Single-Shot End-to-end Road Graph Extraction

Gaetan Bahl[1,3] Mehdi Bahri[2] Florent Lafarge[3]
[1]IRT Saint Exupéry [2]Imperial College London [3]Université Côte d'Azur - Inria
{gaetan.bahl,florent.lafarge}@inria.fr, m.bahri@imperial.ac.uk

Abstract

Automatic road graph extraction from aerial and satellite images is a long-standing challenge. Existing algorithms are either based on pixel-level segmentation followed by vectorization, or on iterative graph construction using next move prediction. Both of these strategies suffer from severe drawbacks, in particular high computing resources and incomplete outputs. By contrast, we propose a method that directly infers the final road graph in a single pass. The key idea consists in combining a Fully Convolutional Network in charge of locating points of interest such as intersections, dead ends and turns, and a Graph Neural Network which predicts links between these points. Such a strategy is more efficient than iterative methods and allows us to streamline the training process by removing the need for generation of starting locations while keeping the training end-to-end. We evaluate our method against existing works on the popular RoadTracer dataset and achieve competitive results. We also benchmark the speed of our method and show that it outperforms existing approaches. Our method opens the possibility of in-flight processing on embedded devices for applications such as real-time road network monitoring and alerts for disaster response.

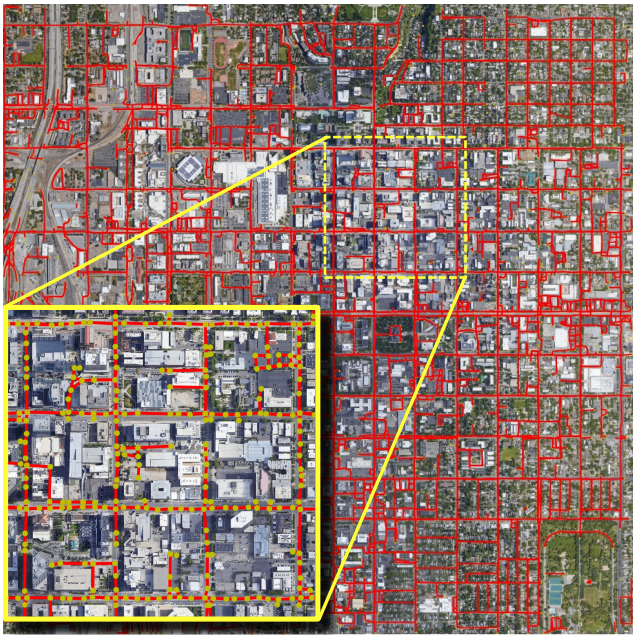

Figure 1. Result of our method on an image from the RoadTracer [3] test set. Our method regresses an entire graph without the need of pre or post-processing. This 8192x8192 pixel image was processed in 22 seconds on a single GPU, which is up to **91 times faster** than previous approaches. The extracted graphs have a low complexity and retain good accuracy compared to existing works.

1. Introduction

Vector-based maps are heavily used in Geographic Information Systems (GIS) such as online maps and navigation systems. The extraction of roads from overhead images has been, and mainly still is, performed by expert annotators. For a long time, handcrafted methods have been developed with the aim of reducing the burden of the annotators. However, they were not precise enough to replace them. Indeed, their outputs were imperfect and had a high dependence on user-adjustable parameters [23]. In recent years, the availability of large amounts of data [32], combined with the increase in compute capabilities, has led to a rise in the popularity of deep learning methods based on Convolutional Neural Networks [19]. Modern road extraction methods mostly fall into two kinds of approaches, each coming with its own drawbacks. The first kind [5,26,44,45] relies on a pixel-based segmentation which is then converted to a graph using compute-intensive handcrafted algorithms that require a lot of manual tuning and can often yield partial graphs. The production of raster road masks using neural networks is a computationally expensive task in itself because of the large number of layers required to retrieve a segmentation that has the same resolution as the input image. Iterative graph construction methods are the second kind of deep learning based methods for road extraction [3, 21, 22, 36]. These approaches explore the map from an initial starting location using successive applications of a neural network on patches centered on the current point to predict the location of the next point (*i.e.* next move pre-

diction). While having the advantage of directly inferring a graph, this is a very slow process. In addition, several starting locations may need be picked in order to cover the possible multiple connected components of the final graph.

Our approach aims to solve the aforementioned issues of current deep learning based methods by offering a fast and convenient way to extract roads from satellite or aerial images. Inspired by recent advances in neural networks for object detection [37], we design a fully convolutional detection head that learns to precisely locate multiple points of interest such as intersections, turns, and dead ends across the whole image in a single pass. Node features attached to these points are simultaneously regressed by this head, and fed to a Graph Neural Network (GNN) which predicts links between them to form the final graph without requiring any post-processing. Both networks are jointly trained end-to-end from ground truth road graphs, which are widely available from sources such as OpenStreetMap. We benchmark our method against state-of-the-art approaches on the RoadTracer [3] dataset and find that it is orders of magnitude faster than recent competing methods while retaining competitive accuracy.

With the increasing interest in deep learning inference on embedded hardware, such as satellites or drones [1], our method, combined with recent advances in CNN and GNN quantization [2, 6], opens the possibility of on board in-flight road extraction, which could reduce ground computation and bandwidth needs by transmitting graphs instead of images. Real-time road network monitoring is also a future possibility which would greatly benefit applications such as disaster response, by rapidly obtaining and transmitting updated maps to facilitate evacuation (*e.g.* in the event of a flood or earthquake).

The main contribution of our work is threefold. First, we propose a fast fully convolutional method for precise localisation of points of interest. Second, we design a Graph Neural Network for road link prediction. Third, we propose a performance-oriented architecture highly competitive against state-of-the art approaches.

2. Related Works

2.1. Detection Neural Networks

In recent works, Convolutional Neural Networks (CNN) such as VGG [35] or ResNet [12] which were originally used for image classification, are stripped of their final classifier and repurposed as fully convolutional feature extractors (often called backbones) for other computer vision tasks, such as semantic segmentation [25]. Object detection methods such as YOLO [30] or FCOS [37] have taken advantage of the fully convolutional nature of these backbones to design very fast single stage object detection neural networks, which infer bounding boxes and classes in a single forward pass and can be trained end-to-end, as opposed to two-stage approaches that perform these tasks separately. Single-stage architectures often follow a similar structure composed of a backbone which extracts features, a neck outputting feature representations at different scales [24], and several detection heads performing the final task.

2.2. Road Extraction

A large number of unsupervised methods have been developed for road graphs extraction. We refer the reader to recent extensive surveys on the subject for more information [22]. In this section, we focus on recent road extraction methods based on deep learning. Early deep learning based methods for road extraction were patch-based applications of fully connected neural networks [28]. However, most recent neural network-based methods proceed differently and can be sorted into two groups.

The first group of methods is based on pixel-level segmentation. Inspired by the success of FCN [25] and U-Net [31], these methods use a backbone as an encoder and learned upscaling as a decoder to retrieve a classification of individual pixels of an input image. In [44], residual connections are added to a U-Net to improve road predictions. D-LinkNet [45] adapts the LinkNet architecture to form a U-Net-like network and achieve better connectivity. DeepRoadMapper [26] uses a ResNet-based [12] FCN to produce a segmentation which is then converted to a graph using thinning and an additional connection classifier. In [5], the authors improve the connectivity of extracted networks by jointly learning the segmentation and orientation of roads. All segmentation-based approaches have the drawback of relying on post-processing techniques such as morphological thinning to convert the predicted road pixels to a road graph. Thus, these methods are slow and the final graphs can lack connectivity.

The second kind of approaches iteratively construct a graph from an initial starting point by successively applying a neural network that predicts the location of next point of the graph on a patch around the current point. These methods are inspired by the way human annotators iteratively create road graphs. In [39], a network predicts which points from the border of the output patch are linked to the center point. RoadTracer [3] uses a CNN to predict the direction of the next move with a fixed step size. PolyMapper [21] uses a Recurrent Neural Network (RNN) guided by segmentation cues to predict the road topology and building polygons. VecRoad [36] implements a variable step size for next move prediction in order to achieve better alignment of intersections. DeepWindow [22] estimates the road direction along a center line regressed by a CNN using a Fourier spectrum analysis algorithm. While these methods have the advantage of outputting a road graph directly, they cannot guarantee the exploration of the whole map from a single

starting location. The choice of initial starting locations is a hard problem in itself. Moreover, the repeated application of CNNs makes these approaches very slow.

2.3. Relational Inference and GNNs

Graph Neural Networks (GNNs) [33] are effective models for encoding interactions, and have recently been applied to link prediction [43], relational structure discovery, such as to model the dynamics of complex systems [15], and to the modeling of relational knowledge [34]. Kipf and Welling [16] proposed a graph autoencoder using a Graph Convolutional Network (GCN) encoder [17] and a simple dot-product decoder to model undirected networks and applied it to the link prediction task. Follow-up work [34] models different edge types in relational data using a graph neural network encoder and a decoder based on the DistMul [42] factorization. In [15] the authors propose a message-passing [10] encoder that alternates between the computation of node and edge features that can later be leveraged to predict the evolution of the system of interest several time steps in advance. The aforementioned models assume the ground-truth graph is partially known, or operate on the complete graph as an uninformative prior [15]. A parallel line of work is that of learning the unknown graph explicitly. The Dynamic Graph CNN model [41] proposes to dynamically build a graph by *k*-NN search in the feature space after each application of the EdgeConv operator also introduced in [41]. The approach has been extended in [13], where the authors address the question of the differentiable construction of a discrete graph using the recently proposed Gumbel Top-*k* trick [18] - a stochastic and differentiable relaxation of *k*-NN search - and decouple the construction of the graph and the learning of graph features amenable for downstream tasks. With the introduction of deeper architectures [11, 20] came increased interest for efficient implementations of GNNs, an issue relevant to our work as on-board processing of satellite images, such as for road graph extraction, is a pressing issue. [40] applied bucket-based quantization of matrix-matrix products to accelerate the GCN [17] operator. [2] proposed a general framework for binarizing graph neural networks and introduced efficient binarized versions of the Dynamic EdgeConv operator [41] with real-world speed-ups on a low-power device.

3. Method

In this section, we present our method based on a fully convolutional head for regression of points of interest and a Graph Neural Network (GNN) for link prediction. An overview of our method is available in Figure 2.

3.1. General architecture

Our neural network architecture follows the trend of using a pre-trained CNN as a feature extractor (backbone). We choose to work at a single level of detail and thus feed the output feature maps of the feature extractor directly to a single head. Using a single head allows us to save computation time and to make the training process more streamlined. Indeed, we remove all ground truth assignment complications that arise when using multiple heads.

Each branch of our head is composed of three convolutional layers working on N_{feat} feature maps. The first two branches are the junction-ness and offset branch, which form the point-of-interest detection model. The third branch is a node feature branch, which computes features that will be used to predict links between points detected by the "point-of-interest" branch.

Let H_I, W_I be the dimensions of the input RGB image $I \in \mathbb{R}^{H_I \times W_I \times 3}$. We call "detection cells" the pixels of the final output feature maps of the point-of-interest detection network. With the ResNet-50 [12] backbone used in our experiments, the height and width H, W of the input feature maps $F^{in} \in \mathbb{R}^{H \times W \times N_{\text{in}}}$ are 32 times smaller than the ones of I. This means that the final layers of our head have $H \times W$ output cells. In the case of an input image size of 512×512 pixels, for example, we thus obtain 16x16=256 output cells, which can each regress the position of one point.

3.2. Point detection branch

Our point of interest detection branch, shown in Figure 2 is inspired by recent developments in single shot detection neural networks [30, 37], which regress a bounding box for each output "cell" of a detection head, even if they do not necessarily contain an object. This allows object detection to be performed using very fast Fully Convolutional Networks. These methods represent bounding boxes as offsets relative to the center of the cell regressed by a first output and filter the detected objects using a second separate output. We adapt this design to the regression of points of interest by outputting a "junction-ness" score J_j for each output cell j (middle of Figure 2), as well as a two-dimensional vector $\mathbf{o}_j = [u_j, v_j] \in [-0.5; 0.5]^2$ representing the offset of the regressed point with respect to the center of its cell (top of Figure 2). A point $\mathbf{p}_j$ is detected in an output cell j if $J_j > J_{thr}$, where J_{thr} is the junction-ness detection threshold. We can retrieve the coordinates (x_j, y_j) of $\mathbf{p}_j$ in the original input image:

$$x_j = \frac{u_j + X_j + 0.5}{W} \cdot W_I, y_j = \frac{v_j + Y_j + 0.5}{H} \cdot H_I \quad (1)$$

where (X_j, Y_j) are the coordinates of the cell j in the output feature map F^{out}. The "junction-ness" and offset ground truths are generated on the fly during data loading by reversing Equation 1 using the sub-graph covered by each input patch.

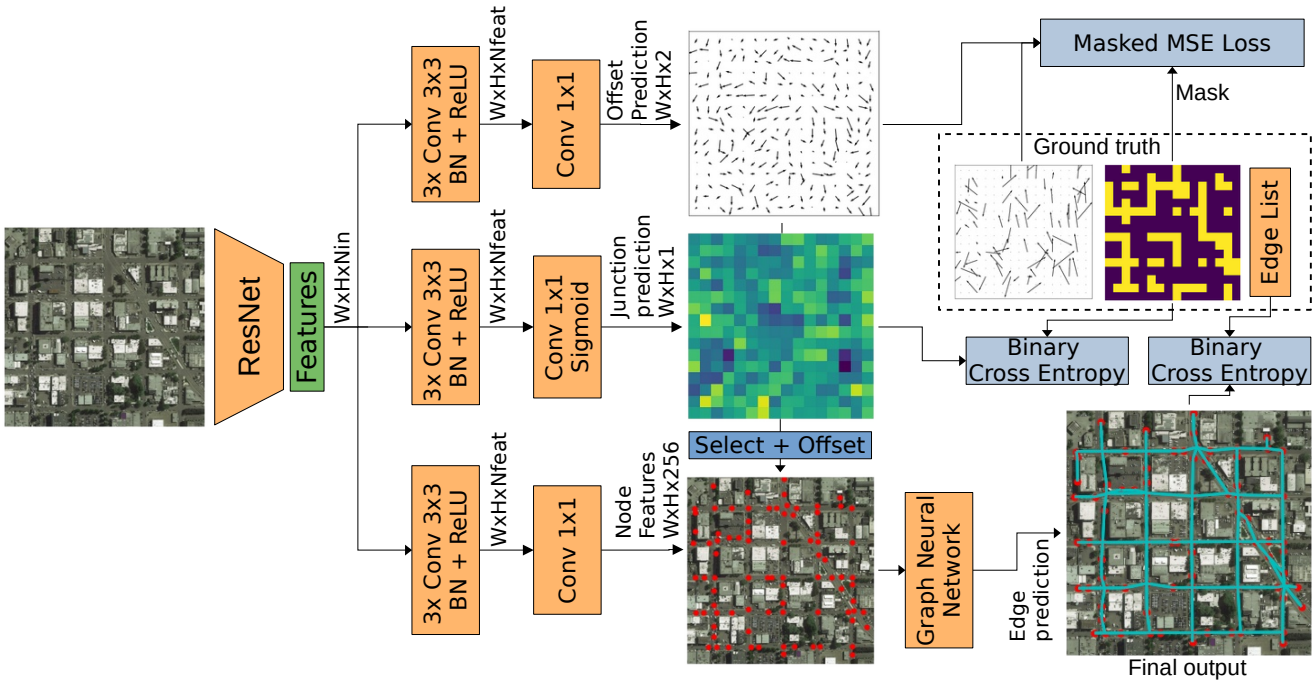

Figure 2. Our single-shot road graph extraction neural network. Features computed using a ResNet [12] backbone are fed to three branches. The junction prediction branch identifies cells which contain points of interest and is trained with binary cross-entropy. The offset prediction branch regresses the precise location of these points inside of the cells by outputting a two-dimensional vector field and is trained with a Mean Squared Error loss. Points selected by the junction and offset branch are paired with node features from a third branch and fed to a Graph Neural Network which predicts the presence of edges between pairs of points to create the final output.

3.3. Edge prediction

Once junctions have been extracted from the image, our task is to predict links between connected points of interest. This is a relational inference - or a latent graph learning - problem which we solve with a graph neural network. Starting with an initial prior connectivity estimation - *i.e.* a set of edges - $\mathcal{E}^0$, we aim at both inferring missing edges and discarding irrelevant ones to learn the true road graph.

Formally, let j be one of the detected junctions. We denote by $X_j = F_j^{out}$ the corresponding features in the output feature map of the backbone, and $\mathbf{p}_j = (x_j, y_j)$ the 2D Cartesian coordinates of the junction in the input image, computed from the offset vectors.

Our method computes initial node embeddings

$$\mathbf{x}_j = f(X_j, \mathbf{p}_j). \tag{2}$$

We choose the function f to be either a 2D Convolutional Neural Network defining an additional node features branch responsible for dimensionality reduction and transformation of the raw image features, or the identity. In the latter case the task of transforming the raw image features is left to the GNN. We then perform several (*e.g.* three) message-passing iterations using the EdgeConv operator [41]:

$$\mathbf{e}_{ij}^{(l)} = \text{ReLU}\left(\boldsymbol{\theta}^{(l)}(\mathbf{x}_j^{(l-1)} - \mathbf{x}_i^{(l-1)}) + \boldsymbol{\phi}^{(l)}\mathbf{x}_i^{(l-1)}\right) \tag{3}$$

$$= \text{ReLU}\left(\boldsymbol{\Theta}^{(l)}\tilde{\mathbf{X}}^{(l-1)}\right) \tag{4}$$

$$\mathbf{x}_i^{(l)} = \max_{j \in \mathcal{N}(i)} \mathbf{e}_{ij}^{(l)} \tag{5}$$

followed by Batch Normalization, where $\tilde{\mathbf{X}}^{(l-1)} = \left[\mathbf{x}_i^{(l-1)} \,||\, \mathbf{x}_j^{(l-1)} - \mathbf{x}_i^{(l-1)}\right]$, $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ are trainable weights, and $\boldsymbol{\Theta}$ their concatenation. The resulting increase in receptive field allows for connectivity patterns to be learned based on both local image and graph properties, and shared context across junctions.

Finally, each possible edge is scored using a simple scoring function g applied on the final node features of the graph. Edge features must be computed explicitly for all possible edge, we therefore choose node features to prevent combinatorial explosion. We note

$$p_{ij} = \sigma(g(\mathbf{x_i}, \mathbf{x_j})) \tag{6}$$

the inferred probability that the edge $\mathbf{e}_{ij}$ exists, where σ is the sigmoid function. In practice, we choose g to be either a single bilinear layer, *i.e.*

$$g(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T W_g \mathbf{x}_j \tag{7}$$

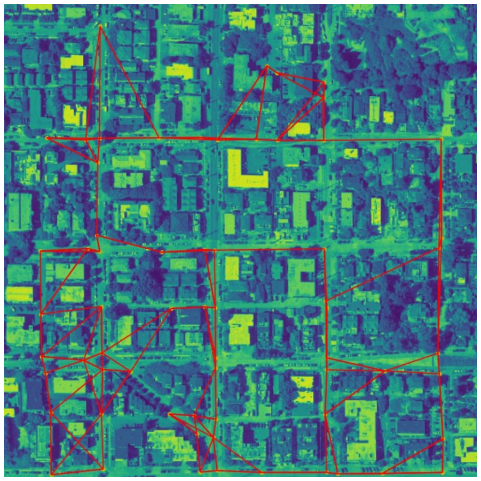

Figure 3. $k = 4$-NN graph constructed in feature space on features learned by an additional *node features head* combined with an MLP edge classifier.

or a multi-layer MLP classifier (please refer to the supplementary material for more details and comparison with other decoders such as the dot product).

Connectivity estimation Our choice of operator is motivated by the fact that the connectivity is unknown, and therefore we may initialize the graph learning process with the complete graph (*i.e.* considering all possible connections between junctions), or introduce a sparse prior by building the graph dynamically by k-NN search in feature space. We also considered the case where the initial graph is complete, while subsequent iterations of message passing are done on dynamically inferred connectivity. We choose $k = 4$ motivated by the observations that junctions with more than four incident roads are rare. To verify this intuition, we trained a model using a three-layer (2D convolutions followed by Batch Normalization and ReLU activations) node features head and an MLP classifier for each possible edge. The model was trained to convergence, and the four nearest neighbours graph was built on the output of the node features branch. The resulting graph is shown in Figure 3, and demonstrates that a high quality sparse initialization of the connectivity can be obtained in this manner.

3.4. Loss function

To train our neural network our loss function is defined as a sum of three terms:

$$\mathcal{L} = \mathcal{L}_{\text{jun}} + \mathcal{L}_{\text{off}} + \mathcal{L}_{\text{edge}} \tag{8}$$

where $\mathcal{L}_{\text{jun}}$ and $\mathcal{L}_{\text{edge}}$ are the binary cross-entropy loss which we use to train our junction-ness branch and our edge prediction Graph Neural Network.

$\mathcal{L}_{\text{off}}$ is a Mean Squared Error offset loss masked with the positive junction predictions, which we use to train our offset regression branch. We define it as:

$$\mathcal{L}_{\text{off}} = \frac{1}{\sum \mathbb{1}_{J_{i,j} > J_{thr}}} \sum_{i,j} \mathbb{1}_{J_{i,j} > J_{thr}} (v_{i,j} - V_{i,j})^2 \tag{9}$$

Where V is the ground truth offset vector field. The mask is used so that the predictions of the offset branch are not conditionned to the presence of nodes.

4. Experiments

In this section, we compare our method against other road extraction approaches on the widely-used RoadTracer dataset [3], which is composed of 35 training cities (180 images of 4096x4096 pixels) and 15 test cities of 8192x8192 pixels. The low number of training images, low ground sampling distance, high resolution and high annotation density of this dataset make it quite challenging. For iterative methods [3, 36], we used the starting locations hard-coded into the provided code.

4.1. Implementation details

We implement our method in the Pytorch 1.9.1 [29] framework based on CUDA 11. We choose a ResNet-50 backbone (thus $N_{\text{in}} = 2048$) and $N_{\text{feat}} = 256$. We perform edge classification on the complete graph (other cases included in the supplementary material) using a three-layer GNN and a 2-layer MLP scoring function. J_{thr} is set to 0.5 for the main results. We use basic data augmentation techniques such as random flips, and train our network on 512x512 pixel random crops for 2350 epochs, which took 24 hours on our system with a single Nvidia RTX 3090 GPU with 24GB of VRAM, an AMD Ryzen 9 3900X CPU and 64GB of RAM. We use the Adam [14] optimizer with a learning rate of $1e^{-3}$. We did not use any loss balancing.

We also run performance benchmarks. All numbers were obtained on a computer with two Intel Cascade Lake 6248 20-core CPUs, 192GB of RAM, and four Nvidia Tesla V100 GPU with 16GB of VRAM. 10 CPU cores, a single GPU, and a quarter of the available memory were assigned to each run.

4.2. Choice of input resolution

By design, our method is only able to find a single point of interest by output cell and is directly trained on ground truth annotations. However, depending on the dataset, and especially in very dense neighborhoods, several ground truth points may fall into the same cell. We chose that these points would effectively be merged into their centroid, while keeping incoming edges from outside of that cell linked to that new point. However, this situation should be avoided as much as possible, since it can shift intersections,

Ours RoadTracer [3] DeepRoadMapper [26]

Figure 4. Comparison of our method against an iterative method [3] and a segmentation based method [26] on a challenging area of the RoadTracer dataset. The first line is the full image. The second line shows crops of the regions in red squares. Our method is able to find more roads, even in sections that are completely missed by other methods.

and link close points that should not be linked (e.g. parallel roads). Images are often resized when being forwarded through a neural network, as a compromise between speed, memory and accuracy. In our case, the resizing ratio has to be chosen carefully according to the ground sampling distance (GSD) of the dataset and the density of ground truth annotations. To estimate the correct ratio for each dataset, we evaluate the average number of points in each positive cell over the whole dataset, at a wide range of ratios. This average should be as close to one as possible, in order to limit the effects of point merging. Figure 5 shows this average for the popular RoadTracer [3], SpaceNet3 [38] and Massachussetts roads [27] datasets.

4.3. Qualitative results

Figure 4 shows qualitative results against an iterative method and a segmentation-based method. Our method is able to find a noticeably larger number of roads than RoadTracer [3] and DeepRoadMapper [26] in this challenging image. We can also notice that there are a few areas with low connectivity compared to RoadTracer. Indeed, since their network "walks" through the entire regressed graph, it is inherently made of a single connected component. However, the result of RoadTracer is missing entire portions of the city (e.g. top left) because they are separated from the starting position by a highway that the network was not able to cross during its exploration. Our method does not suffer from such limitations. More qualitative results are available in Figure 1 and in supplementary material.

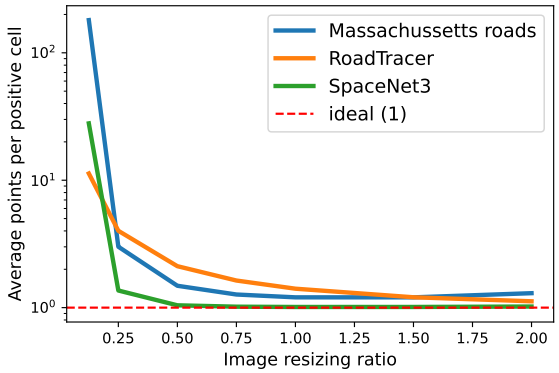

Figure 5. Average ground truth points per positive output cell at a range of image resizing ratios for 3 popular road extraction datasets. The ideal average is one, to prevent the collapse of neighboring nodes as much as possible.

4.4. Quantitative evaluation

We follow the literature and use the three metrics defined in [36] to evaluate the accuracy of our network. P-F1 is a pixel-based F1 score obtained by comparing the rasterized output graph and ground truth. J-F1 is a junction-based F1 score based on local connectivity. APLS is the Average Path Length Similarity defined in the SpaceNet challenge [38] and is based on the comparison of shortest paths in the predicted graph and the ground truth.

The results of our testing are shown in Table 1. Our method achieves results which are competitive with DeepRoadMapper [26] and RoadTracer [3]. It is however outperformed by VecRoad [36]. Our intuition is that VecRoad "brute-forces" these metrics by regressing a lot more points and edges than our method, as shown in Section 4.6 and Table 3. Iterative methods inherently yield a better connectivity as their output is always a single connected component. This is reflected in the scores.

Method	P-F1	J-F1	APLS
DeepRoadMapper [26]	56.85	29.05	21.27
RoadTracer [3]	55.81	49.57	45.09
RoadCNN [3]	71.76	32.22	53.18
VecRoad [36]	**72.56**	**63.13**	**64.59**
Ours	57.2	39.23	46.93

Table 1. Accuracy metrics on the RoadTracer Dataset, evaluated using open source code from [36] and [3].

4.5. Performance benchmarks

Run-time is seldom mentioned in the road extraction literature, but it is a very important metric, especially as we move towards inference on edges devices. Thus, we benchmark the performance of our method against other recent approaches with open source code. We remove data loading and output times from all methods because different libraries and formats were used and thus only account for pre-processing, inference and post-processing steps of each algorithm. We provide average performance numbers for a single 8192x8192 test image from the RoadTracer [3] dataset, as well as numbers for the whole dataset (15 images), since some methods such as VecRoad [36] have optimizations for multiple-image inference. Our method can run on the whole dataset at once by performing the sliding window in batches, which is a bit faster.

Method	Type	Single image	Whole dataset
VecRoad [36]	Iterative	1902.5	8873.4
RoadTracer [3]	Iterative	579.4	8690.5*
DeepRoadMapper [26]	Seg.	1361.6	20423.4
RoadCNN [3]	Seg.	278.2	4172.4
Ours	Graph	**20.9**	**295.0**

Table 2. Run-time in seconds on the RoadTracer Dataset. The single image score is averaged over the whole test set. Our method is the fastest by a large margin. *RoadTracer failed on some images, thus this number is extrapolated from the average.

The results of this experiment are shown in Table 2 and Figure 6. We observe that iterative methods are much slower because of the repeated forward passes required for next move estimation. Segmentation methods are not particularly fast as well, as the upscaling computation performed by a learned decoder is very compute intensive, and the extra post-processing steps required for graph conversion are slow. Our method is the fastest one by a large margin and offers a very good compromise between accuracy and speed. Indeed, it achieves similar APLS scores as RoadTracer [3] while running 27 times faster. Our approach also beats the APLS scores of DeepRoadMapper [26] by a large margin while running 65 times faster. It has lower accuracy scores than VecRoad [36], however it is 91 times faster. Our method could be made even faster by using a smaller and more efficient backbone, such as ResNet-18 [12] or DarkNet-53 [30].

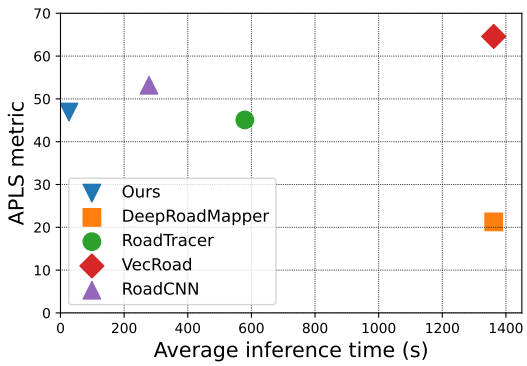

Figure 6. APLS metric with respect to inference time for our method, compared to state-of-the art approaches. Our method is the fastest and offers a very good compromise between accuracy and speed.

4.6. Graph complexity

Another interesting metric which is seldom mentioned in the road extraction literature is the notion of graph complexity. Graphs are a sparse representation and are meant to use a lot less storage space than masks. Most algorithms commonly used on graphs (e.g. shortest path algorithms) have a complexity which depends on the number of nodes and edges of the graph. Thus, offering accurate graphs with a low number of unnecessary nodes and edges is also beneficial in terms of run-time of subsequent applications. To evaluate the complexity of the graphs regressed by our method and compare it to other approaches, we propose a complexity score (lower is better) which is simply the total number of graph elements (nodes and edges) divided by the APLS score. We use the APLS because it seems to be the most popular metric. This score represents the compromise between the accuracy and compactness of graphs.

The results of this experiment are shown in Table 3. We observe that our method obtains the lower complexity compared to VecRoad [36] and DeepRoadMapper [26]. This is mainly due to our sparse approach to the regression of junctions, as shown by the average number of nodes. In Figure 7, we show the way our method finds an optimal representation of a neighborhood compared to the very high number of nodes and edges found by VecRoad [36].

Method	Nodes	Edges	Total	APLS	Complexity
VecRoad [36]	29620	61042	90662	**64.59**	1404
DRM [26]	6071	11963	18034	21.27	848
RoadTracer [3]	8263	17044	25306	45.09	561
Ours	4343	17273	21615	46.93	**461**

Table 3. Comparison of average graph complexity scores (Total elements divided by APLS. Lower is better). Nodes (resp. Edges) is the average number of nodes (resp. edges) in the output graphs over the whole RoadTracer test set. DRM = DeepRoadMapper. Our method achieves the lowest complexity compared to other approaches, which make it a good compromise between accuracy and compactness of graphs.

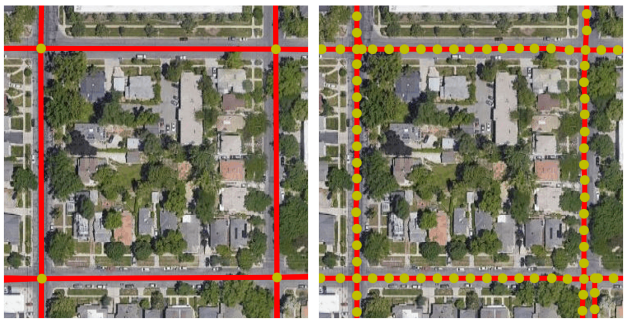

Figure 7. Exemple of the low complexity of the graphs inferred by **our method (left)** compared to VecRoad [36] (right). Our method finds an optimal 4-node representation of this simple neighborhood, whereas iterative methods tend to find overly complex representations. Having graphs with a low number of elements will save storage and speed up the computation of algorithms performed on these graphs, such as shortest path algorithms.

5. Limitations

Our method comes with some limitations. The first one, as said in previous sections, is the difficulty to work in very dense areas with lots of intersections. Since multiple intersections might fall into the same detection cell, our network will only be able to detect one of them, which can lead to shifted junctions. Nevertheless, our method still works well on dense areas of Tokyo shown in Figure 4. The second limitation is the inability to be trained on mask-based datasets such as DeepGlobe [7] without converting the masks to graphs first. Finally, some long sections of road such as highways or bridges might lack points of interest since they do not have any junctions. Our network can miss these sections of road in the final graph as shown in Figure 8 and in some parts of Figure 1. This issue could be addressed by working at different scales using multiple heads or by adding virtual points-of-interest on long stretches of road.

6. Conclusion

We proposed a novel architecture for single-shot extraction of road graphs from satellite images. Our method first predicts sparse interest points using an image CNN feature extractor as well as a junction detection head, and an offset regression head, trained jointly to identify candidate road intersections in a lower-resolution image and predict their position in the high-resolution image. We then combine the extracted image features and regressed point coordinates with a graph neural network to combine local and global information and infer the unknown graph structure. Our method combines aspects of graph learning and link prediction to score candidate connections between road junctions and infer the road graph. We demonstrate competitive performance with state of the art methods on three reference metrics while achieving significantly lower graph complexity and inference times (up to 91x faster than iterative models) compared to iterative and segmentation-based methods.

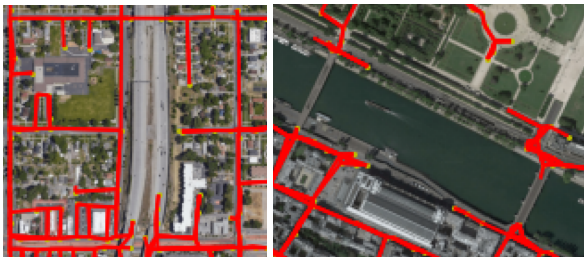

Figure 8. Examples of long sections of road without points of interest, which are occasionally missed by our method.

Even though the experimental validation of our approach was focused on road networks, our single-shot graph extraction framework can be applied to other types of problems. An example would be the extraction of blood vessels in medical images [8]. An interesting follow-up to road extraction could be to not only decide if edges exist or not in the final graph, but also infer edge attributes like road types, amount of traffic, or speed limits, as done in [9]. We also believe that our method could be adapted to tackle map update problems that have been presented in newly released datasets [4]. In addition, our method can further be combined with existing model compression techniques, both from the Euclidean [6] and Geometric [2] deep learning literature, to pave the way for on board single-shot extraction of road graphs on edge devices.

Acknowledgements

G.B. is funded by the CIAR project at IRT Saint Exupéry. M. B. was supported by a Department of Computing scholarship from Imperial College London, and a Qualcomm Innovation Fellowship. The authors are grateful to the OPAL infrastructure from Université Côte d'Azur for providing resources and support. This work was granted access to the HPC resources of IDRIS under the allocation 2020- AD011011311R2 made by GENCI.

References

[1] Gaétan Bahl, Lionel Daniel, Matthieu Moretti, and Florent Lafarge. Low-power neural networks for semantic segmentation of satellite images. In *ICCV Workshops*, 2019. 2

[2] Mehdi Bahri, Gaétan Bahl, and Stefanos Zafeiriou. Binary graph neural networks. In *CVPR*, 2021. 2, 3, 8

[3] Favyen Bastani, Songtao He, Sofiane Abbar, Mohammad Alizadeh, Hari Balakrishnan, Sanjay Chawla, Sam Madden, and David DeWitt. Roadtracer: Automatic extraction of road networks from aerial images. In *CVPR*, 2018. 1, 2, 5, 6, 7, 8

[4] Favyen Bastani and Samuel Madden. Beyond road extraction: A dataset for map update using aerial images. In *ICCV*, 2021. 8

[5] Anil Batra, Suriya Singh, Guan Pang, Saikat Basu, CV Jawahar, and Manohar Paluri. Improved road connectivity by joint learning of orientation and segmentation. In *CVPR*, 2019. 1, 2

[6] Adrian Bulat and Georgios Tzimiropoulos. Xnor-net++: Improved binary neural networks. *arXiv preprint arXiv:1909.13863*, 2019. 2, 8

[7] Ilke Demir, Krzysztof Koperski, David Lindenbaum, Guan Pang, Jing Huang, Saikat Basu, Forest Hughes, Devis Tuia, and Ramesh Raskar. Deepglobe 2018: A challenge to parse the earth through satellite images. In *CVPR Workshops*, 2018. 8

[8] Muhammad Moazam Fraz, Paolo Remagnino, Andreas Hoppe, Bunyarit Uyyanonvara, Alicja R Rudnicka, Christopher G Owen, and Sarah A Barman. Blood vessel segmentation methodologies in retinal images–a survey. *Computer methods and programs in biomedicine*, 108(1), 2012. 8

[9] Zahra Gharaee, Shreyas Kowshik, Oliver Stromann, and Michael Felsberg. Graph representation learning for road type classification. *Pattern Recognition*, 120, 2021. 8

[10] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *ICML*, 2017. 3

[11] Shunwang Gong, Mehdi Bahri, Michael M Bronstein, and Stefanos Zafeiriou. Geometrically principled connections in graph neural networks. In *CVPR*, 2020. 3

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 3, 4, 7

[13] Anees Kazi, Luca Cosmo, Nassir Navab, and Michael Bronstein. Differentiable graph module (dgm) for graph convolutional networks. *arXiv preprint arXiv:2002.04999*, 2020. 3

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[15] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *ICML*, 2018. 3

[16] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016. 3

[17] Thomas N Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Neural Networks. In *ICLR*, 2017. 3

[18] Wouter Kool, Herke Van Hoof, and Max Welling. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. In *ICML*, 2019. 3

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *NeurIPS*, 2012. 1

[20] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. DeepGCNs: Can GCNs go as deep as CNNs? In *ICCV*, 2019. 3

[21] Zuoyue Li, Jan Dirk Wegner, and Aurélien Lucchi. Topological map extraction from overhead images. In *ICCV*, 2019. 1, 2

[22] Renbao Lian and Liqin Huang. Deepwindow: Sliding window based on deep learning for road extraction from remote sensing images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 13, 2020. 1, 2

[23] Renbao Lian, Weixing Wang, Nadir Mustafa, and Liqin Huang. Road extraction methods in high-resolution remote sensing images: A comprehensive review. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 13, 2020. 1

[24] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 2

[25] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 2

[26] Gellert Mattyus, Wenjie Luo, and Raquel Urtasun. Deeproadmapper: Extracting road topology from aerial images. *ICCV*, 2017. 1, 2, 6, 7, 8

[27] Volodymyr Mnih. *Machine learning for aerial image labeling*. University of Toronto (Canada), 2013. 6

[28] Volodymyr Mnih and Geoffrey E. Hinton. Learning to detect roads in high-resolution aerial images. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6316 LNCS, 2010. 2

[29] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *NeurIPS*. 2019. 5

[30] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016. 2, 3, 7

[31] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, 2015. 2

[32] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3), 2015. 1

[33] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1), 2009. 3

[34] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European semantic web conference*. Springer, 2018. 3

[35] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2

[36] Yong-Qiang Tan, Shang-Hua Gao, Xuan-Yi Li, Ming-Ming Cheng, and Bo Ren. Vecroad: Point-based iterative graph exploration for road graphs extraction. In *CVPR*, 2020. 1, 2, 5, 6, 7, 8

[37] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *ICCV*, 2019. 2, 3

[38] Adam Van Etten, Dave Lindenbaum, and Todd M Bacastow. Spacenet: A remote sensing dataset and challenge series. *arXiv preprint arXiv:1807.01232*, 2018. 6

[39] Carles Ventura, Jordi Pont-Tuset, Sergi Caelles, Kevis Kokitsi Maninis, and Luc Van Gool. Iterative deep learning for road topology extraction. *BMVC*, 2018. 2

[40] Junfu Wang, Yunhong Wang, Zhen Yang, Liang Yang, and Yuanfang Guo. Bi-GCN: Binary Graph Convolutional Network. In *CVPR*, 2021. 3

[41] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic Graph CNN for Learning on Point Clouds. *ACM Trans. Graph.*, 38(5), oct 2019. 3, 4

[42] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*, 2014. 3

[43] Muhan Zhang and Yixin Chen. Link Prediction Based on Graph Neural Networks. In S Bengio, H Wallach, H Larochelle, K Grauman, N Cesa-Bianchi, and R Garnett, editors, *NeurIPS*, 2018. 3

[44] Zhengxin Zhang, Qingjie Liu, and Yunhong Wang. Road extraction by deep residual u-net. *IEEE Geoscience and Remote Sensing Letters*, 15(5), 2018. 1, 2

[45] Lichen Zhou, Chuang Zhang, and Ming Wu. D-linknet: Linknet with pretrained encoder and dilated convolution for high resolution satellite imagery road extraction. In *CVPR Workshops*, 2018. 1, 2