex temporal patterns embedded in trajectories. Studies have shown that significant changes in trajectory data during morning and evening peak hours directly affect road congestion and trajectory formation. Irregular time intervals also reveal another temporal dimension of trajectories. In response, ref. [23] proposed a novel self-supervised trajectory representation learning framework, which incorporates time patterns and travel semantics into trajectory learning through a two-stage process, significantly improving downstream prediction tasks. Nevertheless, this model is complex, and the number of parameters increases substantially as the model grows

in complexity. Therefore, while improving model performance, reducing the number of parameters required for training remains an important direction for further optimization.

Many existing methods employ spatiotemporal graph neural networks to enhance travel time prediction accuracy, but they still face certain limitations. First, most approaches treat temporal and spatial features independently, failing to fully exploit the correlation between them. Second, these models often overlook the dynamic nature of traffic congestion, which leads to a significant decline in prediction accuracy under complex traffic conditions. To address these issues, this paper proposes GMTP, a spatiotemporal deep learning framework for travel time prediction based on an adaptive attention mechanism. The method integrates the optimized graph attention network GATv2 [24] with the powerful temporal modeling capabilities of BERT [25]. Compared to existing state-of-the-art models, GMTP simultaneously captures spatial and temporal dependencies while adaptively reducing training parameters, maintaining stable prediction performance, and demonstrating stronger transferability and task adaptability. As a result, our framework outperforms traditional methods, particularly in tasks involving complex relationships and multimodal data.

In integrating GATv2 with BERT, two key challenges must be addressed: handling heterogeneous data and fusing spatiotemporal features. On one hand, GATv2 learns the spatiotemporal characteristics of nodes through graph structure and outputs an embedding that represents node relationships. This embedding is used as the input (token embedding) for BERT, ensuring dimensional alignment between the two models. This enables BERT to capture spatial and temporal dependencies between nodes during task execution. On the other hand, additional location embeddings and time features (such as day and week) are incorporated based on the node embeddings generated by GATv2. This allows the model to dynamically process and fuse information from diverse data sources, enhancing the expressiveness of spatiotemporal data and improving prediction accuracy.

The specific implementation framework of GMTP is as follows: capturing the contextual relationship and spatiotemporal correlation characteristics in the road network information and extracting the temporal characteristics in the trajectory through the adaptive spatiotemporal attention mechanism to accurately capture the changes in traffic congestion. In addition, the framework introduces MLM-based trajectory reconstruction tasks and contrastive learning tasks to further enhance the pre-training model and fully learn the spatiotemporal characteristics of the trajectory data. Finally, the framework obtains more accurate travel time prediction results by fine-tuning downstream tasks. The following is the research innovation part of this article:

- Introduced a high-performance spatiotemporal graph attention network: Constructing a road segment interaction frequency matrix, based on GATv2 [20], to fully exploit spatial and temporal correlations for modeling complex road network structures. This approach deeply integrates with BERT [23], effectively capturing complex interaction relationships between nodes and enhancing the model's ability to analyze spatiotemporal dynamic features.
- Proposed a head information sharing spatiotemporal self-attention mechanism (HSSTA):This mechanism learns contextual information in trajectories by extracting traffic time characteristics such as peak hours and weekdays in the attention layer. A hybrid matrix is introduced in the attention head to adaptively adjust attention layer parameters, improving both computational efficiency and prediction accuracy.
- Designed an adaptive self-supervised learning task: This task reconstructs trajectory sequences by gradually increasing the masking ratio of trajectory subsequences. Combined with contrastive learning, this method reduces interference from other related information, increases the difficulty of true value prediction, and enhances the cross-sequence transferability of the pre-trained model, improving its generalization and robustness.

- Conducted experiments on two real-world trajectory datasets:The results demonstrate that the proposed method significantly outperforms other methods in both performance and computational efficiency.

2. Related Work

2.1. Spatiotemporal Graph Neural Networks

Graph neural networks (GNNs) [26] have been extensively applied in graph-based tasks such as traffic network analysis, knowledge graph creation, recommendation systems, and other domains that leverage graph structures. Their ability to handle non-Euclidean spatial data and complex features makes them highly effective in these areas. Advanced GNNs can generally be classified into four main types: recurrent graph neural networks (RecGNNs), convolutional graph neural networks (ConvGNNs), graph autoencoders (GAEs), and spatiotemporal graph neural networks (STGNNs).

To account for temporal dependencies, DLSTM [27] and GWN [28] introduced prediction frameworks built on recurrent neural networks (RNNs) and temporal convolutional networks (TCNs), respectively. GMAN [29] and STGNN [30] leverage temporal self-attention mechanisms to enhance long-term temporal learning. In comparison to traditional GNNs, STAN [31] incorporates an inter-region correlation model utilizing the graph attention network (GAT) [32]. However, while these studies have exploited spatiotemporal features, they have not jointly modeled both dimensions.

To address this issue, this paper proposes an enhanced dynamic graph attention network, GATv2, designed to dynamically infer spatial and temporal features within graph structures while embedding the temporal characteristics of segment interaction frequencies into the segment representations. Compared to the GAT, GATv2 overcomes its limitations and more effectively captures complex spatiotemporal correlation patterns.

2.2. Transformer-Based Language Models

Transformer-based language models have achieved substantial advancements across various fields. These models are generally classified into three categories: encoder–decoder models (T5 [33], BART [34]), encoder-only models (BERT [25], RoBERTa [35]), and decoder-only models (GPT-2 [36]). In recent years, decoder-based large language models (LLMs), particularly following the introduction of publicly available models like ChatGPT, have garnered significant attention. ChatGPT, through user feedback alignment, can effectively adapt to diverse tasks, accurately identify user intentions, and generate desired outcomes.

Following ChatGPT, several LLMs were released, including Llama2 [37] and Falcon [38]. While LLMs exhibit strong dialogue and interaction capabilities, their context-dependent learning strategies often underperform in specific tasks compared to fine-tuned, smaller-scale language models. Additionally, fine-tuning LLMs remains computationally intensive despite the introduction of more efficient parameter-tuning methods.

In light of these challenges, our study opts for the encoder-only BERT [25] model, aiming to reduce computational costs by improving the attention layer to decrease the number of parameters. Simultaneously, fine-tuning the pre-trained model enhances training accuracy.

2.3. Self-Supervised Learning

The masked language model (MLM) is a fundamental component of the Transformer architecture and is widely employed in tasks such as text generation, machine translation, and sentiment analysis [39]. Its mechanism involves randomly masking certain words in a sentence, allowing the model to predict the masked words based on the remaining unmasked content.

Contrastive learning [40], a self-supervised learning framework, has been extensively applied in tasks related to visual representation [41] and natural language processing [42]. Contrastive Predictive Coding (CPC) [43] is a pioneering approach in deep contrastive learning, optimizing the feature extraction network by maximizing the similarity (i.e.,

consistency) between predicted and actual results in sequence data. It also introduced the InfoNCE loss function, which is now widely used in contrastive learning research. Khosla et al. [44] proposed the supervised contrastive learning loss (SCL Loss), extending contrastive learning into supervised learning, to enhance the model's feature representation by utilizing labeled data.

Drawing inspiration from these studies, this research employs a data-augmented contrastive learning approach, leveraging anchor samples and the normalized cross-entropy loss function to identify trajectory samples that have undergone data augmentation.

3. Methodology

Figure 1 shows the architecture of the proposed framework GMTP, which consists of four modules: a segment interaction pattern-enhanced graph attention network, a traffic congestion-aware trajectory encoder, an adaptive masked trajectory reconstruction task, and a trajectory contrastive learning task.

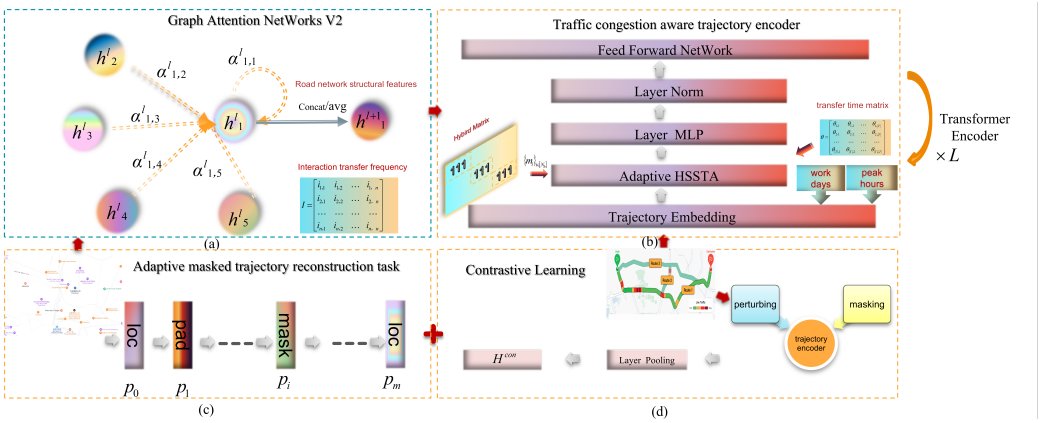

Figure 1. The architecture of GMTP. This figure illustrates the overall design: (**a**) The graph attention network V2 (GATv2) module captures spatial relationships by modeling the road network structure and incorporating Interaction Transfer Frequency for spatial information extraction. (**b**) The traffic congestion-aware trajectory encoder, equipped with adaptive shared attention, encodes road segment representations into trajectory vectors. Workday and peak hour information are integrated into the trajectory embeddings, and the attention mechanism is dynamically adjusted using transfer time and hybrid matrices for better spatiotemporal fusion. (**c**) The adaptive masked trajectory reconstruction task applies a random masking strategy to improve trajectory recovery, enhancing the accuracy of the trajectory representations. (**d**) The contrastive learning module strengthens feature extraction through Time-Frequency Perturbation and road segment masking, enhancing robustness and generalization.

3.1. Road Segment Interaction Pattern to Enhance GATv2

The segment interaction pattern-enhanced graph attention network employs a novel spatiotemporal graph attention mechanism to capture spatiotemporal correlations in complex traffic conditions, including factors such as segment interaction frequency.

A graph neural network (GNN) [26] layer updates the representation of each node by aggregating the representations of its neighboring nodes. Consider a directed graph $G = (V, E)$, where $V = \{1, \ldots, n\}$ is the set of nodes and E is the set of directed edges. An edge $(j, i) \in E$ indicates a directed edge from node j to node i.

In a GNN layer, the input consists of the node representations $\{h_i \in \mathbb{R}^d \mid i \in V\}$, along with the edge set E. The layer then produces a new set of node representations $\{h'_i \in \mathbb{R}^{d'} \mid i \in V\}$, where the same parametric function is applied to each node based on the representations of its neighbors. The set of neighbors for node i is defined as $N_i = \{j \in V \mid (j, i) \in E\}$.

The graph attention network (GAT) [32] is a widely used and advanced graph neural network architecture in graph representation learning. GAT models the constraints between

road segments in a road network by mapping them into a graph structure, allowing them to capture the spatial features of the network, which are then used as input to the model. By introducing a multi-head attention mechanism, GAT assigns distinct attention weights to each node and its neighboring nodes, enhancing the expressiveness of the attention layer. The model then generates a new representation vector for each road segment through a weighted summation of the nodes, effectively capturing the relationships and characteristics between road segments. Additionally, GAT employs a scoring function $e : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ to compute a score for each edge (j, i), reflecting the importance of the features of the neighbor node j to node i:

$$e(h_i, h_j) = \text{LeakyReLU}\Big(\mathbf{a}^\top \cdot \big[\mathbf{W}h_i \,\|\, \mathbf{W}h_j\big]\Big) \tag{1}$$

where $\mathbf{a} \in \mathbb{R}^{2d'}$, $\mathbf{W} \in \mathbb{R}^{d' \times d}$ is a learnable transformation matrix, $\|$ denotes vector concatenation, and LeakyReLU is the activation function whose negative input slope is 0.2 [32]. After calculating the attention scores for the node and all neighboring nodes, softmax is used for normalization:

$$\begin{aligned} z_{ij} &= e(h_i, h_j) \\ \alpha_{ij} &= \text{softmax}_j(z_{ij}) = \frac{\exp(z_{ij})}{\sum_{j' \in \mathcal{N}_i} \exp(z_{ij'})} \end{aligned} \tag{2}$$

Then, GAT computes the new representation of node i by taking a weighted average of the transformed features of its neighboring nodes, using normalized attention coefficients:

$$h_i' = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \cdot \mathbf{W}h_j\right) \tag{3}$$

where σ denotes a nonlinearity, and $\mathbf{W} \in \mathbb{R}^{d' \times d}$ is a learnable transformation matrix (Equation (1)).

In GATs, each node only attends to its neighboring nodes, using its representation as the query vector and the neighbors' representations as the keys. This means that the order of attention scores does not change with variations in the node's features. Consequently, the attention weights are static and do not dynamically adjust based on changes in node features. This approach is known as "static attention". However, in road networks, transition probabilities between segments are often uncertain, and interactions between nodes are complex. These factors may cause static attention to limit the GAT's ability to effectively fit training data in complex models.

To address these limitations, Brody [24] introduced GATv2, a variant of the graph neural network that implements dynamic attention by modifying the order of operations. In the standard GAT scoring function, the learned layers **W** and **a** are performed sequentially, which can lead the attention layer to collapse into a linear layer. GATv2 applies the **a** layer after the nonlinearity (LeakyReLU), and the **W** layer after the concatenation, effectively applying an MLP to compute the score for each query–key pair. This modification theoretically enhances the GATv2 to better fit training data.

- GATv2:

$$e(h_i, h_j) = \mathbf{a}^\top \text{LeakyReLU}\big(\mathbf{W} \cdot \big[h_i \,\|\, h_j\big]\big) \tag{4}$$

Additionally, GATv2 introduces a segment interaction frequency matrix computed from historical data to account for factors like user preferences and traffic conditions. This matrix expands the calculation of attention weights by integrating temporal and spatial characteristics of the road network. As a result, the model can more comprehensively and accurately capture spatiotemporal correlations within the road network, allowing it to adapt more effectively to complex road environments.

- GATv2-Enhanced:

$$e(h_i, h_j) = \mathbf{a}^\top \text{LeakyReLU}\left(\mathbf{W} \cdot \left[h_i \,\|\, h_j\right] + \mathbf{W}_0 I_{ij}\right) \tag{5}$$

where $h_i, h_j \in \mathbb{R}^{d_l}$ are road representations of v_i and v_j, $\mathbf{W}, \mathbf{W}_0 \in \mathbb{R}^{d_l \times d_{l+1}}$ are learnable parameters, LeakyReLU is the activation function whose negative input slope is 0.2 [32], and I_{ij} is the interaction frequency between v_i and v_j, the value of which can be calculated as:

$$I_{ij} = \text{count}(v_i \to v_j)/\text{count}(v_i), \tag{6}$$

where $\text{count}(v_i \to v_j)$ and $\text{count}(v_i)$ are the frequency of edges (v_i, v_j) and road v_i appearing in the trajectory dataset $\mathcal{D}$, respectively.

GATv2 takes into account the complex interactions caused by static road network structure and human mobility and finally represents the road segments containing spatiotemporal feature information as vector outputs. Since the attention weights generated by GATv2 are dynamic, each node has a different attention weight ranking, making it more expressive than GAT. In addition, this dynamic attention mechanism is more robust to noise.

3.2. Traffic Congestion-Aware Trajectory Encoder

With the ongoing advancements in natural language processing (NLP), technologies like LSTM [45], Transformers [46], and BERT [47] have been introduced and widely applied in various research areas. BERT, as a bidirectional encoder built on the Transformer architecture, effectively captures contextual information from both sides of a trajectory, facilitating comprehensive interaction between roads. Therefore, our work adopts BERT as the foundational model for training in the trajectory encoding layer.

After obtaining the segment representation sequences from the GATv2 layer, the next step is to convert these sequences into trajectory representations in the trajectory encoding layer while introducing a head information-sharing attention mechanism. Additionally, a traffic congestion sensor is designed to capture changes in road congestion by incorporating time semantics such as peak hours and weekdays. The trajectory encoding layer consists of two components: the trajectory cycle time refinement module and the adaptive shared attention module. Together, these components enable the model to more comprehensively and accurately understand and analyze the spatiotemporal characteristics of the road network.

3.2.1. Trajectory Cycle Time Refinement Module

This module utilizes week and day as time dimensions to capture the periodic traffic flow patterns on workdays. For each timestamp t_i associated with road segment v_i, vectors t_i^{day} and t_i^{min} are used to refine t_i into corresponding periodic indices. Specifically, t_i^{day} represents the workday index, while t_i^{min} denotes the minute index. Finally, by integrating the embedded representation of the road segment with the traffic period, a fused road segment vector representation is obtained:

$$x_i = S_i + t_i^{\text{day}} + t_i^{\text{min}} + e_i^{\text{pos}} \tag{7}$$

where S_i represents the road segment vector, t_i^{day} and t_i^{min} are the corresponding time representations, and e_i^{pos} is the positional encoding of the trajectory data. Finally, the concatenated road segment vector representations are combined to form the initial vector representation X of trajectory T:

$$X = \text{concat}_{i \in [1,T]}(x_i) \tag{8}$$

3.2.2. Adaptive Shared Attention Module

This module extracts segment transition durations within the same trajectory based on historical data, reflecting traffic congestion during peak hours. A hybrid matrix is introduced in this module, enabling simultaneous learning of key and query projections for all heads and allowing each head to adaptively reweight these projections, thus enhancing the expressive capacity of each head.

In the standard Transformer Encoder, self-attention is typically employed to learn the semantic information within the trajectory [47,48]. Given an input trajectory representation, linear transformations are applied to generate the query matrix Q, key matrix K, and value matrix V, followed by the calculation of attention scores:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{9}$$

Firstly, to reflect the degree of road congestion, this module leverages historical data from peak periods to extract a transition duration matrix for each road segment, thereby capturing the interaction relationships between segments. On this basis, an adaptive matrix $\tilde{\theta}$ is introduced to represent the impact of different segments on the self-attention mechanism.

$$\tilde{\theta} = \begin{bmatrix} \theta'_{1,1} & \theta'_{1,2} & \cdots & \theta'_{1,n} \\ \theta'_{2,1} & \theta'_{2,2} & \cdots & \theta'_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \theta'_{n,1} & \theta'_{n,2} & \cdots & \theta'_{n,n} \end{bmatrix} \tag{10}$$

In the matrix $\tilde{\theta}$, each element θ'_{ij} represents the weighting of the transition duration between segments v_i and v_j in the self-attention mechanism. If the transition time between segments is short, the value of θ'_{ij} is large, indicating that this segment pair has a stronger influence on self-attention. Conversely, if the transition time is longer, the influence of θ'_{ij} weakens accordingly.

For a given road segment v_i with timestamp t_i, the transition duration between any two segments is defined as $\theta_{i,j} = |t_i - t_j|$. To process the transition duration, a logarithmic function is applied, where $\theta'_{ij} = \frac{1}{\log(e+\theta_{ij})}$ (where $e \approx 2.718$). This ensures that as the transition duration increases, θ'_{ij} gradually decreases. Subsequently, a two-layer linear transformation is used to learn the transition time information between segments, effectively capturing the dynamic impact of transition duration on the relationship between segments:

$$\begin{aligned} L_1 &= \text{LeakyReLU}(\mu_1 \theta'_{ij}) \\ \tilde{\theta}_{ij} &= (L_1)\mu_2^T \end{aligned} \tag{11}$$

where μ_1 and μ_2 are learnable parameters, and L_1 is the result of the first layer of linear transformation. As a result, the attention score incorporating transition duration can be calculated as:

$$\begin{aligned} \text{Attention}_{\text{trans}}(Q, K, V) &= \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + \tilde{\theta}\right)V \\ Q = XW_Q, K &= XW_K, V = XW_V \end{aligned} \tag{12}$$

where the layer is parameterized by a query matrix $W_Q \in \mathbb{R}^{D_{\text{in}} \times D_k}$, a key matrix $W_K \in \mathbb{R}^{D_{\text{in}} \times D_k}$, and a value matrix $W_V \in \mathbb{R}^{D_{\text{in}} \times D_{\text{out}}}$.

The multi-head attention mechanism (MHA) generates the final output by concatenating the attention calculation results from each head, defined as Equation (13). However, when training large-scale Transformer models, the attention layer may encounter the problem of over-parameterization. Jean-Baptiste Cordonnier [49] demonstrated that re-parameterizing pre-trained multi-head attention layers can effectively reduce the number of parameters in the attention layer, thus alleviating this issue.

$$H^{(i)} = \text{Attention}_{\text{trans}}\left(XW_Q^{(i)}, XW_K^{(i)}, XW_V^{(i)}\right)$$
$$\text{MHA}(Q, K, V) = \text{concat}_{i\in[N_h]}[H^{(i)}]W_s \tag{13}$$

where $W_Q^{(i)}, W_K^{(i)} \in \mathbb{R}^{D_{\text{in}}\times d_k}$ and $W_V^{(i)} \in \mathbb{R}^{D_{\text{in}}\times d_{out}}$ are learned for each head $i \in [N_h]$, where N_h is the number of heads. In this context, d_k refers to the dimension of each head, and $D_k = N_h d_k$ represents the total dimension of the query/key space. The additional parameter matrix $W_s \in \mathbb{R}^{N_h d_{\text{out}}\times D_{\text{out}}}$ projects the concatenation of the outputs of all N_h heads (each of dimension d_{out}) to the final output space $\mathbb{R}^{D_{\text{out}}}$.

Next, we introduce a hybrid matrix into the attention layer, which already incorporates spatiotemporal semantics. This hybrid matrix enables the simultaneous learning of key and query projections for all heads, allowing each head to adaptively reweight these projections, thereby enhancing the expressiveness of each head. Our multi-head shared attention mechanism (MSA) is defined as follows:

$$H^{(i)} = \text{Attention}_{\text{trans}}\left(X\tilde{W}_Q\text{diag}(h_i), X\tilde{W}_K, XW_V^{(i)}\right)$$
$$\text{MSA}(Q, K, V) = \text{concat}_{i\in[N_h]}[H^{(i)}]W_s \tag{14}$$

where $\tilde{W}_Q \in \mathbb{R}^{D_{\text{in}}\times\tilde{D}_k}$ and $\tilde{W}_K \in \mathbb{R}^{D_{\text{in}}\times\tilde{D}_k}$ are the shared matrices, $\text{diag}(h_i)$ refers to constructing a diagonal matrix from the hybrid vector $h_i \in \mathbb{R}^{\tilde{D}_k}$, which defines a custom dot product over the $\tilde{D}_k$ projected dimensions of the shared matrices. $W_V^{(i)} \in \mathbb{R}^{D_{\text{in}}\times d_{\text{out}}}$ and $W_s \in \mathbb{R}^{N_h d_{\text{out}}\times D_{\text{out}}}$ are the same definition as Equation (13).

For MSA mechanism, instead of each head independently generating key and query matrices, a shared mechanism is used. Each head learns a hybrid vector $h_i \in \mathbb{R}^{\tilde{D}_k}$, which is projected onto the dimension $D_x \times \tilde{D}_k$ via shared matrices $\tilde{W}_Q$ and $\tilde{W}_K$. Then, a custom inner product operation is applied to the projection. The hybrid matrix is computed as:

$$H_{matrix} := \text{concat}_{i\in[N_h]}[h_i] \in \mathbb{R}^{N_h\times\tilde{D}_k}. \tag{15}$$

Figure 2 illustrates the process of using the hybrid matrix in the attention mechanism for the input vectors, where the hybrid vectors are arranged in rows. The total dimension $D_k = N_h d_k$ is defined, allocating a dimension d_k for each head to ensure alignment with the dimension assigned to the i-th head. By learning the hybrid vectors $\{h_i\}_{i\in[N_h]}$, the expressiveness of each head is further enhanced. Typically, d_k is set to 64 [50], but it can be adjusted according to specific requirements, allowing each head to focus on larger or smaller feature spaces.

With this head information-sharing attention mechanism, the Transformer Encoder achieves the following benefits when constructing and training large-scale models:

- Enhanced Expressive Power: Each head can adaptively utilize the shared matrix to capture feature information across different dimensions, enabling richer representation within the attention mechanism.
- Reduced Parameter Redundancy: By sharing the projection matrix across heads, the total number of model parameters is significantly reduced, which improves computational efficiency.
- Flexible Head Configuration: This mechanism allows for head dimensions to be adjusted as needed, enabling the attention mechanism to flexibly adapt to varying levels of complexity.

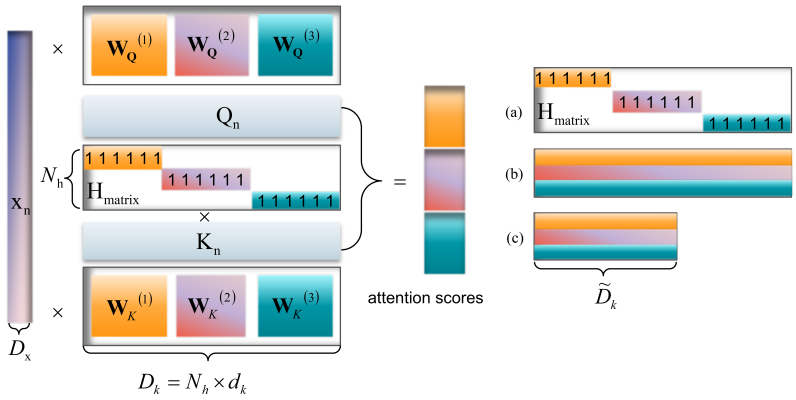

Figure 2. The calculation process of multi-head shared attention mechanism (MSA). For $N_h = 3$, the attention scores for input X_n are computed. The three independent heads are represented by different colors, and the block structure of the mixing matrix H_{matrix} ensures that the dot products for each head are performed on non-overlapping dimensions. (**a**) represents a more generalized hybrid matrix H_{matrix} as opposed to simple head concatenation, and the blocks-of-1 represents a ones matrix. (**b**) involves sharing head projections by learning all entries of the matrix. (**c**) reduces the number of projections from D_k to $\tilde{D}_k$, allowing heads to share redundant projections, thus improving efficiency.

3.2.3. Feedforward Network Layer

After the shared attention layer, a feedforward network layer with two linear transformations, activated by ReLU, is used to obtain the output representation Y of the trajectory T:

$$Y = \text{ReLU}(XW_1 + b_1)W_2 + b_2 \tag{16}$$

where W_1 and W_2 are two learnable parameters, while b_1 and b_2 are the biases for each respective layer.

3.3. Self-Supervised Pre-Training Tasks

During the model pre-training stage, an adaptive masked trajectory reconstruction task and a contrastive learning task are employed to strengthen the model's ability to capture co-occurrence relationships between roads, the spatiotemporal characteristics of trajectories, and their semantic information.

3.3.1. Adaptive Masked Trajectory Reconstruction Task

Masked trajectory reconstruction creates a self-supervised pre-training task by concealing part of the input sample data and leveraging a neural network to reconstruct the masked portions. The BERT model has demonstrated that, through masked reconstruction tasks, a model can learn universal feature representations, thus enhancing performance on downstream tasks [25].

Given a trajectory T, in the initial training phase, we adopt a random masking strategy, masking parts of each trajectory sequence to help the model learn reconstruction capabilities from relatively simple contextual information. At this stage, the masking ratio is kept low, making it easier for the model to predict using the unmasked trajectory information. As training progresses, the masking ratio is gradually increased. Special markers $[m_i]$ and $[m_t]$ replace the corresponding position v_i and timestamp t_i to obtain the masked trajectory subsequence M.

After obtaining the representation Y^{pre} of the masked trajectory T, a linear layer with parameters $W_m \in \mathbb{R}^{d \times |V|}$ and $b_m \in \mathbb{R}^{|V|}$ is used to predict the masked sequence:

$$Y^{\text{pre}} = Y^{\text{mask}} W_m + b_m \tag{17}$$

Next, the cross-entropy loss between the actual and predicted values of the masked sequence is used as the optimization objective:

$$H_{\tau}^{\text{mask}} = -\frac{1}{M}\sum_{i}\log\frac{\exp(Y_{v_i}^{\text{pre}})}{\sum_{v_j\in V}\exp(Y_{v_j}^{\text{pre}})} \tag{18}$$

Eventually, the masking ratio is gradually increased to cover a continuous subsequence of approximately l_{mask}% of the total trajectory length. At this stage, the model can maintain robust trajectory reconstruction performance under a higher masking ratio while significantly enhancing its ability to predict and transfer cross-sequence information.

3.3.2. Trajectory Contrastive Learning Task

Enhanced contrastive learning is a widely used framework that leverages data augmentation to generate different views of input samples, facilitating effective representation learning. The core principle is to maximize the similarity between views of the same sample while minimizing the similarity between views of different samples. In feature learning frameworks based on multi-view invariance, data augmentation is employed to capture multiple perspectives of the input samples. For data augmentation of trajectory sequences, both temporal dependency and variable dependency must be considered. Consequently, we adopt two data augmentation techniques—time-frequency perturbation and road segment masking—to create diverse views for contrastive learning.

Time-Frequency Perturbation: A subset of the trajectory sequence is randomly selected (with a ratio of $r_1 = 0.15$), and a noise factor is randomly added to this subset to generate a new sequence. The noise factor is defined as noise $= t_{\text{now}} - (t_{\text{now}} - t_{\text{his}}) \times r_2$, where $0.15 \leq r_2 \leq 0.30$, and t_{now} and t_{his} represent the average travel times for the current and historical periods, respectively. By applying transformations to travel time, this method helps the model capture semantic information in the temporal dimension.

Road Segment Masking In the adaptive masked trajectory reconstruction task, portions of the trajectory sequence and temporal information are randomly masked. The masked segments can be regarded as missing values within the trajectory sequence, allowing the model to further learn the spatiotemporal correlations of the trajectory by predicting the missing parts.

Subsequently, we adopt the normalized temperature-scaled cross-entropy loss as the contrastive objective [51]. A random selection of N_b trajectories is drawn from the dataset D, and after data augmentation, this yields $2N_b$ trajectories. For each trajectory (referred to as an anchor point), we find the corresponding augmented trajectory (positive pair) from among the $2(N_b - 1)$ negative samples in each batch. The contrastive loss function for positive pairs (i, j) is defined as follows:

$$\begin{aligned} P_{ij} &= \frac{\text{sim}(p_i, p_j)}{\tau} \\ H_{v_i}^{\text{con}} &= -\log\frac{\exp(P_{ij})}{\sum_{k=1}^{2N}\mathbf{1}_{[k\neq i]}\exp(P_{ik})} \end{aligned} \tag{19}$$

Finally, the average value of $H_{v_i}^{\text{con}}$ is calculated to obtain the contrastive loss H^{con} for learning, where τ is the temperature parameter, and $\text{sim}(p_i, p_j)$ represents the cosine similarity between p_i and p_j. The similarity function is defined as:

$$\text{sim}(p_i, p_j) = \frac{p_i \cdot p_j}{\|p_i\|\|p_j\|} \tag{20}$$

where p_k is a training sample in the batch, and $\mathbf{1}_{[k\neq i]}$ equals 1 if $k \neq i$; otherwise, it equals 0.

4. Results

4.1. Dataset Introduction

Our experiment uses two large-scale datasets: Porto and Beijing. The Porto dataset, from the Kaggle competition platform, contains open-source trajectory data collected from

1 July 2013 to 1 July 2014, with 435 user IDs and 695,085 trajectories (sampling frequency: 15 s). The Beijing dataset includes taxi trajectories recorded in November 2015 in Beijing, with 1677 user IDs and 1,018,312 trajectories. Each user ID represents a trajectory made up of multiple road segments, denoted as $R = \{R_1, R_2, R_3, \ldots, R_i, \ldots, R_n\}$. Table 1 provides details on the original datasets.

Table 1. Dataset description.

Field	Description
id	Unique trajectory ID
path	List of road segment IDs
tlist	List of timestamps (UTC)
usr_id	User ID
traj_id	Original trajectory ID
start_time	Travel start time

Note: The original trajectory dataset contains the field, with the corresponding description provided below.

Geographic data for Porto and Beijing were downloaded from the open-source map dataset OSM [52] to construct a directed road network graph $RG = (V, E, R_f, A)$. The OSM data comprise the following three components:

- The set of vertices $V = \{V_1, V_2, V_3, \ldots, V_i, \ldots, V_n\}$, where each road segment in the network is represented as a vertex.
- The set of edges E and binary adjacency matrix A, which are defined by connecting each pair of road segments that share a direct link. Each such connection is represented as an edge between two corresponding vertices.
- A feature matrix R_f that includes four key road characteristics: road type, length, lane count, and maximum speed limit. For each road in the adjacency matrix A, the in-degree and out-degree are computed to form these road attributes. Finally, the constructed directed road network RG is subsequently used as input for the GMTP model.

4.2. Training Details

Experimental Environment and Dataset Division: Our model is trained in an environment with Python 3.8 and PyTorch (2.3.0+cu118). The trajectory data are divided into training, validation, and test sets in specified proportions. For the Porto dataset, 65%, 17%, and 18% of the data are allocated to the training, validation, and test sets, respectively. For the Beijing dataset, the training, validation, and test sets comprise 60%, 20%, and 20% of the data, respectively.

Hyperparameter Settings: During both pre-training and fine-tuning, the model is optimized using the AdamW optimizer, with 30 training epochs, a batch size of 64, a learning rate of 0.0002, and an embedding dimension of 256. The GATv2 network has three layers with attention head sizes of [8, 16, 1]. The trajectory encoding module consists of six layers, each with eight attention heads. The trajectory masking ratio is set to 15%, the dropout rate is 0.1, and the temperature parameter τ is 0.05. The baseline model configuration is identical to that of GMTP.

4.3. Experimental Results and Analysis

Evaluation Metrics: Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and Root Mean Square Error (RMSE) are used to evaluate the GMTP model's performance on travel time prediction. The results, along with comparisons to five baseline models across two datasets, are shown in Table 2. The experimental results demonstrate that GMTP achieves better overall performance.

Table 2. Performance comparison of Porto and Beijing datasets.

Models	Porto			Beijing		
	MAE ↓	**MAPE ↓**	**RMSE ↓**	**MAE ↓**	**MAPE ↓**	**RMSE ↓**
Traj2vec	1.55	23.70	2.35	10.13	37.95	56.83
Transformer	1.74	25.72	2.64	10.74	39.61	57.16
BERT	1.59	24.63	2.29	10.21	37.31	37.31
PIM	1.56	24.68	2.34	10.19	39.04	57.73
START	1.33	20.66	2.00	9.134	30.92	35.40
GMTP	1.26	19.01	1.99	9.010	30.61	34.22

Note: ↓ indicates that lower values indicate better performance.

4.4. Ablation Study

To further investigate the effectiveness of each submodule, a series of ablation studies was conducted on the Porto dataset. Due to space constraints, only one metric is presented for each task, representing the average result across all repeated experiments.

4.4.1. The Impact of Enhanced GATv2

As shown in Figure 3, No-GATv2 replaces GATv2 with randomly initialized, learnable road segment embeddings. Node2vec replaces GATv2 with learnable road segment embeddings initialized by the Node2vec algorithm, which focuses only on the road network structure and ignores road segment features. GAT substitutes GATv2 by combining with the attention mechanism, but it performs worse than GATv2 because it overlooks variations in transition interaction frequency between road segments and exhibits lower robustness under noise interference.

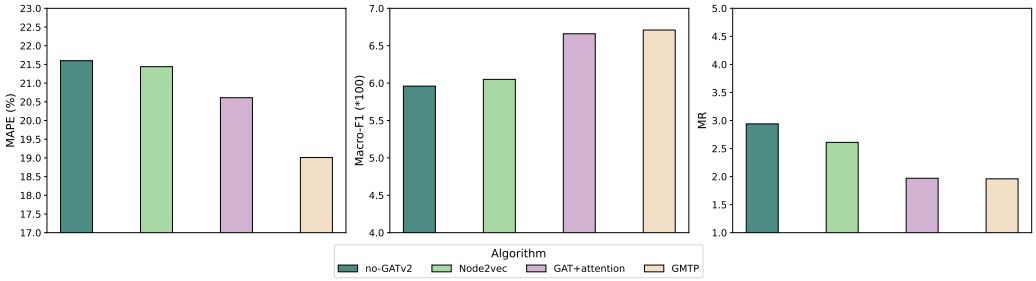

Figure 3. Effectiveness of various algorithm: Comparison of No-GATv2, Node2vec, GAT and GMTP, highlighting differences in embedding initialization and performance with respect to road features. Lower values represent better performance.

4.4.2. The Impact of Adaptive HSSTA

As shown in Figure 4, No-TimeEmb indicates that time characteristics, such as weekdays and peak hours, are not embedded in the vector representation of the trajectory. No-TransferMatrix means that transfer times between road segments are not considered, while No-MixingMatrix denotes that the mixing matrix is excluded from the attention weight calculation. Observations show that ignoring period characteristics results in a significant drop in model performance. Similarly, removing the transfer duration matrix and mixing matrix also degrades performance. This suggests that, compared to the standard multi-head attention mechanism (MHA), HSSTA can dynamically adjust the embedding representation and dimensions of the trajectory based on its spatiotemporal characteristics and the complexity of the attention pattern, leading to a more precise and effective parameter representation.

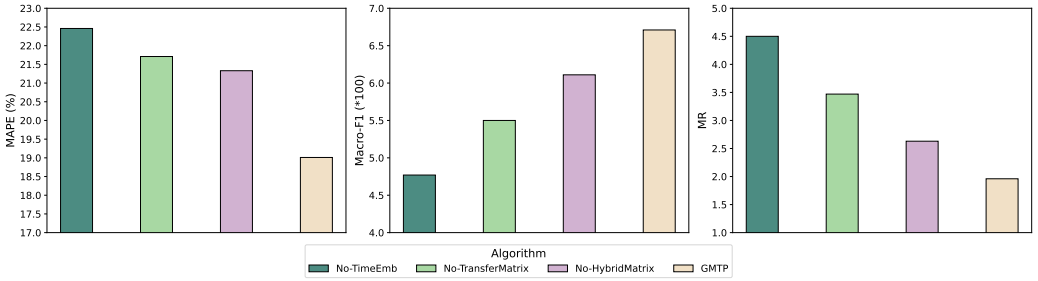

Figure 4. The impact of adaptive HSSTA. "No-TimeEmb" indicates the absence of time characteristics, "No-TransferMatrix" means transfer times are not considered, and "No-HybridMatrix" denotes the exclusion of the mixing matrix in attention weight calculation. Ignoring these components leads to a significant drop in model performance.

4.4.3. The Impact of Self-Supervised Tasks

As shown in Figure 5, No-Mask indicates that cross-entropy loss between the true values and predicted values of the masked sequence is not used for self-supervised training. No-Contra denotes that contrastive learning loss, which involves constructing positive and negative samples, is not applied for self-supervised training. Experimental results show that both self-supervised training methods have a significant impact on the final model performance.

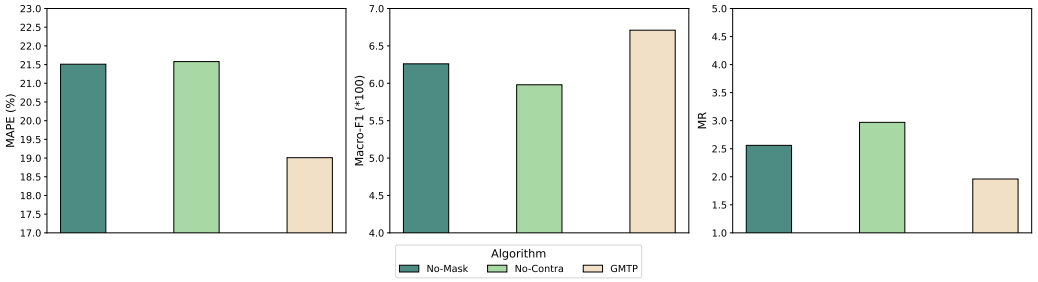

Figure 5. Impact of self-supervised tasks. In the "No-Mask" and "No-Contra" settings, the model's MAPE, Macro-F1, and MR values all increased, highlighting the significance of both trajectory masking and contrastive learning in self-supervised training.

4.4.4. The Impact of Data Augmentation Strategies

The primary objective of data augmentation is to enhance the model's generalization ability. We use a heatmap to visually represent the impact of different data augmentation method combinations on model performance. Darker colors indicate lower MAPE values, signifying better model performance. Figure 6 shows that the combination of Perturb and Mask achieves superior performance. These two methods are based on transformations of temporal characteristics in trajectories, underscoring the importance of capturing time-related features for accurate trajectory prediction.

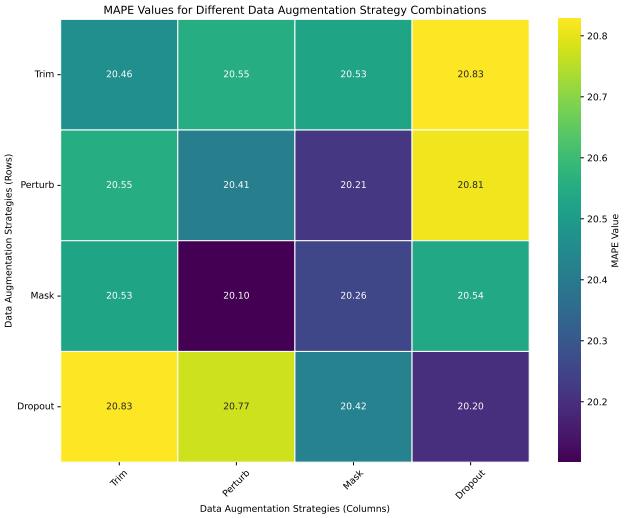

Figure 6. Impact of data augmentation strategies. Darker colors in the heatmap represent lower MAPE values, indicating better performance. The combination of Perturb and Mask yields the best results.

4.4.5. Hyperparameter Sensitivity Analysis

Figure 7 illustrates the impact of three key hyperparameters on model training. As the values of the encoding layer depth L, embedding dimension d, and batch size N increase, model performance improves. However, once these hyperparameters exceed a certain threshold, the model exhibits overfitting. This may be due to excessively large hyperparameter values introducing negative samples in contrastive learning that are too similar to the given anchor, reducing the model's ability to distinguish between samples.

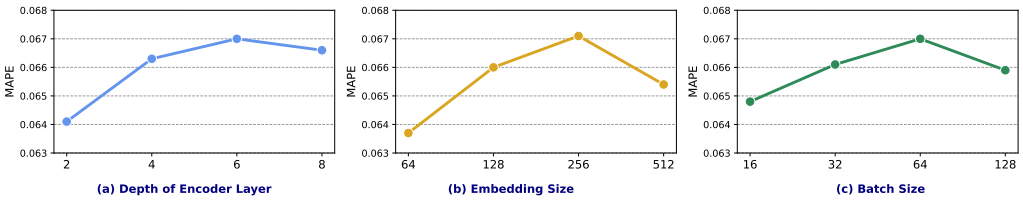

Figure 7. The impact of hyperparameters. The vertical axis represents Macro-F1. (**a**) Encoding layer depth (L): Balances learning capacity and overfitting. (**b**) Embedding dimension (d): Affects representation quality. (**c**) Batch size (N): Impacts gradient estimation and contrastive learning.

4.4.6. Performance and Computational Cost at Different Embedding Dimensions

As shown in Figure 8, we compared the trajectory encoding time costs of GMTP with five baseline models. GMTP requires more encoding time than Transformer and BERT due to the inclusion of temporal calculations. However, GMTP is faster than START, as it leverages a head information-sharing attention mechanism that adaptively adjusts the embedding dimension, improving computational efficiency. Additionally, GMTP outperforms traj2vec and PIM in speed, as the self-attention model has lower complexity in sequential processing than RNN-based models.

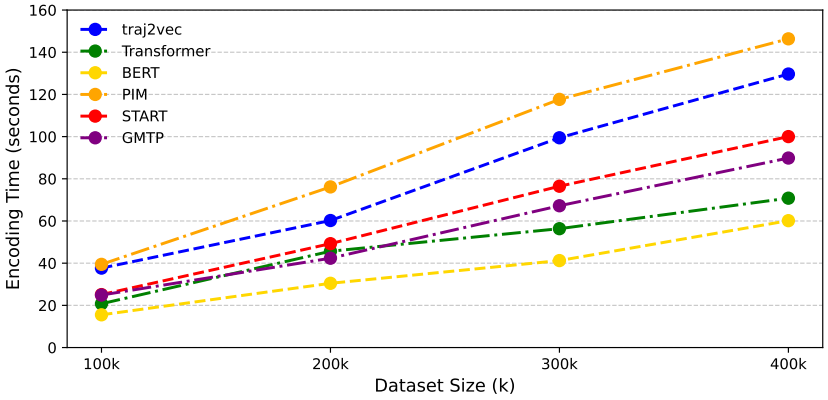

Figure 8. Trajectory encoding time comparison. The horizontal axis represents dataset size (in K), and the vertical axis represents encoding cost (in seconds).

To further assess the impact of the shared attention mechanism on model performance, we trained models with varying embedding dimensions for the Q/K shared attention mechanism on the Porto dataset. Table 3 indicates that as the embedding dimension decreased from 256 to 64, both the total parameter count and training time were reduced, while model performance remained relatively stable.

Table 3. Comparison of model performance, parameters, and training time across different embedding dimensions D_k.

D_k	R^2		Params ($\times 10^5$)		Time (h)	
	MHA	HSSTA	MHA	HSSTA	MHA	HSSTA
256	0.71	0.75	80.1	81.9	13.0	14.3
128	0.69	0.72	77.2	77.5	12.6	13.8
64	0.64	0.71	74.6	74.7	11.5	12.9

Note: MHA refers to multi-head attention, and HSSTA refers to the proposed adaptive head shared spatiotemporal attention.

5. Conclusions and Future Work

This paper proposes a new deep spatiotemporal neural network model (GMTP), which successfully integrates the multi-dimensional spatiotemporal characteristics of road networks and time series trajectories, improves the accuracy of travel time prediction to a certain extent, and reduces the cost of model training, providing an innovative solution for spatiotemporal data modeling in practical applications such as traffic management and logistics transportation.

First, GMTP enhances the road network's structural fitting by introducing GATv2 with a dynamic attention mechanism, which enables more accurate capture of spatiotemporal characteristics and improves the prediction of traffic congestion dynamics. The model further improves prediction accuracy by constructing a road segment interaction transfer frequency matrix, which captures the deep spatial and temporal correlations in the road network. Second, the proposed spatiotemporal self-attention mechanism efficiently captures time-specific features, such as peak hour characteristics and road section transfer durations, while reducing training parameters, maintaining model stability, and significantly boosting prediction efficiency. Lastly, the model's cross-sequence transfer capability and generalization performance are further strengthened through self-supervised learning tasks. Experimental results, validated through simulations, demonstrate the model's superior performance in travel time prediction.

Despite these achievements, the method has some limitations. On one hand, the model currently relies solely on vehicle GPS data and does not incorporate other data sources, such as POI check-in trajectories and Beidou positioning data. On the other hand, while

the model performs well in experiments, its application in real-world traffic scenarios needs further validation. Future research will integrate additional multimodal data to enhance the model's adaptability and scope and apply it to visual demonstrations and performance evaluations of practical scenarios to advance the development of intelligent transportation systems.

Author Contributions: Conceptualization, T.L. and Y.L.; methodology, T.L.; software, T.L.; validation, T.L. and Y.L.; formal analysis, T.L.; investigation, T.L.; resources, Y.L.; data curation, T.L.; writing—original draft preparation, T.L.; writing—review and editing, T.L. and Y.L.; visualization, T.L.; supervision, Y.L.; project administration, Y.L.; funding acquisition, Y.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: https://github.com/liuting001001/GMTP/tree/master.

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

References

1. Kong, X.; Li, M.; Ma, K.; Tian, K.; Wang, M.; Ning, Z.; Xia, F. Big trajectory data: A survey of applications and services. *IEEE Access* **2018**, *6*, 58295–58306. [CrossRef]
2. Yue, Y.; Zhuang, Y.; Li, Q.; Mao, Q. Mining time-dependent attractive areas and movement patterns from taxi trajectory data. In Proceedings of the 2009 17th International Conference on Geoinformatics, Fairfax, VA, USA, 12–14 August 2009; pp. 1–6.
3. Fang, Z.; Pan, L.; Chen, L.; Du, Y.; Gao, Y. MDTP: A multi-source deep traffic prediction framework over spatiotemporal trajectory data. *Proc. VLDB Endow.* **2021**, *14*, 1289–1297. [CrossRef]
4. Wang, J.; Wu, N.; Zhao, W.X.; Peng, F.; Lin, X. Empowering A*search algorithms with neural networks for personalized route recommendation. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 539–547.
5. Wang, J.; Jiang, J.; Jiang, W.; Li, C.; Zhao, W.X. Libcity: An open library for traffic prediction. In Proceedings of the 29th International Conference on Advances in Geographic Information Systems, Beijing, China, 2–5 November 2021; pp. 145–148.
6. Ji, J.; Wang, J.; Jiang, Z.; Jiang, J.; Zhang, H. STDEN: Towards physics-guided neural networks for traffic flow prediction. In Proceedings of the AAAI Conference on Artificial Intelligence, Philadelphia, PA, USA, 27 February–2 March 2022; pp. 4048–4056.
7. Wang, J.; Lin, X.; Zuo, Y.; Wu, J. Dgeye: Probabilistic risk perceptionand prediction for urban dangerous goods management. *ACM Trans. Inf. Syst.* **2021**, *39*, 28:1–28:30. [CrossRef]
8. Li, G.; Hung, C.; Liu, M.; Pan, L.; Peng, W.; Chan, S.G. Spatial temporal similarity for trajectories with location noise and sporadic sampling. In Proceedings of the 37th IEEE International Conference on Data Engineering, (ICDE), Chania, Greece, 19–22 April 2021; pp. 1224–1235.
9. Amirian, P.; Basiri, A.; Morley, J. Predictive analytics for enhancing travel time estimation in navigation apps of Apple, Google, and Microsoft. In Proceedings of the 9th ACM SIGSPATIAL International Workshop on Computational Transportation Science, Burlingame, CA, USA, 31 October 2016; pp. 31–36.
10. Zin, T.T.; Hama, H. A robust road sign recognition using segmentation with morphology and relative color. *J. Inst. Image Inf. Telev. Eng.* **2005**, *59*, 1333–1342. [CrossRef]
11. Stessens, P.; Khan, A.Z.; Huysmans, M.; Canters, F. Analysing urban green space accessibility and quality: A GIS-based model as spatial decision support for urban ecosystem services in Brussels. *Ecosyst. Serv.* **2017**, *28*, 328–340. [CrossRef]
12. Yildirimoglu, M.; Geroliminis, N. Experienced travel time prediction for congested freeways. *Transp. Res. Part B Methodol.* **2013**, *53*, 45–63. [CrossRef]
13. Liu, Y.; Jia, R.; Ye, J.; Qu, X. How machine learning informs ride-hailing services: A survey. *Commun. Transp. Res.* **2022**, *2*, 100075. [CrossRef]
14. Simroth, A.; Zähle, H. Travel time prediction using floating car data applied to logistics planning. *IEEE Trans. Intell. Transp. Syst.* **2010**, *12*, 243–253. [CrossRef]
15. Carrion, C.; Levinson, D. Value of travel time reliability: A review of current evidence. *Transp. Res. Part A Policy Pract.* **2012**, *46*, 720–741. [CrossRef]
16. Wang, H.; Tang, X.; Kuo, Y.H.; Kifer, D.; Li, Z. A simple baseline for travel time estimation using large-scale trip data. *ACM Trans. Intell. Syst. Technol. (TIST)* **2019**, *10*, 1–22. [CrossRef]
17. Wang, Y.; Zheng, Y.; Xue, Y. Travel time estimation of a path using sparse trajectories. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 24–27 August 2014; pp. 25–34.

18. Li, Y.; Fu, K.; Wang, Z.; Shahabi, C.; Ye, J.; Liu, Y. Multi-task representation learning for travel time estimation. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 1695–1704.
19. Wang, D.; Zhang, J.; Cao, W.; Li, J.; Zheng, Y. When will you arrive? Estimating travel time based on deep neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32.
20. Fang, X.; Huang, J.; Wang, F.; Zeng, L.; Liang, H.; Wang, H. Constgat: Contextual spatial-temporal graph attention network for travel time estimation at baidu maps. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Virtual, 6–10 July 2020; pp. 2697–2705.
21. Wang, Z.; Fu, K.; Ye, J. Learning to estimate the travel time. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 858–866.
22. Cheng, H.T.; Koc, L.; Harmsen, J.; Shaked, T.; Chandra, T.; Aradhye, H.; Anderson, G.; Corrado, G.; Chai, W.; Ispir, M.; et al. Wide & deep learning for recommender systems. In Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, Boston, MA, USA, 15 September 2016; pp. 7–10.
23. Jiang, J.; Pan, D.; Ren, H.; Jiang, X.; Li, C.; Wang, J. Self-supervised trajectory representation learning with temporal regularities and travel semantics. In Proceedings of the 2023 IEEE 39th International Conference on Data Engineering (ICDE), Anaheim, CA, USA, 3–7 April 2023; pp. 843–855.
24. Brody, S.; Alon, U.; Yahav, E. How attentive are graph attention networks? *arXiv* **2021**, arXiv:2105.14491.
25. Devlin, J. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
26. Yu, R.; Li, Y.; Shahabi, C.; Demiryurek, U.; Liu, Y. Deep learning: A generic approachfor extreme condition traffic forecasting. In Proceedings of the 2017 SIAM international Conference on Data Mining, Houston, TX, USA, 27–29 April 2017; pp. 777–785.
27. Wu, Z.; Pan, S.; Long, G.; Jiang, J.; Zhang, C. Graph wavenet for deep spatial-temporal graph modeling. *arXiv* **2019**, arXiv:1906.00121.
28. Zheng, C.; Fan, X.; Wang, C.; Qi, J. Gman: A graph multi-attention network for traffic prediction. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; pp. 1234–1241.
29. Wang, X.; Ma, Y.; Wang, Y.; Jin, W.; Wang, X.; Tang, J.; Jia, C.; Yu, J. Traffic flow prediction via spatial temporal graph neural network. In Proceedings of the Web Conference 2020, Taipei, Taiwan, 20–24 April 2020; pp. 1082–1092.
30. Jin, G.; Wang, M.; Zhang, J.; Sha, H.; Huang, J. STGNN-TTE: Travel time estimation via spatial–temporal graph neural network. *Future Gener. Comput. Syst.* **2022**, *126*, 70–81. [CrossRef]
31. Guo, S.; Lin, Y.; Feng, N.; Song, C.; Wan, H. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 922–929.
32. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph attention networks. *STAT* **2017**, *1050*, 10-48550.
33. Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P.J. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* **2020**, *21*, 1–67.
34. Lewis, M. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv* **2019**, arXiv:1910.13461.
35. Liu, Y. Roberta: A robustly optimized bert pretraining approach. *arXiv* **2019**, arXiv:1907.11692.
36. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language models are unsupervised multitask learners. *Openai Blog* **2019**, *1*, 9.
37. Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv* **2023**, arXiv:2307.09288.
38. Almazrouei, E.; Alobeidli, H.; Alshamsi, A.; Cappelli, A.; Cojocaru, R.; Debbah, M.; Goffinet, É.; Hesslow, D.; Launay, J.; Malartic, Q.; et al. The falcon series of open language models. *arXiv* **2023**, arXiv:2311.16867.
39. Chen, Y.; Li, X.; Cong, G.; Bao, Z.; Long, C.; Liu, Y.; Chandran, A.K.; Ellison, R. Robust road network representation learning: When traffic patterns meet traveling semantics. In Proceedings of the CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Online, 1–5 November 2021; pp. 211–220.
40. Hadsell, R.; Chopra, S.; LeCun, Y. Dimensionality reduction by learning an invariant mapping. In Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR'06, New York, NY, USA, 17–22 June 2006; Volume 2, pp. 1735–1742.
41. He, K.; Fan, H.; Wu, Y.; Xie, S.; Girshick, R. Momentum contrast for unsupervised visual representation learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 9729–9738.
42. Gao, T.; Yao, X.; Chen, D. SimCSE: Simple contrastive learning of sentence embeddings. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Punta Cana, Dominican, 7–11 November 2021; pp. 6894–6910.
43. Van Den Oord, A.; Li, Y.Z.; Vinyals, O. Representation Learning with Contrastive Predictive Coding [Online]. Available online: https://arxiv.org/abs/1807.03748 (accessed on 22 January 2019).
44. Khosla, P.; Teterwak, P.; Wang, C.; Sarna, A.; Tian, Y.; Isola, P.; Maschinot, A.; Liu, C.; Krishnan, D. Supervised contrastive learning. In Proceedings of the Advances in Neural Information Processing Systems, Online, 6–12 December 2020; Volume 33, pp. 18661–18673.
45. Hochreiter, S. Long Short-term Memory. In *Neural Computation*; MIT-Press: Cambridge, MA, USA, 1997.

46. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2017; pp. 5998–6008.
47. Bhat, M.; Francis, J.; Oh, J. Trajformer: Trajectory prediction with local self-attentive contexts for autonomous driving. *arXiv* **2020**, arXiv:2011.14910.
48. Chen, Z.; Xiao, X.; Gong, Y.J.; Fang, J.; Ma, N.; Chai, H.; Cao, Z. Interpreting trajectories from multiple views: A hierarchical self-attention network for estimating the time of arrival. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 14–18 August 2022; pp. 2771–2779.
49. Cordonnier, J.B.; Loukas, A.; Jaggi, M. Multi-head attention:Collaborate instead of concatenate. *arXiv* **2020**, arXiv:2006.16362.
50. Khan, S.; Naseer, M.; Hayat, M.; Zamir, S.W.; Khan, F.S.; Shah, M. Transformers in vision: A survey. *Acm Comput. Surv. (CSUR)* **2022**, *54*, 1–41. [CrossRef]
51. Chen, T.; Kornblith, S.; Norouzi, M.; Hinton, G. A simple framework for contrastive learning of visual representations. In Proceedings of the International Conference on Machine Learning, PMLR, Virtual, 13–18 July 2020; pp. 1597–1607.
52. OpenStreetMap Contributors. 2017. Available online: https://www.openstreetmap.org (accessed on 20 September 2021).