Automatic Road Extraction with Multi-Source Data Revisited: Completeness, Smoothness and Discrimination

Haitao Yuan*
NTU, Singapore
haitao.yuan@ntu.edu.sg

Sai Wang*
BJTU, China
wangsai18@bjtu.edu.cn

Zhifeng Bao
RMIT University, Australia
zhifeng.bao@rmit.edu.au

Shangguang Wang
BUPT, China
sgwang@bupt.edu.cn

ABSTRACT

Extracting roads from multi-source data, such as aerial images and vehicle trajectories, is an important way to maintain road networks in the filed of urban computing. In this paper, we revisit the problem of road extraction and aim to boost its accuracy by solving three significant issues: the insufficient complementarity among multiple sources, rough edges of extracted roads, and many false positives caused by confusing pixels. In particular, we design an end-to-end neural network model to achieve this goal. At first, this model leverages two encoding networks to extract relative information from the inputs of two sources respectively, and then applies the attention mechanism to fuse them for sufficiently capturing the complementary correlation. Next, we introduce an auxiliary task, predicting road edges based on fused representations, to make the extracted roads smooth and continuous. At last, to reduce false positives relative to confusing pixels, we propose a pixel-aware contrastive-learning module to distinguish positive (roads) and negative (objects similar to roads) pixels. In addition, to improve the model's learning effectiveness, we propose a model-agnostic transfer learning method, which first builds auxiliary tasks to pre-train the whole model, and then fine-tunes the model's parameters for the main task. Extensive experiments on real datasets verify the superiority of our method as well as the importance of solving the three issues outlined above.

PVLDB Reference Format:
Haitao Yuan, Sai Wang, Zhifeng Bao, Shangguang Wang. Automatic Road Extraction with Multi-Source Data Revisited: Completeness, Smoothness and Discrimination. PVLDB, 16(11): 3004 - 3017, 2023.
doi:10.14778/3611479.3611504

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/BubbleSai/DICN.

1 INTRODUCTION

As the core of urban computing, road network plays an important role in its sub-fields such as map navigation and road status recognition [22, 35, 45, 48, 56–60]. However, due to the change/construction of some old/new roads, road networks would be out of date, which could lead to serious traffic hazards, such as traffic congestion and navigation error. Therefore, automatically extracting roads from aerial images [19, 34, 37, 54, 55] becomes a significant way to address this issue. There are two typical learning-based paradigms of solving the road extraction problem: traditional [20, 46, 49] and end-to-end [4, 23, 25]. In particular, the traditional learning-based methods first manually design road-related features to identify coarse road regions in aerial images, and then apply traditional machine learning methods, such as Support Vector Machine and Markov Random Field, to extract fine-grained roads. Differently, the end-to-end methods regard the problem as an object segmentation problem, and exploit the powerful representation ability of neural networks to segment roads in aerial images.

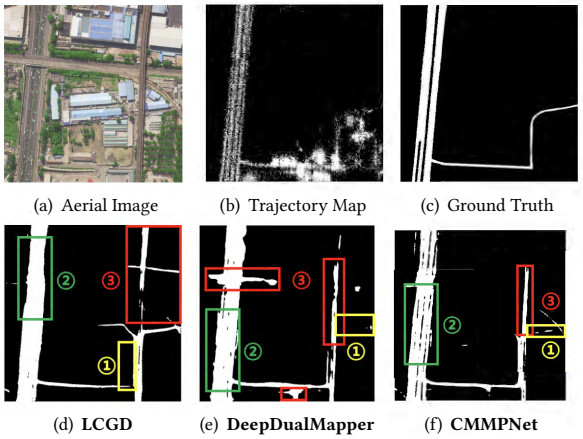


Figure 1: Three Challenges Faced by Existing Multi-source Fusion Methods.

Due to deviated photographic angles and extreme weather such as snow and fog, aerial images may have low quality, which would greatly affect the above methods' performance. In addition, images captured at different time periods also cause great fluctuations [41]. Instead of purely using the single-source data, some studies [28, 41, 52] try to introduce the data from other sources such as vehicles' GPS trajectories. In particular, the regions with a large number of trajectories can be considered as potential roads. To this end, we can take the complementary correlation between different sources into consideration for the road extraction problem. For example, the areas occluded by trees in aerial images can be recovered by GPS information, and the missing road sections with few vehicles in GPS trajectories can be predicted with the help of aerial images. To better leverage multi-source data, existing methods focus on designing effective modules to fuse them. For example, **LCGD** [41] directly aggregates multi-source inputs, which cannot fully capture the complex correlation between different sources. Hence, **DeepDualMapper** [52] feeds inputs of two sources into two different deep networks, and then fuses two encoding results

Proceedings of the VLDB Endowment, Vol. 16, No. 11 ISSN 2150-8097.
doi:10.14778/3611479.3611504

Haitao Yuan is the corresponding author. The asterisk (*) indicates equal contribution.

through a gated module. However, the fusion strategy is too pale to make the generalization performance poor. To address these issues, **CMMPNet** [28] proposes a dual enhancement module, including a message passing mechanism, to fuse and refine intermediate feature maps of different sources.

However, three open challenges still remain unsolved by existing multi-source fusion methods.

(1) **Insufficient complementarity.** Existing methods leverage the concatenating operation or the weighted summation to fuse multi-source features. However, if the road region is occluded and there are also few GPS trajectories, these fusion approaches would not make full use of the complementary correlation. For example, the regions marked with ① in Figure 1 point out the incomplete extraction results of existing methods.

(2) **Rough edges.** Actually, the edge of a road is usually smooth and consistent, but existing methods have no specific optimization based on this constraint, which could easily lead to rough and discontinuous edges of the extracted roads. In particular, we mark these roads with ② in Figure 1.

(3) **Confusing pixels.** In aerial imagery, certain municipal facilities, such as railways and bridges, bear similar attributes to roads in terms of shape and color. While trajectory data can facilitate the differentiation between roads and these facilities, complications arise primarily when such structures are in proximity to roads. Existing methodologies do not adequately address this scenario, resulting in difficulties distinguishing these potentially confusing pixels. As shown in Figure 1, there are a large number of false positives, which are marked with ③.

To this end, we propose a novel fusion model, termed Dual Information Crossing Network (**DICN**), to address the above challenges. Specifically, **DICN** is built on a dual-stream architecture, where we first leverage two encoding modules to respectively encode the input features of two sources, and then propose a fusion module to fuse them. AS compared to existing studies, our work introduces a fine-grained fusion module, ingeniously inspired by the attention mechanism [44]. This module is unique in its application of a self-attention operator on the concatenated representation of dual-source features, generating a global encoding that captures comprehensive information from both sources. Following this, we employ a cross-attention operator to extract complementary information from the global encoding for each source. This approach allows for a thorough capture of complementary correlation by the fusion module, **effectively addressing the first challenge**. Furthermore, we introduce an independent sub-task focused on predicting road edges. This sub-task utilizes the semantic features extracted from aerial images and is designed to maintain the consistency and smoothness of road edges. This explicit consideration of smooth and continuous edges in our DICN model is a first of its kind in the field, a feature not found in existing methods [28, 41, 52], **thereby solving the second challenge**. To tackle the confusion between road pixels and surrounding background pixels, we propose a pixel-aware contrastive learning module. This module is designed to distinguish the confusing encoding representations of pixels by marking positive and negative with respect to the ground truth. This approach allows pixels with similar shape and color to be classified into two distinct classes, **thereby addressing the last challenge**. Furthermore, we introduce a model-agnostic transfer learning method to enhance the learning of DICN. This method first constructs external tasks to pre-train the entire model, followed by fine-tuning it in the main task.

In summary, we make the following technical contributions:

- We design a comprehensive neural network model, **DICN**, that can fully exploit multi-source features to achieve an accurate road extraction. (Sec. 3)
- We leverage the attention mechanism to fuse multi-source features, which can make full use of the complementary correlation between two sources, and also propose an auxiliary task, edges prediction, to explicitly capture the smoothness of road edges. (Sec. 4)
- We design a pixel-aware contrastive learning module to achieve the model's discrimination ability for confusing pixels, and apply a transfer learning method to improve the model learning. (Sec. 5)
- We conduct a comprehensive evaluation on two real-world datasets and find **DICN** significantly outperforms state-of-the-art in terms of accuracy and robustness. (Sec. 6)

2 PRELIMINARY

In this section, we first describe the data source from two modalities (i.e., aerial image and GPS trajectory), and then present how to convert the GPS data into trajectory map, which is a single-channel image. At last, we formally define the road extraction problem, which takes a pair of aerial image and trajectory map as input.

2.1 Data Source

The data source consists of two modalities: aerial images from satellite cameras and GPS trajectories from vehicles. Given a region of a city, we take a corresponding aerial image and all GPS trajectories passing through this region into account. On the one hand, the aerial image I can be regarded as a tensor with the shape $\mathbb{R}^{H\times W\times C}$, where $H\times W$ represents the spatial resolution and C denotes the channel information. In particular, the spatial resolution, depending on satellite imaging equipments (usually between $10cm\times 10cm$ and $100cm\times 100cm$ per pixel), determines the size of the actual area represented by each pixel block. The lower the resolution is, the bigger the area is. On the other hand, all GPS trajectories are collected from in-vehicle devices, which record vehicles' actual positions on the road network. Specifically, each GPS trajectory can be regarded as a series of GPS points. To extract the road network, we consider the spatial coordinates (i.e., the longitude lon and the latitude lat) for each GPS point, which is denotes as $G_{coor} = (lon,\ lat)$. Hence, we have the set of points $\{G_{coor}\}$ from all GPS trajectories for the given region.

2.2 Trajectory Map Generation

Considering the gap between the discrete GPS data $\{G_{coor}\}$ and the continuous road network, we follow the existing work [28, 41] to convert $\{G_{coor}\}$ into a single-channel image $T \in \mathbb{R}^{H\times W}$, namely trajectory map. This process consists of the following four steps: **1. Data Filtering:** At first, we extract valid GPS points from $\{G_{coor}\}$ according to the coordinate ranges for the given region or aerial image. **2. Creating Initial Map:** Next, we initiate a single-channel image $T \in \mathbb{R}^{H\times W}$ with zero values, and then project valid GPS

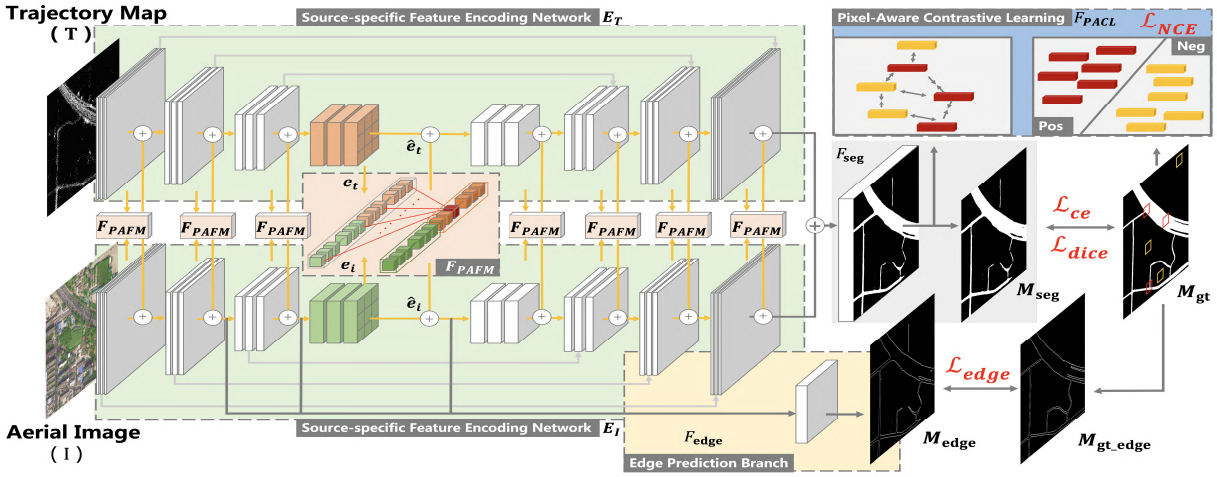


Figure 2: The Architecture of Dual Information Crossing Network

points into the image for getting non-zero values. More specifically, we normalize each point into the range of $H \times W$, and then count the number of located points at each pixel. Hence, the pixel with denser GPS points corresponds to a larger non-zero value. **3. Improving Map Continuity:** However, the GPS data is inherently sparse due to the sampling interval, so only some pixels in the trajectory map T contain non-zero values. In order to maintain the continuity, we employ a morphological operations to process the trajectory map, where we expand the range of data points through dilation operations, and connect trajectory segments that are closer together. **4. Smoothing the Map:** Finally, we leverage Kernel Density Estimation [7] and Gaussian kernel Filter to make the trajectory map more smooth.

Discussion. (1) Trajectory map generation is indeed a common pre-process for the road extraction task, and thus is orthogonal to the main focus of our current study. To maintain a fair comparison, we employed the same method as that of the baseline approach. This ensures that any observed differences in the results are attributable to our proposed solution rather than variances in pre-processing. (2) Although the sequence information (i.e., the order and timing of individual GPS points) is lost in the process of creating the trajectory map, the collective patterns of all trajectories, such as congestion areas, and frequently traveled paths, are preserved. Such collective information provides valuable insights for tasks such as road extraction. (3) In the context of our research problem, we are provided with temporally aligned multi-source data. Therefore, the temporal alignment between trajectories and aerial images is not the primary focus of this paper. (4) To detect road closures, users can compare different road extraction results by our model from trajectories recorded on different days.

2.3 Problem Definition

Given an aerial image $I \in \mathbb{R}^{H \times W \times C}$ and its corresponding trajectory map $T \in \mathbb{R}^{H \times W}$, our objective is to learn a mapping function $\mathcal{F}$ to generate the road extraction result $M \in \mathbb{R}^{H \times W}$, which can be regarded as a binary classification problem for each pixel. In other words, each element M_{ij} would be set as 1 if the relative pixel is passed by a road; otherwise, it would be set as 0. Formally, the problem can be formulated as follows: $M = \mathcal{F}([I,\ T]\,|\theta)$, where θ represents all learnable weights of the function.

3 MODEL OVERVIEW

In this section, we outline the architecture of our proposed **DICN**, and explain how it incorporates the desired components. As shown in Figure 2, the whole framework consists of three tasks: one main task (i.e., generating the road extraction result M_{seg}) and two auxiliary tasks (i.e., the edge prediction task and the pixel-aware contrastive learning task). Hence, we introduce different modules of **DICN** and describe the pipeline according to these three tasks respectively.

The aim of the main task is to learn a function to generate the road extraction result M_{seg} based on the given inputs (i.e., a trajectory map T and an aerial image I). Generally, this procedure includes two phases: encoding input features and inferring outputs based on encoded results. **In the first phase**, to fully extract the information specific to different sources, we first design two source-specific feature encoding networks, denoted by E_I and E_T, to generate hidden representations for the two inputs (i.e., an aerial image I and a trajectory T) respectively. In particular, both E_I and E_T adopt the UNet structure [39], which is empirically proven to be useful for the object extraction problem [66]. More specifically, the UNet structure consists of four encoder blocks and four decoder blocks, which are connected by a serial way, and each decoder block additionally has a skip connection with the corresponding encoder block. The encoder blocks are designed to progressively downsample the input and extract higher-level semantic features. Conversely, the decoder blocks gradually upsample the features back to the original scale and refine the detailed information. Adding more encoder-decoder blocks could potentially extract deeper or more complex features. However, this also increases the risk of overfitting, especially when training data are limited. Consequently, we have chosen to employ four encoder and decoder blocks in our experimental design, and this has been shown to be optimal for our data in [28]. Formally, given the aerial image $I \in \mathbb{R}^{H \times W \times C}$ and trajectory maps $T \in \mathbb{R}^{H \times W}$ as input, the feature encoders E_I and E_T extract different features corresponding to different modal

information as follows:

$$\begin{aligned} \{e_i^1,\ e_i^2, ...,\ e_i^8\} &= E_I(I) \\ \{e_t^1,\ e_t^2, ...,\ e_t^8\} &= E_T(T) \end{aligned} \tag{1}$$

Next, to capture the correlation between different sources, we design the module F_{PAFM} to fuse $\{e_i^1,\ e_i^2, ...,\ e_i^8\}$ and $\{e_t^1,\ e_t^2, ...,\ e_t^8\}$. Instead of directly concatenating or aggregating them, which cannot complete the deep crossover of features between the two modalities, we leverage the attention mechanism to achieve the goal. The reason is that the complementary correlation between two modalities is dynamic in different encoding stages, which can be captured by the adaptive weight in the attention mechanism. Formally, the fused process is formulated as follows:

$$\begin{aligned} \hat{e}_i^n,\ \hat{e}_t^n &= F_{PAFM}(e_i^n,\ e_t^n), & n \in \{1, 2, \cdots, 8\} \\ e_i^n &= e_i^n\ +\ \hat{e}_i^n, & n \in \{1, 2, \cdots, 8\} \\ e_t^n &= e_t^n\ +\ \hat{e}_t^n, & n \in \{1, 2, \cdots, 8\} \end{aligned} \tag{2}$$

where $\hat{e}_i^n,\ \hat{e}_t^n$ represents the corresponding attention representations for two modalities, and we respectively add them to original representations for getting final fused results. **In the second phase**, we apply the segmentation head module F_{seg} to convert the last encoding results e_i^8, e_t^8 into the road extraction result $M_{seg} \in \mathbb{R}^{H \times W}$, which is formulate as follows:

$$M_{seg} = Bin(F_{seg}(e_i^8 \oplus e_t^8), \mu) \tag{3}$$

Here, $\oplus$ represents the concatenation operation; F_{seg} consists of convolutional layers and activation functions, which are used to reduce the dimension of features and generate a predicted probability map; *Bin* represents the binarization functiona and μ is set to 0.5, which can convert the value larger than the threshold μ into 1, otherwise 0.

To alleviate rough edge segmentation results, we introduce the *edge prediction task* to assist in learning feature encoding networks (e.g., E_T, E_I, F_{PAFM}). In particular, we first generate the ground truth M_{gt_edge} based on the ground truth of the main task M_{gt}, and then design the module edge prediction branch, F_{edge}, to convert three selected feature encodings e_i^2, e_i^3, e_i^4 into the predicted result M_{edge}. We compute the loss $\mathcal{L}_{edge}$ between M_{edge} and M_{gt_edge} to jointly optimize related neural networks with losses in the main task. Notably, the reason of taking as input e_i^2, e_i^3, e_i^4 is two-fold. On the one hand, the trajectory map T is built on discrete GPS points, and has rough road edges, which would deteriorate the edge prediction performance, so we avoid directly leverage all encoding outputs of T. On the other hand, the process of edge prediction is to distill the edge information from the aerial image, so the shallow blocks containing more detailed information are more helpful. To leverage the information of texture and shape, we select the first three encoding blocks' outputs e_i^2, e_i^3, e_i^4 as the input of this auxiliary task.

In addition, we design a pixel-aware contrastive learning task to explicit regularize foreground and background pixels to promote the discriminative ability of **DICN**. At first, we sample some pixels from the ground truth M_{gt} to build pixel labels for the task, and then extract pixel embeddings from the feature map in the segmentation head module F_{seg} as the inputs. At last, we compute the contrastive loss [36, 47] $\mathcal{L}_{NCE}$ based on these pixel embeddings and labels.

4 MODEL STRUCTURE

In this section, we will describe different components of the model in detail. In Section 4.1, we propose the source-specific feature encoding networks (i.e., E_T and E_I) to individually facilitate feature extraction of two modalities (i.e., T and I). In Section 4.2, we describe the pixel-attention fusion module F_{PAFM} to achieve feature crossover between two modalities, and explain how to concatenate E_T, E_I and F_{PAFM} to generate informative encoding results. In Section 4.3, we describe the edge prediction branch F_{edge} to assist in predicting road edges.

4.1 Source-specific Feature Encoding Network

As illustrated in Figure 2, we leverage two source-specific feature encoding networks, constructing a classical dual-stream architecture, to extract hidden representations from the trajectory map T and the aerial image I respectively. Compared with existing studies [41] of aggregating/concatenating multi-source features as a joint input, our dual-stream architecture can better extract the individual information from different sources and avoid mutual interference. In particular, the network is designed based on UNet, which is a universally useful framework for the object segmentation problem in the filed of computer vision.

In particular, each encoder aims to capture the shallow texture and location information of images. To achieve this goal, we implement each encoder with a convolution neural network layer (denoted as Con_i^n and Con_t^n for E_I and E_T respectively, where $n \in \{1, 2, 3, 4\}$), which would down-sample raw images into low-dimensional feature maps. In contrast, each decoder aims to capture deep semantic information of images. In addition, we need to recover the shape of feature maps from low dimensions, so we implement each decoder with a transposed convolution [14] neural network layer (denoted as $TCon_i^n$ and $TCon_t^n$ for E_I and E_T respectively, where $n \in \{5, 6, 7, 8\}$), which would up-sample low-dimensional feature maps. To fuse shallow location information and deep semantic information, we take a skip connection to merge each encoder layer with a decoder layer, where the two layers' outputs have the same shape. In summary, the above procedure (i.e., Formula 4) can be elaborated as follows:

$$\begin{aligned} e_i^n, e_t^n &= Con_i^n(e_i^{n-1}), Con_t^n(e_t^{n-1}), & n \in \{1, 2, 3, 4\} \\ e_i^n &= TCon_i^n(e_i^{n-1}) + e_i^{9-n}, & n \in \{5, 6, 7, 8\} \\ e_t^n &= TCon_t^n(e_t^{n-1}) + e_t^{9-n}, & n \in \{5, 6, 7, 8\} \end{aligned} \tag{4}$$

where e_i^0 and e_t^0 represent I and T respectively.

4.2 Pixel-Attention Fusion Module

There is a complementary correlation between the aerial image I and the trajectory map T. For example, roads may be occluded by trees in the aerial image while there may exist a lack of trajectories in sparsely populated regions, which leads to sparse roads in the trajectory map. Hence, it is significant to fuse multi-source data by making full use of the relationship, which would reduce the inherent defect of single source. However, it remains insufficient to simply aggregate multi-source features by concatenating them, which cannot capture fine-grained and dynamic complementarity of features. Therefore, we propose the pixel-attention fusion module F_{PAFM}, which uses the attention-based mechanism [27, 38, 44] to

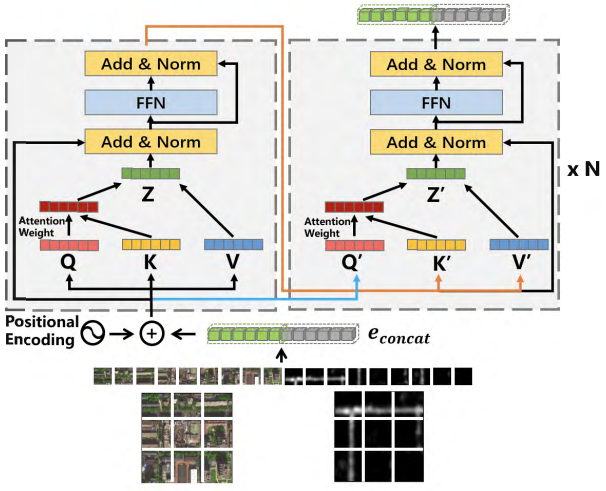


Figure 3: Elaboration of the Module F_{PAFM}.

cross multi-source features. In particular, each feature point in a source can interact with all feature points in another source, and the interaction intensity is represented by the learned attention weight, hence making it fine-grained and dynamic to fuse these features. In addition, given a feature in the current source, the fused features can be regarded as the supplement of other sources to the feature.

In this paper, we leverage 8 F_{PAFM} modules to fuse 8 pairs of e_i^n and e_t^n ($n \in \{1, 2, \cdots, 8\}$) respectively, which are generated by 8 layers in the source-specific feature encoding networks E_I and E_T. Details are illustrated in Figure 3. At first, we assume that the feature maps generated by the jth layer are $e_i^j \in \mathbb{R}^{h_j \times w_j \times c_j}$ and $e_t^j \in \mathbb{R}^{h_j \times w_j \times c_j}$ in E_I and E_T respectively, where h_j, w_j, and c_j respectively represent the height, width and number of channels of the feature maps. Next, due to the constraint of the attention structure, by only taking a sequence (e.g., $e \in \mathbb{R}^{N \times d}$) as input, where N is the number of tokens in the sequence and d is the embedding dimension, we reshape e_i^j and e_t^j into sequences $\hat{e}_i^j \in \mathbb{R}^{h_j * w_j \times c_j}$ and $\hat{e}_t^j \in \mathbb{R}^{h_j * w_j \times c_j}$ to meet the input requirement. Afterwards, we concatenate the reshaped features of different sources to form a unit feature sequence. In addition, in order to model each token's position in the sequence, we additionally learn the positional encoding for each order j, which is denoted as $Pos(j)$. Hence, the feature sequence e_{con}^j is computed as follows:

$$e_{con}^j = concat(\hat{e}_i^j,\ \hat{e}_t^j) + Pos(j) \tag{5}$$

In particular, we design two attention stages to capture the complementary correlation between two sources. In the first stage, we aim to generate a global encoding by merging multi-source information with the self-attention mechanism, and in the second stage, we aim to extract the supplementary information from the global encoding for each source with the cross-attention mechanism. Next, we will describe the two stages in details. For clarity of notations, we omit the superscript j in the following discussion.

Stage 1: The self-attention module is implemented with a multi-head attention framework, where the number of head is N. In each head m, we first perform linear transformation on e_{con} by the learnable weight matrices W_{q_m}, W_{k_m}, W_{v_m} to obtain the desired Q_m, K_m, $V_m \in \mathbb{R}^{2*h*w \times d_m}$:

$$Q_m, K_m, V_m = W_{q_m} \cdot e_{con}, W_{k_m} \cdot e_{con}, W_{v_m} \cdot e_{con} \tag{6}$$

Next, we leverage the query Q_m and the key K_m to calculate the attention score with the Softmax function as follows:

$$A_m = Softmax(\frac{Q_m K_m^T}{\sqrt{d_m}}) \tag{7}$$

where $A_m \in \mathbb{R}^{2*h*w \times 2*h*w}$ denotes the attention weight matrix. Next, the attention matrix A_m is multiplied by V_m to obtain the output Z_m of the m-th head:

$$Z_m = A_m \cdot V_m \tag{8}$$

The outputs of all heads are concatenated together and multiplied by the output weight matrix W_o to obtain the final output Z:

$$Z = concat(Z_1, ..., Z_N) \cdot W_o \tag{9}$$

At last, we perform Add&Norm operations, connect the input and output using residual connection, and perform a layer normalization [3] to get the first stage's output Z_o:

$$Z_o = LN(LN(e_{con} + Z) + MLP(Z)) \tag{10}$$

where each feature in Z_o fully interacts with other features and contains global semantic information.

Stage 2: The cross-attention module also includes N heads, and it aims to find the supplementary information from Z_o according to the given query (i.e., the initial embedding of each modal feature). Therefore, in each head m, we adopt Z_o and e_{con} as the inputs to obtain the corresponding Q'_m, K'_m and V'_m:

$$Q'_m, K'_m, V'_m = W'_{q_m} \cdot e_{con}, W'_{k_m} \cdot Z_o, W'_{v_m} \cdot Z_o \tag{11}$$

The remaining is similar to the self-attention stage, including the attention matrix A'_m, the output Z'_m, the concatenated output Z', and the final output Z'_o, which are calucated as follows:

$$\begin{aligned} A'_m &= Softmax(\frac{Q'_m {K'_m}^T}{\sqrt{d'_m}}) \\ Z'_i &= A'_i \cdot V'_i \\ Z' &= concat(Z'_1, ..., Z'_N) \cdot W'_o \\ Z'_o &= LN(LN(Z_o + Z') + MLP(Z')) \end{aligned} \tag{12}$$

At last, the output $Z'_o \in \mathbb{R}^{2*h*w \times c}$ would be split into two parts $Z'_o[1] \in \mathbb{R}^{h*w \times c}$ and $Z'_o[2] \in \mathbb{R}^{h*w \times c}$, which respectively represent the supplementary information for two sources. Hence, we reshape and add them to corresponding feature maps e_i and e_t. That is, according to Formula 4, we can recalculate $\{e_i^1, e_i^2, \cdots, e_i^8\}$ and $\{e_t^1, e_t^2, \cdots, e_t^8\}$ as follows:

$$\begin{aligned} e_i^n &= Con_i^n(e_i^{n-1}) + rsh(Z'^n_o[1]), && n \in \{1, \cdots, 4\} \\ e_t^n &= Con_t^n(e_t^{n-1}) + rsh(Z'^n_o[2]), && n \in \{1, \cdots, 4\} \\ e_i^n &= TCon_i^n(e_i^{n-1}) + e_i^{9-n} + rsh(Z'^n_o[1]), && n \in \{5, \cdots, 8\} \\ e_t^n &= TCon_t^n(e_t^{n-1}) + e_t^{9-n} + rsh(Z'^n_o[2]), && n \in \{5, \cdots, 8\} \end{aligned} \tag{13}$$

where $rsh(\cdot)$ denotes the reshape operator.

4.3 Edge Prediction Branch

In order to strengthen the segmentation performance of our model on road edges, we design the sub-task of road edge prediction inspired by [24] to maintain consistency and smoothness on segmentation results. Specifically, we propose an edge segmentation module F_{edge} to handle the task, where F_{edge} takes as input shallow

feature maps e_i^2, e_i^3, e_i^4 in E_I. The reason of using e_i^2, e_i^3, e_i^4 has been described in Sec 3, so we do not repeat it here. In addition, considering that these feature maps have different shapes, we respectively apply three convolution blocks $\hat{Con}(\cdot)$ to align their shapes, which can be formulated as follows:

$$e_i'^n = \hat{Con}_i^n(e_i^n),\ n \in \{2,3,4\} \tag{14}$$

where $e_i'^n \in \mathbb{R}^{h' \times w' \times c'_n}$ represents the corresponding aligned output of e_i^n. Afterwards, we concatenate three aligned outputs and leverage an transpose convolution block $\hat{TCon}(\cdot)$ to generate the predicted result, which is formulated as follows:

$$M_{edge} = \hat{TCon}(concat(e_i'^2, e_i'^3, e_i'^4)) \tag{15}$$

where $M_{edge} \in \mathbb{R}^{H \times W}$ represents the predicted result. Notably, we obtain the ground truth M_{gt_edge} by directly processing the ground truth M_{gt} in the main task. Specifically, we first perform morphological operations, such as erosion, on M_{gt} to obtain a thinned road mask, and then generate M_{gt_edge} by subtracting the mask from M_{gt}, which is formulated as follows:

$$M_{gt\_edge} = M_{gt} - Er(M_{gt}) \tag{16}$$

where $Er(\cdot)$ represents the erosion operator.

5 MODEL LEARNING

In this section, we first discuss the pixel-aware contrastive learning to enhance segmentation results in Section 5.1, then introduce all training losses in Section 5.2, and finally propose a transfer learning framework to better train **DICN** in Section 5.3.

5.1 Pixel-Aware Contrastive Learning

In some aerial images, we observe that some background pixels (e.g., gray roof) are similar to road pixels. Although we add constraints to the road edge using edge prediction branch, the model still suffers from ambiguity around the road edge area. In addition, each pixel's classification (i.e., road or not) is conducted separately in the module F_{seg}, which may lead to the lack of integrity and consistency in the prediction result (i.e., the extracted road is interrupted). Therefore, we propose a pixel-aware contrastive learning to discriminate foreground/positive and background/negative, and hence we can implicitly model the correlation between pixel labels to enhance the extraction performance. In particular, this procedure includes two steps: building positive/negative and computing the contrastive loss.

Building positive/negative samples. Given an image, we define all road pixels as positive samples, and others as negative samples. However, the number of pixel samples is too large, and there exist uneven distribution between positive and negative. Hence, it is unrealistic to directly use all samples for contrastive learning. To address this issue, we design a sampling approach to guide model training, which samples representative hard samples. Therefore, we take the road and background pixels adjacent to road edges as hard positive and negative samples respectively, which can be done by morphological operations on M_{gt}. In particular, the former leverages the erosion operator and the latter uses the dilation operator, which can be formulated as follows:

$$Pos = [(M_{gt} - Er(M_{gt})) > 0] \tag{17}$$

$$Neg = [(Di(M_{gt}) - M_{gt}) > 0] \tag{18}$$

where $Er(\cdot)$ and $Di(\cdot)$ represent erosion and dilation respectively; $[condition]$ helps extract pixels satisfied the given $condition$.

Computing loss. As shown in Figure 2, we extract pixel embedding $i \in \mathbb{R}^D$ from the feature map $f \in \mathbb{R}^{H \times W \times D}$ in the segmentation head F_{seg}. Later, we take InfoNCE [36] [47] as the loss function to improve intra-class compactness and inter-class separation, which makes semantically similar features as close as possible, and pushes different features farther away. To this end, given any road pixel embedding i, the contrastive loss can be formulated as follows:

$$\mathcal{L}_{NCE} = -\sum_{i^+} log \frac{exp(\frac{i \cdot i^+}{\tau})}{exp(\frac{i \cdot i^+}{\tau}) + \sum_{i^-} exp(\frac{i \cdot i^-}{\tau})} \tag{19}$$

where i^+ and i^- represent the pixel embedding from the positive sample set Pos and the negative sample set Neg, respectively. τ is the temperature parameter. Through our pixel-aware contrastive learning method, we can effectively alleviate the confusion of positive and negative samples, and suppress a large number of false positives generated by visually similar areas.

5.2 Overall Learning Objective

The overall learning objective is composed of the cross-entropy loss $\mathcal{L}_{ce}$, the dice loss $\mathcal{L}_{dice}$, the edge prediction loss $\mathcal{L}_{edge}$ and the contrastive loss $\mathcal{L}_{NCE}$ as follows:

$$\mathcal{L} = \mathcal{L}_{ce} + \mathcal{L}_{dice} + \lambda_1 \mathcal{L}_{edge} + \lambda_2 \mathcal{L}_{NCE} \tag{20}$$

where λ_1 and λ_2 are hyper-parameters to control the weights of these losses. At first, $\mathcal{L}_{ce}$ and $\mathcal{L}_{dice}$ represent cross-entropy loss and dice loss [33], which are used to optimize the classification ability of the model and focus on the mining of foreground regions. Specifically, given the predicted result M_{seg} and the target M_{gt}, $\mathcal{L}_{ce}$ and $\mathcal{L}_{dice}$ are defined as follows:

$$\mathcal{L}_{ce} = \sum_{y \in M_{seg}, \bar{y} \in M_{gt}} \bar{y} log(y) + (1 - \bar{y}) log(1 - y) \tag{21}$$

$$\mathcal{L}_{dice} = 1 - \frac{2|M_{seg} \cap M_{gt}|}{|M_{seg}| + |M_{gt}|} \tag{22}$$

Afterwards, $\mathcal{L}_{edge}$ also leverages the cross entropy loss to measure the difference between the edge prediction result M_{edge} and the ground truth M_{gt_edge}, which is denoted as follows.

$$\mathcal{L}_{edge} = \sum_{y \in M_{edge}, \bar{y} \in M_{gt\_edge}} \bar{y} log(y) + (1 - \bar{y}) log(1 - y) \tag{23}$$

Notably, since foreground and background pixels are severely unbalanced in the road extraction scene, in order to reduce the domination of the loss by the background region, we weight $\mathcal{L}_{ce}$ and $\mathcal{L}_{edge}$ according to the pixel ratio. At last, we jointly train the model by minimizing $\mathcal{L}$ in an end-to-end fashion.

5.3 Transfer Learning

Most existing studies [26, 28] initialize model parameters by pre-training on ImageNet. However, due to the difference between natural images in ImageNet and our used aerial images, such pre-training methods cannot generate optimal initial parameters. To address this issue, we propose a simple yet effective transfer learning method, which first extends the sample space by fusing data in the original space, then builds new tasks and pre-trains model parameters in extended sample spaces.

Extending sample spaces. As shown in Figure 4, given an input pair (I, T) in the original space, we fuse them into new images IT

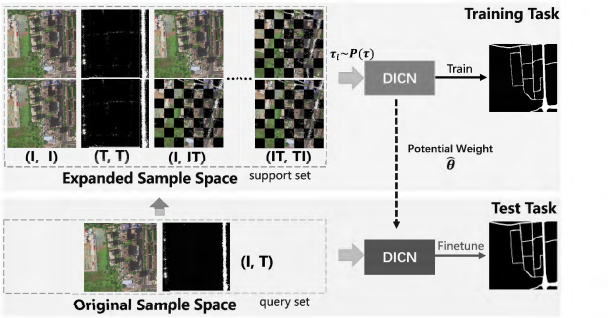


Figure 4: Transfer Learning for DICN

Algorithm 1 Transfer Learning for **DICN**

Input: initial parameter weight θ_0, original training inputs $X = (I, G)$, original training labels Y, transfer learning epochs S_1, fine-tuning epochs S_2, learning rate α
Output: optimal parameter weight $\hat{\theta}$

1: initialize $\hat{\theta}$ with θ_0
2: generate extended tasks $P(\tau)$ using X and Y
3: **for all** $i \leftarrow 1 \cdots S_1$ **do**
4: random sample a task $\tau_i = (X_{\tau_i}, Y_{\tau_i}) \sim P(\tau)$
5: generate the gradient ∇_θ with $\mathcal{L}$ on the task τ_i
6: update $\hat{\theta} = \hat{\theta} - \alpha * \nabla_\theta$
7: **end for**
8: **for all** $i \leftarrow 1 \cdots S_2$ **do**
9: generate the gradient ∇_θ with $\mathcal{L}$ on the task (X, Y)
10: update $\hat{\theta} = \hat{\theta} - \alpha * \nabla_\theta$
11: **end for**
12: **return** $\hat{\theta}$

and TI so as to explicitly capture the information crossover between different sources. Specifically, we divide I and T into two $N \times N$ grids, and then exchange them with each other at a fixed interval to generate fused images. For example, if we have $I = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$, $T = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$ and the interval is set to 1, the fused results would be denoted as $IT = \begin{bmatrix} 1 & 6 \\ 7 & 4 \end{bmatrix}$ and $TI = \begin{bmatrix} 5 & 2 \\ 3 & 8 \end{bmatrix}$. Actually, the explicitly fused data can be regarded as coming from new sources, so pre-training on them can drive **DICN** to learn how to better extract features from multi-sources. At this point, we can construct new sample space by combining sample pairs from the set $\{I, T, IT, TI\}$, where the combination number is 10 (i.e., (I, I), (I, T), (I, IT), (I, TI), (T, T), (T, IT), (IT, T), (IT, IT), (IT, TI), (TI, TI)). Notably, the pair (I, T) belongs to the original space, so the number of new extended spaces is 9. Accordingly, we can build 9 new road extraction tasks to provide data to pre-train **DICN**.

Pre-training & Fine-tuning. As shown in Algorithm 1, the whole learning process includes two steps: pre-training (lines 1-7) and fine-tuning (lines 8-11). In particular, the first step is to generate initial weight for the model parameter $\hat{\theta}$ based on extended tasks, and the second step is to train the model with the initial parameter for the main task.

Discussion. Our proposed transfer learning method is fundamentally different from the process of meta learning [15]. On the one hand, meta learning is to learn an initialization weight more suitable for new task from the perspective of task optimization; in contrast, we are to find the weight more suitable for the objective task from the perspective of searching sample spaces. On the other hand, meta learning cannot guarantee that the weight generated by the training task is optimal before completing the test task training; instead, it can only guarantee the relative optimization through gradient propagation. Our validation set is the same as the test task, so it provides a strong prior knowledge to ensure that the weight generated by the training task is currently optimal.

6 EXPERIMENTS

In this section, we perform a comprehensive comparison of our **DICN** with existing state-of-the-art approaches to the road extraction problem, and conduct ablation studies to evaluate each part's contribution in **DICN**.

6.1 Experimental Settings

Datasets. We evaluate our proposed **DICN** on two public multi-source (i.e., aerial image and GPS trajectory) road datasets, which are respectively called *BJRoad* [1] and *Porto* [2].

(1) *BJRoad* represents the road dataset on Beijing, containing 350 aerial images and 50 million GPS trajectory points. In particular, the image resolution is $50cm$ per pixel and each image includes $1,024 \times 1,024$ pixels, so each image covers a total area of over $100km^2$. In addition, we transform all GPS points into 350 trajectory maps according to Sec. 2.2. At last, we treat the pairing of an image and its corresponding trajectories as a singular sample, and then partition these samples into training, validation, and test sets, maintaining a ratio of 70%:10%:20% respectively.

(2) *Porto* represents the road dataset on Porto, and contains 6,048 aerial images and 1.71 million trajectories generated by 442 taxis spanning from 2013 to 2014. In particular, each image has a shape of 512×512. Similar to [28], we use the mean and standard variance of the five-fold experiment as the final result, and do not explicitly split the validation set.

Evaluation Metrics. To be consistent with existing work [28, 41], we adopt Intersection of Union (IoU) as the evaluation metric. IoU, defined as the ratio of intersection and union, is mainly used to measure the overlap between ground truth and segmentation result. For our road extraction task, it is used to evaluate how well the extracted road area fits the real road area. In particular, IoU includes two forms of Average IoU (AIoU) and Global IoU (GIoU). The former calculates IoU separately for each image and then takes an average of all IoUs, and the latter splices all images into one big image, and then calculates the IoU. Assuming that there are $IoU_1 = \frac{Int_1}{Uni_1}, \cdots, IoU_m = \frac{Int_m}{Uni_m}$, we have $AIoU = \frac{\sum IoU_i}{m}$ and $GIoU = \frac{\sum Int_i}{\sum Uni_i}$, where Int and Uni respectively means the interaction and the union. It is worth mentioning that the AIoU calculation method reported in previous work [41] (i.e. computing the AIoU for each batch and then taking an average of all batches) is not strictly correct, so we use our duplication results following the default setting of [28, 41, 52] for a fair comparison.

Baseline Methods. We compare our models with six methods:

- **DeepLab [11]**: This method adopts the Atrous Spatial Pyramid Pooling structure and the dilated convolution operator for extracting semantic features.
- **UNet [39]**: This method adopts an encoder-decoder structure, and fuses detailed features and semantic features through skip connections.
- **D-LinkNet [65]**: This method leverages the model Linknet [9] as the backbone, and additionally adds dilated convolution layers in the center part.
- **LCGD [41]**: This method considers both trajectory data and aerial image data, and uses 1D transpose convolution to further enrich feature extraction.
- **DeepDualMapper [52]**: This method designs a gated module to fuse multi-source features, where the gated module is implemented by the weighted adding operator.
- **CMMPNet [28]**: This method uses the feature extractor to extract the features of different modalities, and refines the features through the fusion module to achieve feature propagation across modalities.

Notably, **DeepLab**, **UNet**, **D-LinkNet** and **LCGD** belong to the single-stream network, which cannot seperately take as input T and I. To make the comparison fair, we directly concatenate T and I into a unit feature for these methods.

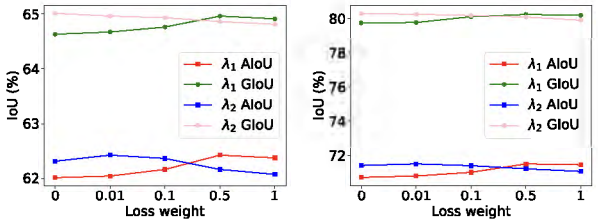
Implementation Details. We choose ResNet-34 [16] as the backbone of source-specific feature encoding network unless otherwise specified. In order to ensure the training quality of the fusion module F_{PAFM}, we leverage some data augmentation techniques, including horizontal and vertical flipping, rotation scaling and random erasing under a certain probability. We use the AdamW optimizer [29] to learn parameters, and set the weight decay as 0.01. The initial learning rate is set as $2e^{-4}$ and obeys the cosine schedule decay. For the joint loss function, according to the experimental results, we set the optimal weight of each term as $\lambda_1 = 0.5$ and $\lambda_2 = 0.01$. Similarly, for the number of transfer learning epochs S_1, we set it with the optimal value 25.

6.2 Settings of Important Hyper-parameters

We consider the following hyper-parameters: (1) two loss weights λ_1 and λ_2, which respectively correspond to the influence of two auxiliary tasks on the mask task; (2) the number of transfer learning epochs S_1.

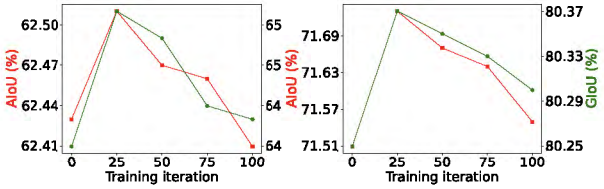
(1) **Loss Weights of Auxiliary Tasks.** We performed systematic experiments to optimize two hyper-parameters, λ_1 and λ_2, which correspond to the loss weights of two auxiliary tasks. Specifically, we divided the range from 0 to 1 into fixed intervals, choosing five values (0, 0.01, 0.1, 0.5, and 1) to span the entire range of possible weights. As depicted in Figure 5, the optimal values for λ_1 and λ_2 were found to be 0.5 and 0.01, respectively.

(2) **Transfer Learning Epochs.** The effectiveness of transfer learning is further demonstrated through a series of experiments as shown in Figure 6. As the number of training iterations increases, the model performance typically exhibits an upward trend, which signifies the consistent enhancement brought by transfer learning. Notably, upon reaching a special number of training rounds (e.g., $S_1 = 25$ for *BJRoad*), the model attains its best performance. Further training beyond this point, however, leads to a minor decrease in accuracy due to overfitting in the expanded sample space.



(a) *BJRoad* (b) *Porto*

Figure 5: Effect of λ_1 and λ_2 on BJRoad and Porto.



(a) *BJRoad* (b) *Porto*

Figure 6: Effect of Transfer learning Epochs.

Table 1: Accuracy comparison with baseline methods.

Method	*BJRoad*		*Porto*	
	AIoU (%)	GIoU (%)	AIoU (%)	GIoU (%)
DeepLab	59.83	62.87	64.42 ± 0.55	68.90 ± 0.32
UNet	58.27	61.77	68.39 ± 0.30	72.73 ± 0.28
D-LinkNet	59.51	62.51	67.53 ± 0.31	71.70 ± 0.14
LCGD	59.70	62.74	67.48 ± 0.29	71.74 ± 0.14
DeepDualMapper	59.55	62.91	65.41 ± 0.48	69.85 ± 0.15
CMMPNet	60.19	63.59	68.01 ± 0.31	72.32 ± 0.22
DICN	62.51 **(+2.32%)**	65.06 **(+1.47%)**	72.47 ± 0.87 **(+4.46%)**	80.66 ± 0.57 **(+8.34%)**
DICN + TTA	62.84 **(+2.65%)**	65.26 **(+1.67%)**	74.29 ± 0.79 **(+6.28%)**	82.41 ± 0.46 **(+10.09%)**

6.3 Accuracy Comparison

In this section, we compare **DICN** with six baseline methods by evaluating the metrics on *BJRoad* and *Porto*. In addition, we apply the TTA (Test Time Augmentation) strategy to improve the performance. Specifically, given a test sample, we first generate new samples with some augmentation operators, such as flipping or rotating, and then use the trained model to infer segmentation results for new samples. Afterwards, we leverage the voting strategy to fuse these segmentation results as the final result. Table 1 reports the evaluation results of all methods, and we have the following observations:

(1) These methods with the dual-stream architecture, including our method, outperform others on the dataset *BJRoad*. For example, the worst method is the single-stream method **UNet**, which corresponds to 58.27% AIoU and 61.77% GIoU. The reason is that these

Table 2: Effects of core modules on Test set of *BJRoad*.

Group	E_I, E_T	F_{PAFM}	F_{edge}	AIoU (%)	GIoU (%)
$\mathbb{A}$	√			60.27	63.30
$\mathbb{B}$	√	ConvFuse		60.90	63.74
$\mathbb{C}$	√	√		62.06	64.71
$\mathbb{D}$	√	ConvFuse	√	62.08	64.62
$\mathbb{E}$	√	√	√	62.32	65.01

single-stream methods have no specific fusion approaches to effectively capture the complementary correlation between different sources.

(2) Existing dual-stream methods except our method have no obvious advantages on *Porto* when comparing with single-stream methods. Particularly, **DeepDualMapper** has the worst performance. The reason is two-fold: the data quality in *Porto* is better than *BJRoad*, and thereby single-stream methods can get some passable results; the fusion modules in previous dual-stream methods cannot capture the correlation between different sources, and are ineffective to solve the road extraction problem.

(3) The performance of all method on *Porto* is better than that on *BJRoad*. That is because *BJRoad* contains a large number of noise signals and *Porto* has higher annotation quality and a larger amount of data.

(4) Our method **DICN** outperforms all existing methods on both *BJRoad* and *Porto*, where **DICN** surpasses all previous methods with solid margins. For example, the best existing method is **CMMPNet**, whose AIoU and GIoU are 60.19% and 63.59% on *BJRoad*, but **DICN** obtains the highest AIoU of 62.51% and GIoU of 65.06%. By further using the simple TTA strategy, we can achieve up to 2.65% of AIoU and 1.67% of GIoU improvement over **CMMPNet**, setting a new state-of-the-art.

(5) Our method *DICN* has greater improvement on *Porto* than *BJRoad*, demonstrating that *DICN* is more robust than others. In addition, the TTA strategy plays a significant role in promoting the performance of **DICN**.

(6) The value of GIoU is greater than that of AIoU for all methods on both *BJRoad* and *Porto*. Hence, we have the inequality $\frac{\sum Int_i}{\sum Uni_i} >= \frac{\sum IoU_i}{m}$ according to the definitions of GIoU and AIoU, and then infer that $\frac{\frac{Uni_1}{\sum Int_i/m}+\cdots+\frac{Uni_m}{\sum Int_i/m}}{m} <= \frac{m}{\frac{1}{Uni_1/Int_1}+\cdots+\frac{1}{Uni_m/Int_m}} <= \frac{\frac{Uni_1}{Int_1}+\cdots+\frac{Uni_m}{Int_m}}{m}$ based on the inequality of arithmetic and geometric means, which means the average of $\frac{Uni_i}{\sum Int_i/m}$ is smaller than the average of $\frac{Uni_i}{Int_i}$.

6.4 Ablation Studies

We conduct extensive ablation studies on *BJRoad* to evaluate the effectiveness of key designs in **DICN**.

6.4.1 Effect of core modules. As illustrated in Figure 2, the core modules of **DICN** include the source-specific feature encoding networks (E_I, E_T), the edge prediction branch F_{edge}, and the fusion module F_{PAFM}. In particular, we replace **DICN** with five variations, namely Group $\mathbb{ABCDE}$, to evaluate the effectiveness of different modules in **DICN**. As shown in Table 2, we only use E_T and E_I to individually extract features from different sources, and fuse these features by averaging last features maps in group $\mathbb{A}$. As a result, $\mathbb{A}$ obtains 60.27% AIoU and 63.30% GIoU, which surpass the best baseline **CMMPNet** by 0.08% in AIoU, fully demonstrating the excellent performance of E_I and E_T. In groups $\mathbb{B}$ and $\mathbb{C}$, we respectively apply the convolution network and the attention mechanism to fuse multi-source feature maps in the module F_{PAFM}. The result shows that 60.9% of AIoU and 63.74% of GIoU for $\mathbb{B}$, and 62.06% of AIoU and 64.71% of GIoU for $\mathbb{C}$, indicating that the attention mechanism is significant for better fusing multi-source feature maps. In addition, comparing $\mathbb{A}$ and $\mathbb{C}$, it's proved that F_{PAFM}, improving AIoU by 1.79% and GIoU by 1.41%, can effectively complete the feature crossover between different sources. At last, groups $\mathbb{D}$ and $\mathbb{E}$ append the module F_{edge} based on $\mathbb{B}$ and $\mathbb{C}$, respectively. By comparing $\mathbb{B}$ and $\mathbb{D}$, or $\mathbb{C}$ and $\mathbb{E}$, we can conclude that the edge prediction branch can stably improve the model's effect, up to 62.32% AIOU and 65.01% GIOU.

Moreover, to further verify the fusion effect of the module F_{PAFM}, we present the feature maps of **CMMPNet** and **DICN** in Figure 7. It can be seen from (b)-(c) and (g)-(h) that **DICN** pays more attention to the feature extraction of specific modalities than **CMMPNet**, fully excavating the unique information of each modal and the information of the other required. (d) and (i) represent the feature maps generated by the final fusion of image features and trajectory features. It can be found that (i) uses the information of the two modalities to outline the features of the road area more accurately and comprehensively, and is not affected by other noise disturbance. The red boxes in (e) and (j) represent railways, which **CMMPNet** cannot distinguish effectively, while **DICN** successfully suppresses false positives. Although **DICN** still predicts the road in the area of the yellow box in (j), we find that the area does have a road by observing (a), and the trajectory map (f) also contains GPS information here, so we think this road segment is omitted from ground truth, which indicates that **DICN** has strong generalization ability.

6.4.2 Effect of Trajectory Map Generation. To assess the map generation task, we compare the performance across various models when given trajectory maps of differing qualities. Intuitively, with more trajectories available, the quality of the generated trajectory map improves. To substantiate this, we adjust the ratio of trajectories used for trajectory map generation for each aerial image. As shown in Figure 8, the left and right figures plot the change of AIoU and GIoU with the percentage of used trajectory information, where 0% means that no trajectory information is used in the training process, and 100% means that all trajectory information is utilized. In particular, we have the following observations:

(1) The more trajectory information used, the better all methods' performance. Therefore, we can conclude that the trajectory information is significant for all methods.

(2) **DICN** achieves 60.07% AIoU and 63.27% GIoU without applying any trajectory information, which is much higher than other methods, so **DICN** is more robust than others.

(3) **DeepDualMapper** and **LCGD** have similar performance, and they cannot sufficiently leverage the trajectory information. When the percentage is greater than 25%, the performance lift of both *AIoU* and *GIoU* is very marginal, because their fusion methods are too simple to be effective.

(4) The effect of **CMMPNet** gradually increases with the introduction of trajectory information, indicating that **CMMPNet** relies

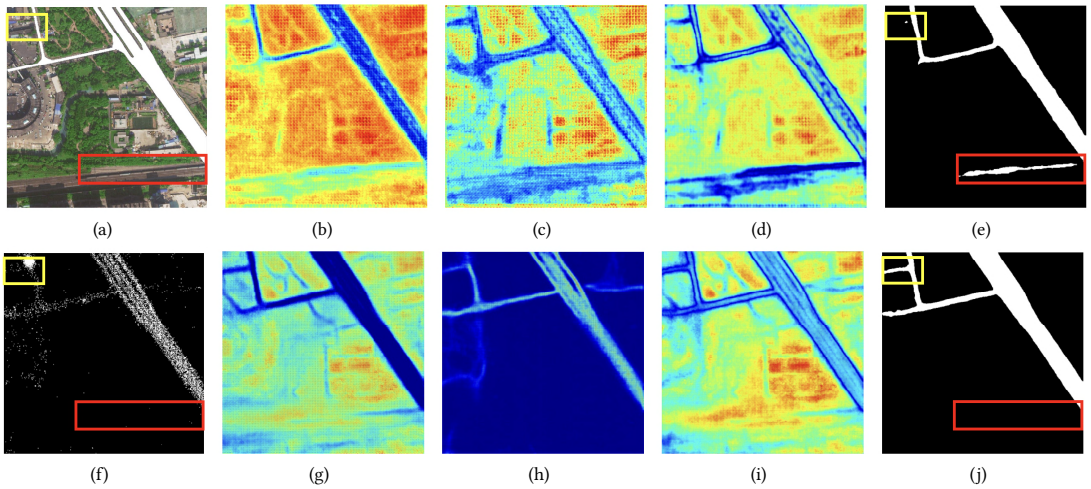
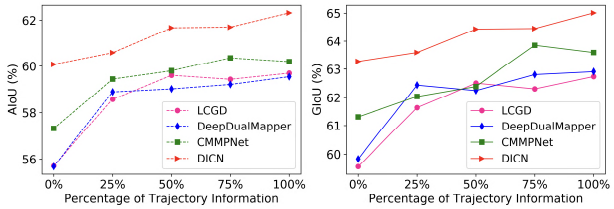


Figure 7: Visualization of Feature Maps for CMMPNet and DICN. Firstly, (a) is the aerial image with ground truth and (f) is its corresponding trajectory map. Secondly, (b)-(d), (g)-(i) are the fused feature maps generated by CMMPNet and DICN, respectively. Thirdly, (e) and (j) are the prediction results of CMMPNet and DICN, respectively.



(a) Evaluating AIoU (%) (b) Evaluating GIoU (%)

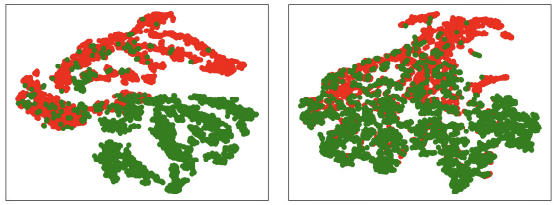
Figure 8: Effect of Trajectory Information on *BJRoad*

Table 3: Effects of Learning strategies for DICN on *BJRoad*.

$\mathcal{L}_{ce}$	$\mathcal{L}_{dice}$	$\mathcal{L}_{edge}$	$\mathcal{L}_{NCE}$	Transfer Learning	AIoU	GIoU
√					61.92	64.59
√	√				62.06	64.71
√	√	√			62.32	65.01
√	√	√	√		62.43	64.96
√	√	√		√	62.39	65.03
√	√	√	√	√	62.51	65.06

more on trajectory information in its decision, and its performance is greatly impacted when there is insufficient trajectory data.

6.4.3 Effect of Different Learning Strategies. To explore how the learning strategies affect the performance, we conduct ablation studies on various learning method in Table 3. We employ the cross-entropy loss $\mathcal{L}_{ce}$ as basic loss function, and add the dice loss $\mathcal{L}_{dice}$, edge prediction loss $\mathcal{L}_{edge}$ and contrastive loss $\mathcal{L}_{NCE}$ to it. Only using the basic loss functions $\mathcal{L}_{ce}$ and dice loss $\mathcal{L}_{dice}$ can obtain 62.06% AIoU and 64.71% GIoU, and further equipping the edge prediction loss $\mathcal{L}_{edge}$ can increase AIoU by 0.26% and GIoU by 0.3% respectively. The Pixel-Aware Contrastive Learning method $\mathcal{L}_{NCE}$ strengthens the ability of model to distinguish foreground and background pixels, which increases AIoU from 62.32% to 62.43%, with a slight drop in GIoU. We argue that the decline of GIoU is mainly due to the noise contained in the ground truth of *BJRoad*, and its overall discriminative ability is still improved steadily, as shown in Figure 7. After applying the Transfer Learning method,



(a) **LCGD** (b) **DeepDualMapper**

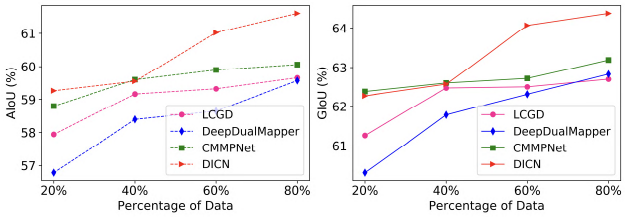
(c) **CMMPNet** (d) **DICN**

Figure 9: Visualization of t-SNE Embeddings learned with Different Approaches on *Porto*.

we can get consistent improvement in AIoU and GIoU, indicating that the Transfer Learning method can better initialize parameters, which is more suitable for current multi-task training. By adopting in both Pixel-Aware Contrastive Learning and Transfer Learning strategies, we obtain a noticeable improvement, and our **DICN** can reach up to 62.51% AIoU and 65.06% GIoU respectively. In conclusion, integrating the above learning strategies can effectively facilitate model learning on multi-tasks.

6.5 Evaluating the Discrimination Ability

To analyze the discriminative effect of different methods on foreground and background pixels, we use t-SNE [43] to visualize the embeddings' distribution of prediction results on *Porto*. As shown in Figure 9, the red and green points respectively denote embeddings of foreground and background pixels, where foreground pixels means the pixels on roads. Specifically, we extract the fused feature maps from the last layer as the embeddings for different dual-stream methods, and use t-SNE to reduce their dimensions



(a) Evaluating AIoU (%) (b) Evaluating GIoU (%)

Figure 10: Effect of Scalability on *BJRoad*.

to a two-dimensional space. Notably, we explicitly keep the same number of positive and negative samples. To be discriminative, the embeddings in the same class should be close to each other, and there should be a clear boundary between different classes. According to Figure 9, we have the following observations:
(1) For **DeepDualMapper**, the embeddings of foreground and background pixels overlap heavily, and there is no clear boundary. Hence, the discriminative ability of **DeepDualMapper** is weak, which makes it easy to confuse foreground and background pixels.
(2) **LCGD** is better than **DeepDualMapper**, and there is a relatively clear boundary that can isolate the embeddings in two classes. However, a large number of background pixels are still misjudged as foreground (i.e., green points appearing around red points). The reason is due to the fact that **LCGD** directly concatenates the inputs of different sources, resulting in the model not making full use of the auxiliary information between sources to suppress false positives.
(3) Although the foreground pixels of **CMMPNet** are relatively compact and form a clear classification cluster, the inter-class cohesion of background pixels is still poor. In particular, the overall embedding distribution of background pixels is scattered.
(4) **DICN** makes a very obvious decision boundary for pixel embeddings in different classes. In addition, either foreground pixels or background pixels form a close cluster. These demonstrates that **DICN** can generate discriminative features and thus yield competitive results.

6.6 Scalability Comparison

We evaluate the scalability of all methods by varying the size of training data. In particular, we sample 20%, 40%, 60% and 80% of the data from training set, and collect the associated AIoU and GIoU over the test data. As shown in Figure 10, we have the following observations:
(1) All methods perform better if more training data is used. The reason is that more data indicates more situations, making the model learn better.
(2) When the amount of data is scarce (e.g., using 20% training data), **DeepDualMapper** and **LCGD** have worse performance than others. The reason is that fusion modules in both **DeepDualMapper** and **LCGD** are direct, which cannot capture sufficient correlations among different features.
(3) With the sampling rate of training data increased, the gap between our **DICN** and other methods becomes larger. Hence, our **DICN** is more scalable than others, indicating that **DICN** can make full use of training data.

Table 4: Efficiency of Different Models.

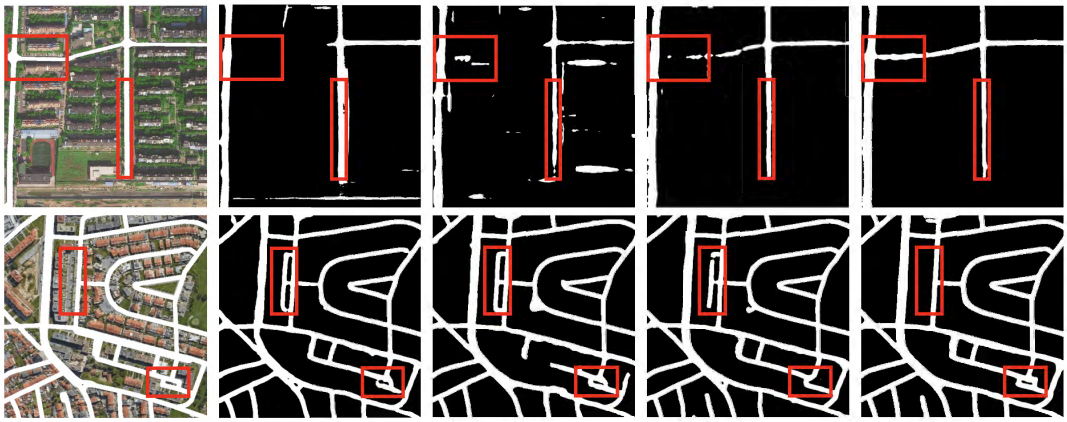
	LCGD	DeepDualMapper	CMMPNet	DICN
Parameters(MByte)	31.22	11.20	84.99	57.11
FPS(image/s)	91.97	55.01	26.94	27.69
Training Cost(s/iteration)	1.04	1.30	1.44	1.53

6.7 Efficiency Comparison

We use parameters, frame per second (FPS) and training cost for efficiency evaluation. In particular, parameters represent the size of different models, and it is used to evaluate memory efficiency. FPS is used to evaluate computing efficiency, which indicates the number of images that the model can process per second, and its value is positively correlated with inference speed. The training cost represents the time spent on training each iteration. The results are reported in Table 4. We observe the following:
(1) The FPS of **LCGD** is much higher than other methods, which shows that the inference speed of single-stream architecture is significantly better than that of dual-stream architecture. The speed bottleneck of the latter lies in the need to use different sub-networks to extract the information of the corresponding modality, but it can achieve more sufficient feature interaction.
(2) Although **DeepDualMapper** has the least amount of parameters among all methods, its inference speed is still slower than **LCGD** due to its dual-stream architecture.
(3) **DICN** is similar to **CMMPNet** in terms of the FPS and the training cost, but with fewer parameters, it has achieved a better trade-off between accuracy and computational efficiency.

6.8 Case Study

To further evaluate the effectiveness of our proposed **DICN**, we sample some cases from *BJRoad* and *Porto*, and illustrate different approaches' extraction results in Figure 11. On the one hand, due to the lack of effective fusion for multi-source data, **LCGN** is easily affected by biased image information such as tree occlusion. In particular, there are a large number of vacancies on the extracted road, which demonstrate that the trajectory information is not fully utilized. On the other hand, although **DeepDualMapper** and **CMMPNet** design some fusion modules to capture the complementary correlation between the aerial image and the trajectory information, there are a large number of isolated fragments and false positives in their prediction results, which lack the overall consistency and integrity. At the same time, road edges in these extraction results are rough and easily disturbed by the background. In contrast, our method **DICN** obtains more accurate results, where the complementary correlation between different sources is captured by the attention mechanism, and the edge prediction sub-task makes the predicted road edge smooth and consistent. Moreover, compared with *BJRoad*, samples in *Porto* have better annotation quality. Therefore, even if the road network of *Porto* is more complex, the visualization effect is better for all methods. To this end, we use the red box to highlight some visually indistinguishable regions, where existing methods generate false positives in their extraction results, but **DICN** has a strong ability to distinguish between foreground and background pixels. Specifically, **DICN** can effectively recognize background regions similar to roads while maintaining a high recall.



(a) Image & Ground Truth (b) **LCGD** (c) **DeepDualMapper** (d) **CMMPNet** (e) **DICN**

Figure 11: The Qualitative Results of Different Methods for Some Samples in *BJRoad* (top row) and *Porto* (bottom row).

7 RELATED WORK

7.1 Aerial Image-based Road Extraction

Numerous studies are based on aerial images for road extraction and we only introduce the methods with higher accuracy of neural networks instead of traditional methods [12, 31, 61], which use hand-crafted features to feed into shallow models with relatively limited performance. Methods using aerial images can be broadly divided into segmentation-based methods [8, 50, 63] and graph-based methods [5, 18, 21, 42]. Segmentation-based methods model road extraction as a semantic segmentation problem, and predict through pixel-wise classification. Their main disadvantage is that they are susceptible to interfere from image content and quality such as object occlusion, weather conditions, etc. Graph-based methods treat roads as a collection of edges and points, and directly extract the road network by predicting the relationship between vertices and edges. Although this type of method has high accuracy, its pipeline is relatively complex and relies on structured labeled data.

7.2 Trajectory-based Road Extraction

Many studies [51, 64] are based on GPS trajectories for road extraction, because trajectories intuitively reflect the driving path of vehicles. The work using trajectory data is mainly divided into point clustering-based methods [10, 17, 53] and Kernel Density Estimation (KDE)-based methods [6, 13, 53]. These clustering-based methods first use some clustering algorithms, such as k-means, to cluster discrete GPS points, and then further extract road segments through cluster centers to generate road networks. The unavoidable large amount of noise in GPS data is the main bottleneck, which would lead to wrong road segments and affect the overall performance. The authors in [40] propose DeepMG to extract features from trajectories in both spatial view and transition view, and use neural network to infer road centerlines. The KDE-based methods apply KDE to transform data points into density maps, which are further processed into road segments. Although the latter is more robust to noise, it is still less effective. Similar to the image modality, only using the trajectory modality will cause a lot of information loss, and there is not sufficient content in sparsely populated places to generate a complete trajectory map.

7.3 Multi-Source-based Road Extraction

Using only trajectory data or image data has limitations and drawbacks, so integrating features from multiple sources is optimal for road extraction. The mainstream approach is usually to use image data combined with other modal data [26, 30, 32, 38, 62], such as Lidar, SAR, and trajectory, to achieve modal fusion through the interaction between features. One idea is to concatenate the features of different modalities in the input stage through early fusion, so that the network can automatically adapt to the combined input, but it will cause interference related to the learning process of different features. The other is the late fusion method, which uses different feature extractors for different modalities and fuses in the middle or at the end of the network. Since the effect of late fusion is better, we also refer to this practice.

8 CONCLUSION

In this paper, we presented an effective approach, namely Dual Information Crossing Network (**DICN**) to solve the road extraction problem, which utilizes both aerial images and trajectories information. **DICN** seamlessly incorporates the source-specific feature encoding network, the pixel-attention fusion module and the edge prediction branch module for the feature interaction between different modalities. We further boost its performance by introducing the pixel-aware contrastive learning and the transfer learning method. Extensive experiments on two public datasets have demonstrated the superiority of our approach. At last, we find that noises in the ground truth can significantly affect the model's performance, and we would like to study how to improve the model's robustness under noisy labels in the future work.

ACKNOWLEDGMENTS

Haitao Yuan is the corresponding author. This work is supported by NSF of China (62032003), ENN, and Singtel. Zhifeng Bao is supported in part by ARC DP220101434 and DP200102611.

REFERENCES

[1] 2023. BJRoad. https://github.com/suniique/Leveraging-Crowdsourced-GPS-Data-for-Road-Extraction-from-Aeria.

[2] 2023. Porto. https://github.com/algorithm-panda/.

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[4] Favyen Bastani, Songtao He, Sofiane Abbar, Mohammad Alizadeh, Hari Balakrishnan, Sanjay Chawla, Sam Madden, and David DeWitt. 2018. Roadtracer: Automatic extraction of road networks from aerial images. In *CVPR*. 4720–4728.

[5] Davide Belli and Thomas Kipf. 2019. Image-conditioned graph generation for road network extraction. *arXiv preprint arXiv:1910.14388* (2019).

[6] James Biagioni and Jakob Eriksson. 2012. Map inference in the face of noise and disparity. In *SIGSPATIAL GIS*. 79–88.

[7] Zdravko I Botev, Joseph F Grotowski, and Dirk P Kroese. 2010. Kernel density estimation via diffusion. *AoS* 38, 5 (2010), 2916–2957.

[8] Alexander Buslaev, Selim Seferbekov, Vladimir Iglovikov, and Alexey Shvets. 2018. Fully convolutional network for automatic road extraction from satellite imagery. In *CVPRW*. 207–210.

[9] Abhishek Chaurasia and Eugenio Culurciello. 2017. Linknet: Exploiting encoder representations for efficient semantic segmentation. In *VCIP*. IEEE, 1–4.

[10] Chen Chen, Cewu Lu, Qixing Huang, Qiang Yang, Dimitrios Gunopulos, and Leonidas Guibas. 2016. City-scale map creation and updating using GPS collections. In *SIGKDD*. 1465–1474.

[11] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. 2018. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*. 801–818.

[12] Guangliang Cheng, Ying Wang, Yongchao Gong, Feiyun Zhu, and Chunhong Pan. 2014. Urban road extraction via graph cuts based probability propagation. In *ICIP*. IEEE, 5072–5076.

[13] Jonathan J Davies, Alastair R Beresford, and Andy Hopper. 2006. Scalable, distributed, real-time map generation. *IEEE PERVAS COMPUT* 5, 4 (2006), 47–54.

[14] Vincent Dumoulin and Francesco Visin. 2018. A guide to convolution arithmetic for deep learning. *stat* 1050 (2018), 11.

[15] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*. PMLR, 1126–1135.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*. 770–778.

[17] Songtao He, Favyen Bastani, Sofiane Abbar, Mohammad Alizadeh, Hari Balakrishnan, Sanjay Chawla, and Sam Madden. 2018. RoadRunner: improving the precision of road network inference from GPS trajectories. In *SIGSPATIAL GIS*. 3–12.

[18] Songtao He, Favyen Bastani, Satvat Jagwani, Mohammad Alizadeh, Hari Balakrishnan, Sanjay Chawla, Mohamed M Elshrif, Samuel Madden, and Mohammad Amin Sadeghi. 2020. Sat2graph: Road graph extraction through graph-tensor encoding. In *ECCV*. Springer, 51–67.

[19] Abdeltawab M Hendawi, Jie Bao, and Mohamed F Mokbel. 2013. iRoad: a framework for scalable predictive query processing on road networks. *PVLDB* 6, 12 (2013), 1262–1265.

[20] Stefan Hinz and Albert Baumgartner. 2003. Automatic extraction of urban road networks from multi-view aerial imagery. *ISPRS* 58, 1-2 (2003), 83–98.

[21] Jilin Hu, Chenjuan Guo, Bin Yang, and Christian S Jensen. 2019. Stochastic weight completion for road networks using graph convolutional networks. In *ICDE*. IEEE, 1274–1285.

[22] Hoyoung Jeung, Man Lung Yiu, Xiaofang Zhou, and Christian S Jensen. 2010. Path prediction and predictive range querying in road network databases. *The VLDB Journal* 19 (2010), 585–602.

[23] Meng Lan, Yipeng Zhang, Lefei Zhang, and Bo Du. 2020. Global context based automatic road segmentation via dilated convolutional neural network. *INFORM SCIENCES* 535 (2020), 156–171.

[24] Peike Li, Yunqiu Xu, Yunchao Wei, and Yi Yang. 2020. Self-correction for human parsing. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 6 (2020), 3260–3271.

[25] Justin Liang, Namdar Homayounfar, Wei-Chiu Ma, Shenlong Wang, and Raquel Urtasun. 2019. Convolutional recurrent network for road boundary extraction. In *CVPR*. 9512–9521.

[26] Yinyi Lin, Luoma Wan, Hongsheng Zhang, Shan Wei, Peifeng Ma, Yu Li, and Zhuoyi Zhao. 2021. Leveraging optical and SAR data with a UU-Net for large-scale road extraction. *INT J APPL EARTH OBS* 103 (2021), 102498.

[27] Hao Liu, Jindong Han, Yanjie Fu, Jingbo Zhou, Xinjiang Lu, and Hui Xiong. 2020. Multi-modal transportation recommendation with unified route representation learning. *PVLDB* 14, 3 (2020), 342–350.

[28] Lingbo Liu, Zewei Yang, Guanbin Li, Kuo Wang, Tianshui Chen, and Liang Lin. 2022. Aerial images meet crowdsourced trajectories: a new approach to robust road extraction. *TNNLS* (2022).

[29] Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).

[30] Ali Akbar Matkan, Mohammad Hajeb, and Saeed Sadeghian. 2014. Road extraction from lidar data using support vector machine classification. *PHOTOGRAMM ENG REM S* 80, 5 (2014), 409–422.

[31] Juan B Mena. 2003. State of the art on automatic road extraction for GIS update: a novel classification. *PATTERN RECOGN LETT* 24, 16 (2003), 3037–3058.

[32] Zelang Miao, Wenzhong Shi, Alim Samat, Gianni Lisini, and Paolo Gamba. 2015. Information fusion for urban road extraction from VHR optical satellite images. *J-STARS* 9, 5 (2015), 1817–1829.

[33] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. 2016. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *3DV*. IEEE, 565–571.

[34] Kyriakos Mouratidis, Man Lung Yiu, Dimitris Papadias, and Nikos Mamoulis. 2006. Continuous nearest neighbor monitoring in road networks. *PVLDB* (2006).

[35] Mashaal Musleh, Sofiane Abbar, Rade Stanojevic, and Mohamed Mokbel. 2021. QARTA: an ML-based system for accurate map services. *PVLDB* 14, 11 (2021), 2273–2282.

[36] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).

[37] Dian Ouyang, Long Yuan, Lu Qin, Lijun Chang, Ying Zhang, and Xuemin Lin. 2020. Efficient shortest path index maintenance on dynamic road networks with theoretical guarantees. *PVLDB* 13, 5 (2020), 602–615.

[38] Aditya Prakash, Kashyap Chitta, and Andreas Geiger. 2021. Multi-modal fusion transformer for end-to-end autonomous driving. In *CVPR*. 7077–7087.

[39] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*. Springer, 234–241.

[40] Sijie Ruan, Cheng Long, Jie Bao, Chunyang Li, Zisheng Yu, Ruiyuan Li, Yuxuan Liang, Tianfu He, and Yu Zheng. 2020. Learning to generate maps from trajectories. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 890–897.

[41] Tao Sun, Zonglin Di, Pengyu Che, Chun Liu, and Yin Wang. 2019. Leveraging crowdsourced GPS data for road extraction from aerial imagery. In *CVPR*. 7509–7518.

[42] Yong-Qiang Tan, Shang-Hua Gao, Xuan-Yi Li, Ming-Ming Cheng, and Bo Ren. 2020. Vecroad: Point-based iterative graph exploration for road graphs extraction. In *CVPR*. 8910–8918.

[43] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *JMLR* 9, 11 (2008).

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *NIPS* 30 (2017).

[45] Ting Wang and Ling Liu. 2009. Privacy-aware mobile services over road networks. *PVLDB* 2, 1 (2009), 1042–1053.

[46] Weixing Wang, Nan Yang, Yi Zhang, Fengping Wang, Ting Cao, and Patrik Eklund. 2016. A review of road extraction from remote sensing images. *JTTE* 3, 3 (2016), 271–282.

[47] Wenguan Wang, Tianfei Zhou, Fisher Yu, Jifeng Dai, Ender Konukoglu, and Luc Van Gool. 2021. Exploring cross-image pixel contrast for semantic segmentation. In *ICCV*. 7303–7313.

[48] Yong Wang, Guoliang Li, Kaiyu Li, and Haitao Yuan. 2022. A Deep Generative Model for Trajectory Modeling and Utilization. *PVLDB* 16, 4 (2022), 973–985.

[49] Jan D Wegner, Javier A Montoya-Zegarra, and Konrad Schindler. 2013. A higher-order CRF model for road network extraction. In *CVPR*. 1698–1705.

[50] Yanan Wei, Zulin Wang, and Mai Xu. 2017. Road structure refined CNN for road extraction in aerial image. *GRSL* 14, 5 (2017), 709–713.

[51] Guojun Wu, Yichen Ding, Yanhua Li, Jie Bao, Yu Zheng, and Jun Luo. 2017. Mining spatio-temporal reachable regions over massive trajectory data. In *ICDE*. IEEE, 1283–1294.

[52] Hao Wu, Hanyuan Zhang, Xinyu Zhang, Weiwei Sun, Baihua Zheng, and Yuning Jiang. 2020. DeepDualMapper: A gated fusion network for automatic map extraction using aerial images and trajectories. In *AAAI*, Vol. 34. 1037–1045.

[53] Wei Yang, Tinghua Ai, and Wei Lu. 2018. A method for extracting road boundary information from crowdsourcing vehicle GPS trajectories. *Sensors* 18, 4 (2018), 1261.

[54] Xiaojing Yu, Xiang-Yang Li, Jing Zhao, Guobin Shen, Nikolaos M Freris, and Lan Zhang. 2022. ANTIGONE: Accurate Navigation Path Caching in Dynamic Road Networks leveraging Route APIs. In *INFOCOM*. IEEE, 1599–1608.

[55] Haitao Yuan and Guoliang Li. 2019. Distributed in-memory trajectory similarity search and join on road network. In *ICDE*. IEEE, 1262–1273.

[56] Haitao Yuan and Guoliang Li. 2021. A Survey of Traffic Prediction: from Spatio-Temporal Data to Intelligent Transportation. *Data Sci. Eng.* 6, 1 (2021), 63–85.

[57] Haitao Yuan, Guoliang Li, and Zhifeng Bao. 2022. Route Travel Time Estimation on A Road Network Revisited: Heterogeneity, Proximity, Periodicity and Dynamicity. *PVLDB* 16, 3 (2022), 393–405.

[58] Haitao Yuan, Guoliang Li, Zhifeng Bao, and Ling Feng. 2020. Effective Travel Time Estimation: When Historical Trajectories over Road Networks Matter. In *SIGMOD*, David Maier, Rachel Pottinger, AnHai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo (Eds.). 2135–2149.

[59] Haitao Yuan, Guoliang Li, Zhifeng Bao, and Ling Feng. 2021. An Effective Joint Prediction Model for Travel Demands and Traffic Flows. In *ICDE*. 348–359.
[60] Haitao Yuan, Guoliang Li, Ling Feng, Ji Sun, and Yue Han. 2020. Automatic View Generation with Deep Learning and Reinforcement Learning. In *ICDE*. 1501–1512.
[61] Jiangye Yuan, DeLiang Wang, Bo Wu, Lin Yan, and Rongxing Li. 2011. LEGION-based automatic road extraction from satellite imagery. *TGRS* 49, 11 (2011), 4528–4538.
[62] Qianqian Zhang, Qingling Kong, Chao Zhang, Shucheng You, Hai Wei, Ruizhi Sun, and Li Li. 2019. A new road extraction method using Sentinel-1 SAR images based on the deep fully convolutional neural network. *EuJRS* 52, 1 (2019), 572–582.
[63] Zhengxin Zhang, Qingjie Liu, and Yunhong Wang. 2018. Road extraction by deep residual u-net. *GRSL* 15, 5 (2018), 749–753.
[64] Lisheng Zhao, Jiali Mao, Min Pu, Guoping Liu, Cheqing Jin, Weining Qian, Aoying Zhou, Xiang Wen, Runbo Hu, and Hua Chai. 2020. Automatic calibration of road intersection topology using trajectories. In *ICDE*. IEEE, 1633–1644.
[65] Lichen Zhou, Chuang Zhang, and Ming Wu. 2018. D-LinkNet: LinkNet with pretrained encoder and dilated convolution for high resolution satellite imagery road extraction. In *CVPRW*. 182–186.
[66] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. 2019. Unet++: Redesigning skip connections to exploit multiscale features in image segmentation. *IEEE T MED IMAGING* 39, 6 (2019), 1856–1867.