

---

# Image-Conditioned Graph Generation for Road Network Extraction

---

**Daide Belli**  
Informatics Institute  
University of Amsterdam  
davidbelli95@gmail.com

**Thomas Kipf**  
Informatics Institute  
University of Amsterdam  
t.n.kipf@uva.nl

## Abstract

Deep generative models for graphs have shown great promise in the area of drug design, but have so far found little application beyond generating graph-structured molecules. In this work, we demonstrate a proof of concept for the challenging task of road network extraction from image data. This task can be framed as image-conditioned graph generation, for which we develop the *Generative Graph Transformer* (GGT), a deep autoregressive model that makes use of attention mechanisms for image conditioning and the recurrent generation of graphs. We benchmark GGT on the application of road network extraction from semantic segmentation data. For this, we introduce the *Toulouse Road Network* dataset, based on real-world publicly-available data. We further propose the *StreetMover* distance: a metric based on the Sinkhorn distance for effectively evaluating the quality of road network generation. The code and dataset are publicly available<sup>1</sup>.

## 1 Introduction

Hundreds of thousands kilometres of roads around the world have not been mapped yet [1]. Collecting and regularly updating world maps is key to improving autonomous driving systems, optimizing industrial transportation and helping first-aid operations in case of natural disasters. Current research on automated road network extraction tries to find efficient and scalable solutions to this task by using state-of-the-art deep learning models [2, 3, 4]. In particular, existing approaches [5, 6, 7, 8, 9] generate a semantic segmentation of road networks and then apply manually-engineered heuristics and post-processing steps to extract graph representations. Post-processing is a critical component in those methods, used for example to join disconnected road sections or to remove isolated subgraphs resulting from noisy segmentations. In this work, we present a graph generative model to automatically extract road networks from semantic segmentation data, removing the necessity for post-processing heuristics and allowing for a complete end-to-end solution to the problem.

The contribution of this paper is threefold:

- We release the *Toulouse Road Network* dataset for the task of road network extraction from semantic segmentation of satellite images.
- We introduce the *Generative Graph Transformer*, a deep autoregressive model for conditional graph generation which scales well to large graphs thanks to a recurrent formulation and self-attention mechanisms.
- We propose the *StreetMover* distance, an efficient metric for the comparison of road networks. This metric is based on the Sinkhorn distance [10] between point clouds, and it is invariant with respect to node permutation, graph translation and rotation.

---

<sup>1</sup>The code and dataset are available in  
<https://github.com/davide-belli/generative-graph-transformer> and  
<https://github.com/davide-belli/toulouse-road-network-dataset>

## 2 Generative Graph Transformer

There are two main approaches to neural network-based graph generation explored in recent literature: *one-shot* and *recurrent* generation. One-shot approaches [11, 12, 13] emit graphs at once in the form of complete adjacency and feature matrices. In the case of recurrent generation [14, 15, 16, 17], a deep-autoregressive model sequentially expands a graph by iteratively adding new nodes and edges. Conditional generation of graphs has been applied for scene graph generation [18, 19], drug discovery [20, 11] and for modeling chemical reactions [21]. Concurrently and independently of our work Chu et al. [22] introduced a generative model for street network extraction from satellite images. The proposed GGT model is designed for the recurrent, conditional generation of graphs and consists of an encoder-decoder architecture as outlined in Fig. 1.

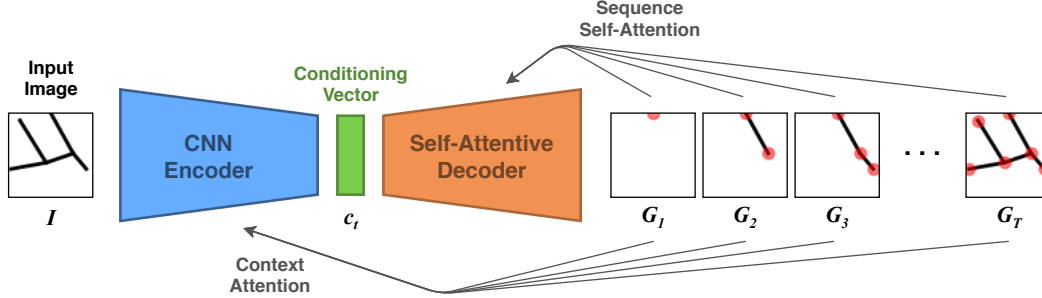


Figure 1: Outline of GGT. An image  $I$  is passed through the encoder which produces a conditioning vector  $c_t$  using a context attention mechanisms on the previously generated node. The self-attentive decoder uses the conditioning vector and the previously generated nodes to predict the next node in the graph. This sequential process incrementally generates the graph  $G_T$ .

### 2.1 Self-Attentive Graph Generation

The proposed model takes as input a grayscale image  $I \in \mathbb{R}^{64 \times 64}$  and incrementally generates a graph  $\mathcal{G} = (\tilde{\mathbf{A}} \in \mathbb{R}^{N \times N}, \tilde{\mathbf{X}} \in \mathbb{R}^{N \times 2})$  representing the road network. Here,  $N$  is the number of nodes in the graph,  $\tilde{\mathbf{A}}$  is a soft adjacency matrix containing probability values in range  $[0,1]$  and  $\tilde{\mathbf{X}}$  is a node feature matrix containing normalized coordinates in range  $[-1,+1]$ .

The decoder network is inspired by the self-attentive decoder in the Transformer [23], which has proven effective in a variety of tasks and domains [24, 25, 26, 27, 28, 29]. At each time-step  $t$  in the generation, the inputs to the decoder are the previously generated node features  $\tilde{\mathbf{x}}_{<t}$ , adjacency vectors  $\tilde{\mathbf{a}}_{<t}$ , and a conditioning vector  $c_t$  obtained from the image encoder. The concatenated inputs are positionally encoded using a sinusoidal vector  $\mathbf{p}_t$  [23], and then fed to a linear layer to obtain the initial hidden representation  $\mathbf{h}_t^{(0)} = \mathbf{W}_{in}([\tilde{\mathbf{a}}_{<t-1}, \tilde{\mathbf{x}}_{<t-1}, c_t] + \mathbf{p}_t) \in \mathbb{R}^d$ , where  $d = 256$ . Afterwards, a series of  $L$  decoding blocks with  $1 \leq l \leq L$ , defined as follows, are applied:

$$\tilde{\mathbf{h}}_t^{(l)} = \text{LN}(\mathbf{h}_t^{(l)} + \text{MultiHead}(\mathbf{h}_t^{(l)}, \mathbf{h}_{<t}^{(l)})), \quad \mathbf{h}_t^{(l+1)} = \text{LN}(\tilde{\mathbf{h}}_t^{(l)} + \mathbf{W}_n^{(l)} \text{ReLU}(\mathbf{W}_m^{(l)} \tilde{\mathbf{h}}_t^{(l)})) \quad (1)$$

where the MultiHead operator refers to the self-attention as in Vaswani et al. [23], LN is layer normalization [30], and  $\forall l, \mathbf{W}_m^{(l)} \in \mathbb{R}^{2048 \times d}, \mathbf{W}_n^{(l)} \in \mathbb{R}^{d \times 2048}, \mathbf{h}_t^{(l)}, \tilde{\mathbf{h}}_t^{(l)} \in \mathbb{R}^d$ . The representation after the last layer is then fed to two distinct MLP heads which emit the predicted node coordinates and (soft) adjacency vector as follows:

$$\tilde{\mathbf{a}}_t = \sigma(\mathbf{W}_{a2} \text{ReLU}(\mathbf{W}_{a1} \mathbf{h}_t^{(L)})), \quad \tilde{\mathbf{x}}_t = \tanh(\mathbf{W}_{x2} \text{ReLU}(\mathbf{W}_{x1} \mathbf{h}_t^{(L)})) \quad (2)$$

where  $\mathbf{W}_{a1}, \mathbf{W}_{x1} \in \mathbb{R}^{128 \times d}, \mathbf{W}_{a2} \in \mathbb{R}^{M \times 128}, \mathbf{W}_{x2} \in \mathbb{R}^{2 \times 128}$ , and  $M$  is the maximum size of the frontier in the BFS-ordering (see Appendix A.2 for details).

### 2.2 Image Conditioning

**Image encoder** To condition the generative process, we use a simple CNN encoder which takes as input a semantic segmentation  $I \in \mathbb{R}^{64 \times 64}$  and emits a low-dimensional representation as  $\mathbf{c} = \text{CNN}(I) \in \mathbb{R}^{900}$ . To speed up the training and improve the convergence of the end-to-end model, we pre-train the encoder as part of an autoencoder trained for a reconstruction task.

**Image attention** In the basic implementation of the CNN architecture, the image features are kept the same for every time-step in the graph generation, i.e.,  $\mathbf{c}_t = \mathbf{c} \forall t$ . However, the decoder model may benefit from focusing on different parts of the image depending on what components are currently being generated. For this reason, we introduce an image attention mechanism on the CNN encoder based on the context attention introduced by Xu et al. [31]. This mechanism is implemented as an MLP which takes as input the flattened visual features  $\mathbf{c} = \text{CNN}(\mathbf{I}) \in \mathbb{R}^{900}$  and the previously generated node features  $\tilde{\mathbf{x}}_{<t}$  and  $\tilde{\mathbf{a}}_{<t}$ , and outputs a mask vector:

$$\mathbf{s}_t = \mathbf{W}_{c2} \text{ReLU}(\mathbf{W}_{c1}[\tilde{\mathbf{a}}_{<t}, \tilde{\mathbf{x}}_{<t}, \mathbf{c}]), \quad \mathbf{m}_t = \frac{\exp(\mathbf{s}_t)}{\sum_{i=1}^{|\mathbf{s}_t|} \exp(\mathbf{s}_{ti})}, \quad \mathbf{c}_t = \mathbf{c} \odot \mathbf{m}_t \quad (3)$$

where  $\mathbf{W}_{c1}, \mathbf{W}_{c2}^\top \in \mathbb{R}^{1800 \times 900}$ ,  $\mathbf{s}_t$  is the vector of attention scores of length  $|\mathbf{s}_t| = 900$ , and  $\mathbf{m}_t$  is the mask vector applied on the visual features through the element-wise product operation  $\odot$ .

## 3 Experiments

### 3.1 Toulouse Road Network Dataset

To benchmark the Generative Graph Transformer, we introduce the *Toulouse Road Network dataset*, based on publicly available data from OpenStreetMap<sup>2</sup>. The dataset contains patches of road maps from the city of Toulouse, represented both as graphs  $\mathcal{G} = (\mathbf{A}, \mathbf{X})$  and as grayscale segmentation images  $\mathbf{I}$ . The generation of the dataset includes a sequence of preprocessing and filtering steps to clean the data, followed by data augmentation and the representation of graphs under a canonical ordering, as in You et al. [16]. In Appendix A.1, we present additional dataset details and discuss the procedure used to generate the dataset.

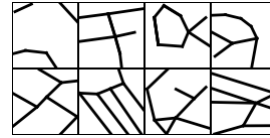


Figure 2: Samples from the Toulouse Road Network dataset.

### 3.2 StreetMover Distance

This work introduces the *StreetMover* distance to evaluate generative models for road networks. Road networks are first represented as point clouds by sampling a fixed number of equidistant points over the edges of the graphs. Then, the StreetMover distance is computed as the optimal cost of moving the predicted proposal point cloud to the ground-truth target point cloud. Sinkhorn iterations [10] are used for an efficient approximation of the Wasserstein distance. The StreetMover distance can be interpreted as describing the cost of moving road segments in the predicted graph to match the shape of the ground-truth target graph, as shown in Fig. 3. The main benefits of the StreetMover distance are its interpretability, scalability, and invariance with respect to node permutation graph translations, and rotations. In Appendix A.3, we further motivate the introduction of the StreetMover distance by discussing other related evaluation metrics.

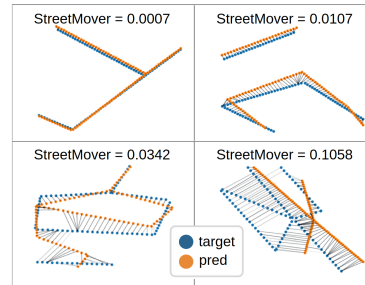


Figure 3: Examples of StreetMover distances between several graphs.

### 3.3 Experimental Setup

We evaluate the proposed Generative Graph Transformer in the road extraction task on the *Toulouse Road Network* dataset. We compare the performance of the model with simple MLP and RNN baselines and with an extension of GraphRNN [16] for labeled graph generation. We also conduct an ablation study comparing the GGT with and without context attention in the encoder. We choose the StreetMover distance as the main metric, supported with additional metrics such as the average error in the number of nodes and edges ( $\Delta|V|$ ,  $\Delta|E|$ ). We report additional details on baseline implementations and hyper-parameter setup in Appendix A.2.

The models are trained optimizing the following loss:

$$\mathcal{L} = \lambda \mathcal{L}_{\mathbf{A}} + (1 - \lambda) \mathcal{L}_{\mathbf{X}} = \lambda \text{BCE}(\tilde{\mathbf{A}}, \mathbf{A}) + (1 - \lambda) \text{MSE}(\tilde{\mathbf{X}}, \mathbf{X}) \quad (4)$$

<sup>2</sup><https://www.openstreetmap.org/>

which combines the reconstruction errors in the predicted adjacency and node feature matrices  $\tilde{\mathbf{A}}$  and  $\tilde{\mathbf{X}}$ . The hyper-parameter  $\lambda$  regulates the trade-off between the two components. We use teacher forcing in the self-attentive decoder during training.

### 3.4 Results

Table 1 reports the evaluations on the *Toulouse Road Network* dataset. The proposed GGT model achieves the best performance according to all metrics. The GGT decreases the average StreetMover distance by 45% with respect to the simple RNN baseline, compared to only 15% decrease when choosing GraphRNN. The one-shot generation using the MLP does not seem to be effective for this task, as proven by the sharp decline in all the scores. The ablation study on the context attention confirms that introducing attention in the encoder contributes to improvements in the overall results.

Table 1: Comparison with the different baselines, and ablation study removing the context attention from the encoder (GGT without CA). Standard deviation is computed over 3 runs with each model.

	StreetMover	$\mathcal{L}_{\text{valid}}$	$\Delta E $	$\Delta V $
MLP	$0.1380 \pm 0.0050$	$0.1090 \pm 0.0020$	$3.38 \pm 0.07$	$3.02 \pm 0.05$
RNN	$0.0289 \pm 0.0003$	$0.0330 \pm 0.0002$	$1.01 \pm 0.05$	$0.96 \pm 0.02$
GraphRNN	$0.0245 \pm 0.0004$	$0.0311 \pm 0.0001$	$0.87 \pm 0.06$	$0.85 \pm 0.06$
GGT without CA	$0.0192 \pm 0.0007$	$0.0213 \pm 0.0001$	$0.75 \pm 0.04$	$0.79 \pm 0.03$
GGT	<b><math>0.0158 \pm 0.0006</math></b>	<b><math>0.0205 \pm 0.0001</math></b>	<b><math>0.65 \pm 0.05</math></b>	<b><math>0.71 \pm 0.06</math></b>

To better understand the performance of GGT, we present in Fig. 4 a set of qualitative studies to analyze the reconstructions and attention mechanisms. Overall, the reconstructed graphs have high fidelity, even in more complicated cases with loops, cluttered edges or large graphs (see Fig. 4a). Moreover, we show in Fig. 4b how graphs from adjacent patches can be easily merged to reconstruct road networks at larger scales. Finally, by inspecting the self-attention layers in the GGT, we see in Fig. 4c how some heads are responsible for learning the structure in the graphs, emitting attention weights that highly correlate with corresponding lower triangular adjacency matrices.

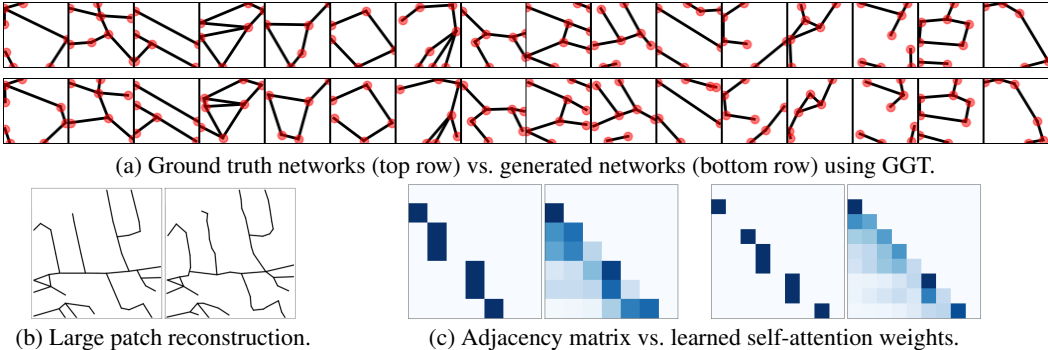


Figure 4: Qualitative studies on the GGT. In a) we compare ground-truth road networks (top row), with generated ones (bottom row). Nodes are added in red for visualization purposes. In b) we show the reconstruction of a larger  $4 \times 4$  patch of the map (ground-truth on the left, reconstruction on the right). In c) we explore with two examples the correlation between ground-truth adjacency matrices (left) and attention weights emitted by self-attention heads in intermediate GGT layers (right).

## 4 Conclusions and Future Work

In this work we presented the real-world *Toulouse Road Network* dataset to benchmark methods for road network extraction from images. We propose the *Generative Graph Transformer*, a deep autoregressive model based on self-attention for the recurrent, conditional generation of graphs. Moreover, we introduced the *StreetMover* distance, a scalable, efficient and permutation-invariant metric for graph comparison.

A challenge that remains open in this field is the development of a complete end-to-end solution combining semantic segmentation and graph extraction. Applying the proposed GGT model to other graph generation tasks, such as drug design, is another interesting direction for future work.

## Acknowledgements

T.K. acknowledges funding by SAP SE. We would like to thank Sindy Löwe, Gabriele Cesa and Gabriele Bani for their feedback on the first draft of this paper.

## References

- [1] Christopher Barrington-Leigh and Adam Millard-Ball. The world’s user-generated road map is more than 80% complete. *PloS one*, 12(8):e0180698, 2017.
- [2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [3] 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*, pages 234–241. Springer, 2015.
- [4] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.
- [5] Lin Gao, Weidong Song, Jiguang Dai, and Yang Chen. Road extraction from high-resolution remote sensing imagery using refined deep residual convolutional neural network. *Remote Sensing*, 11(5):552, 2019.
- [6] Corentin Henry, Seyed Majid Azimi, and Nina Merkle. Road segmentation in sar satellite images with deep fully convolutional neural networks. *IEEE Geoscience and Remote Sensing Letters*, 15(12):1867–1871, 2018.
- [7] Shivaprakash Muruganandham. Semantic segmentation of satellite images using deep learning, 2016.
- [8] Adam Van Etten, Dave Lindenbaum, and Todd M Bacastow. Spacenet: A remote sensing dataset and challenge series. *arXiv preprint arXiv:1807.01232*, 2018.
- [9] Xiaoming Gao, Christopher Klaiber, Drishtie Patel, and Jeff Underwood. Ai is supercharging the creation of maps around the world [blog post. july 23, 2019]. retrieved from <https://tech.fb.com>, Jul 2019.
- [10] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in neural information processing systems*, pages 2292–2300, 2013.
- [11] Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *International Conference on Artificial Neural Networks*, pages 412–422. Springer, 2018.
- [12] Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018.
- [13] Shuangfei Fan and Bert Huang. Deep generative models for generating labeled graphs. 2019.
- [14] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018.
- [15] Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander Gaunt. Constrained graph variational autoencoders for molecule design. In *Advances in Neural Information Processing Systems*, pages 7795–7804, 2018.
- [16] Jiaxuan You, Rex Ying, Xiang Ren, William L Hamilton, and Jure Leskovec. Graphrnn: A deep generative model for graphs. *arXiv preprint arXiv:1802.08773*, 2018.

- [17] Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, Charlie Nash, William L Hamilton, David Duvenaud, Raquel Urtasun, and Richard S Zemel. Efficient graph generation with graph recurrent attention networks. *arXiv preprint arXiv:1910.00760*, 2019.
- [18] Sebastian Schuster, Ranjay Krishna, Angel Chang, Li Fei-Fei, and Christopher D Manning. Generating semantically precise scene graphs from textual descriptions for improved image retrieval. In *Proceedings of the fourth workshop on vision and language*, pages 70–80, 2015.
- [19] Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. Graph r-cnn for scene graph generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 670–685, 2018.
- [20] Yibo Li, Liangren Zhang, and Zhenming Liu. Multi-objective de novo drug design with conditional graph generative model. *Journal of cheminformatics*, 10(1):33, 2018.
- [21] John Bradshaw, Matt J Kusner, Brooks Paige, Marwin HS Segler, and José Miguel Hernández-Lobato. A generative model for electron paths. *arXiv preprint arXiv:1805.10970*, 2018.
- [22] Hang Chu, Daiqing Li, David Acuna, Amlan Kar, Maria Shugrina, Xinkai Wei, Ming-Yu Liu, Antonio Torralba, and Sanja Fidler. Neural turtle graphics for modeling city road layouts. *arXiv preprint arXiv:1910.02055*, 2019.
- [23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.
- [24] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8), 2019.
- [25] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.
- [26] Chung-Cheng Chiu, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjali Kannan, Ron J Weiss, Kanishka Rao, Ekaterina Gonina, et al. State-of-the-art speech recognition with sequence-to-sequence models. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4774–4778. IEEE, 2018.
- [27] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. *arXiv preprint arXiv:1802.05751*, 2018.
- [28] Benson Chen, Regina Barzilay, and Tommi Jaakkola. Path-augmented graph transformer network. *arXiv preprint arXiv:1905.12712*, 2019.
- [29] Łukasz Kaiser, Aidan N Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. One model to learn them all. *arXiv preprint arXiv:1706.05137*, 2017.
- [30] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.
- [31] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015.
- [32] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [33] Chu-Cheng Lin and Jason Eisner. Neural particle smoothing for sampling from conditional sequence models. *arXiv preprint arXiv:1804.10747*, 2018.
- [34] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.
- [35] Zhou Wang, Alan C Bovik, Hamid R Sheikh, Eero P Simoncelli, et al. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

## A Appendix

### A.1 Dataset details

The *Toulouse Road Network* dataset contains 111,034 data points (map tiles), of which 80,357 are in the training set (around 72.4%), 11,679 in the validation set (around 10.5%), and 18,998 in the test set (around 17.1%). Each tile represents a squared region of side 0.001 degrees of latitude and longitude on the map, which corresponds to a square of around 110 meters. The semantic segmentation of each patch is represented as a  $64 \times 64$  grayscale image. The three splits are obtained using the criteria in Fig. 5. This criterion is chosen to optimize the diversity inside each split while keeping similar the data distributions between different splits. Also, this criterion minimizes the amount of data that has to be discarded in boundary regions between splits due to the use of horizontal and vertical translation in the augmentation procedure.



Figure 5: Criteria for defining the different splits in the dataset. Gray areas are discarded due to the content overlap resulting from the augmentation by translation.

**Dataset generation** To generate the *Toulouse Road Network* dataset we start from publicly available data from OpenStreetMap, where the road network is represented as a set of segments defined by the coordinates of extreme points. In the first step of dataset generation, we extract squared tiles from the map, each with side 0.001 degrees. The following preprocessing steps include: i) detection of edge intersections, ii) merging nodes that are distant less than 0.00005 degrees apart, and iii) merging consecutive edges resulting in an almost straight road, where the incidence angle is between  $75^\circ$  and  $90^\circ$ . Furthermore, we filter the proposed data points in order to remove trivial graphs ( $|V| \leq 3$ ), and extremely cluttered graphs, removing the right tail of the population after the 95th percentile ( $|V| \geq 10$ ,  $|E| \geq 16$ ). Finally, we include the possibility to augment the dataset with translation, flip and rotation, resulting in an augmentation factor of up to 128 times the number of original data points. Samples of networks in our dataset are shown in Fig. 2, where every graph is plotted as a  $64 \times 64$  image.

In order to train auto-regressive models, we also define a canonical ordering for each graph in the dataset based on a BFS-ordering over the nodes, breaking ties in the ordering consistently. When choosing the initial node in the sequence, or when continuing the BFS after completing the traversing of a connected component, we select the top-left node among the unvisited ones. In the case of multiple edges branching out from the current node, we order the edges clockwise with regards to the incoming edge.

### A.2 Details on Experimental Setup

**Training settings** For all the baselines and Generative Graph Transformer variations, we run the same hyper-parameter search on: learning rate, batch size, weight decay, and  $\lambda$  coefficient. In all the experiments we use the Adam optimizer [32] with parameters  $\eta \in [3 \cdot 10^{-4}; 5 \cdot 10^{-4}]$ ,  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$  and  $\epsilon = 10^{-8}$ . For all the models, the batch size is set to 64, except for the RNN where we find 16 to be better. Weight decay parameters in the range  $[10^{-5}, 5 \cdot 10^{-5}]$  are found to be optimal for regularization. For the  $\lambda$  hyper-parameter, we notice best performance in the range  $[0.30, 0.70]$ , and we set  $\lambda = 0.5$  in our experiments. We also try different sizes for the output of the node-wise GRU in GraphRNN, and for the number of decoder blocks and heads in the GGT.

We train all the models for 200 epochs (around 10 hours) on an Nvidia GeForce 1080Ti GPU. We use early stopping according to the StreetMover distance in the validation set to select the best model.

At test time, we greedily generate binary vectors from the predicted adjacency vectors  $\tilde{a}$  by thresholding the values at 0.5. We expect that more advanced techniques like beam search or better sampling strategies [33] could significantly improve the results. We leave these experiments as future works.

**Implementation details and baselines** The baselines for our experiments are implemented as follows. The MLP decoder consists of two MLP heads each with a single hidden layer of 1600

neurons, followed by a ReLU non-linearity. The two heads emit the node features  $\tilde{\mathbf{X}}$  and a symmetric adjacency matrix  $\tilde{\mathbf{A}}$  (the simmetricity is enforced by modeling only the upper-triangular portion). The RNN decoder introduces a single-layer GRU [34] with 256-dimensional output before the two MLP heads. In this case, the MLP heads only emit a feature vector and adjacency vector describing the current node in the BFS-ordering. For the GraphRNN, we extend the original architecture with an MLP head on the node-level GRU in order to emit node features. We find a 16-dimensional node representation to work best for the modified GraphRNN. Similarly to the GGT decoder, both the simple RNN and the GraphRNN decoders are conditioned on the image  $\mathbf{I}$  by concatenating the visual features  $\mathbf{c}$  to the inputs of the node-level GRU. Finally, the Generative Graph Transformer has  $L = 12$  self-attention + MLP decoding blocks, with input and output dimensionalities fixed to:  $d = 256$ . Each multi-head self-attention has 8 heads, meaning that each head attends over 32-dimensional vectors. On top of the decoding blocks, two MLP heads emit node features and adjacency vectors as in the RNN decoder.

The CNN encoder is composed of two  $3 \times 3$  convolutional layers followed by a  $1 \times 1$  convolution, with a  $2 \times 2$  max pooling after the first convolutional layer. Batch normalization and Leaky-ReLU are used after every hidden layer.

**Loss function** The loss presented in Eq. 4 can be expanded as:

$$\begin{aligned} \mathcal{L} &= \lambda \mathcal{L}_{\mathbf{A}_{\text{valid}}} + (1 - \lambda) \mathcal{L}_{\mathbf{X}_{\text{valid}}} \\ &= \lambda \text{BCE}(\tilde{\mathbf{A}}, \mathbf{A}) + (1 - \lambda) \text{MSE}(\tilde{\mathbf{X}}, \mathbf{X}) \\ &= \frac{\lambda}{N \cdot M} \sum_{i=1}^N \sum_{j=1}^M -\left(a_{i,j} \log(\tilde{a}_{i,j}) + (1 - a_{i,j}) \log(1 - \tilde{a}_{i,j})\right) + \frac{1 - \lambda}{2N} \sum_{i=1}^N \|\mathbf{x}_i - \tilde{\mathbf{x}}_i\|_2^2, \end{aligned} \quad (5)$$

$\tilde{\mathbf{A}}$  and  $\tilde{\mathbf{X}}$  are the predicted adjacency and feature matrices, respectively.  $N$  is the length of the graph sequence under BFS-ordering (including termination tokens).  $M$  is the maximum size of the frontier of the BFS-ordering, set to the 99th percentile in the dataset population as in You et al. [16].

### A.3 Evaluation Metrics

An accurate choice of the evaluation metrics is necessary for a meaningful evaluation of the methods we compare. In particular, an optimal metric should jointly capture the accuracy of  $\tilde{\mathbf{X}}$  and  $\tilde{\mathbf{A}}$  while being invariant to changes in graph representation, graph transformations, and graph dimensionality ( $|V|$  and  $|E|$ ).

Pixel-based metrics like IoU, PSNR and SSI [35] computed on the image representation of the ground-truth and predicted graphs are not good candidates. Indeed, these metrics measure the local similarity between corresponding pixels in images but are not able to capture the magnitude of errors in the reconstructed graphs. Etten et al. [8] introduce Average Path Length Similarity (APLS), a metric to compare pairs of graphs based on simulated routing tasks between nodes in the graph. A relevant issue with APLS are the several post-processing steps used to clean and convert segmentation in graphs. Moreover, APLS is designed to evaluate problems related to semantic segmentation, like the presence of gaps between segments of roads. Metrics based on the sequential representation of the graphs, like the ones used in the loss function, are not useful at inference time because susceptible to mismatches between the ground-truth and reconstructed sequence.

The proposed StreetMover distance overcomes the problems presented for other metrics. Besides, the StreetMover distance is easily interpretable by plotting the alignment weights and efficient to compute thanks to the use of Sinkhorn iterations.

### A.4 Robustness to noisy segmentations

To investigate the effectiveness of the GGT in a setting closer to real-world applications, we simulate the quality seen in neural network trained for semantic segmentations by manually injecting random noise in the ground-truth segmentations. As shown for a few random samples in Fig. 6, we see good graph reconstructions in case of simple road networks. Low and medium levels of noise result in significant inaccuracies for graphs with more cluttered edges. The results are obtained without pre-training of the CNN encoder. We expect that pre-training the encoder as part of a denoising auto-encoder would significantly improve the robustness to input noise.



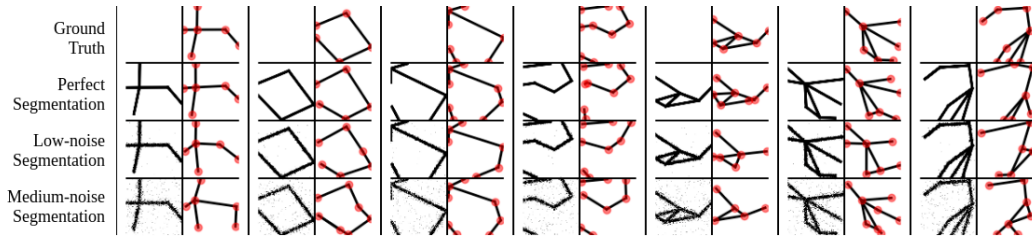
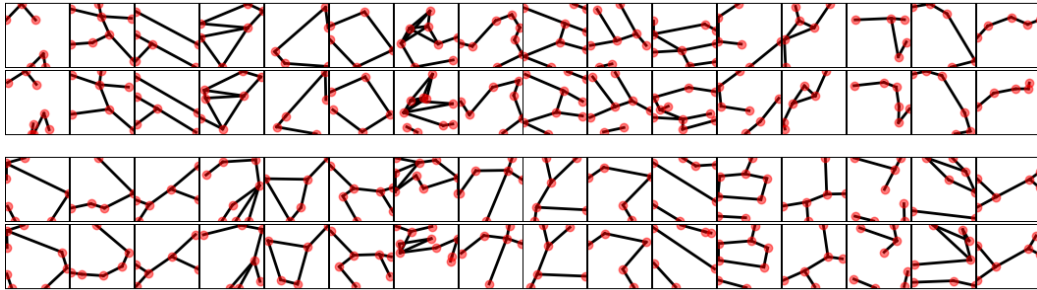
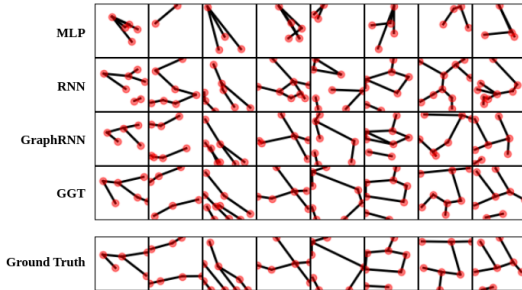


Figure 6: Experiment on the robustness of GGT to noisy input segmentations. In each column, the three left images show the input segmentations with zero, low and medium noise. On the right side of each column, the corresponding reconstructed graphs are compared with the ground-truths in the top row. Results are randomly sampled from the test set. Best seen zoomed-in on a screen.

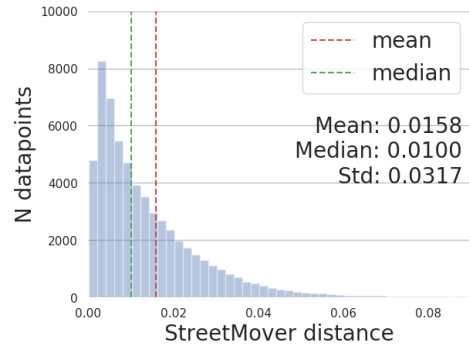
### A.5 Additional Qualitative Results



(a) Additional qualitative results showing reconstructions (bottom rows) and ground-truths (top rows) sampled from the test set.



(b) Comparing road network reconstructions generated by GGT with respect to other baseline models.



(c) Distribution of StreetMover distances between pairs of reconstructed and ground-truth networks in the test set, using GGT.

Figure 7: Additional qualitative studies on the GGT. In a) we report more reconstructions (bottom rows) and ground-truths (top rows) sampled from the test set. Nodes are added in red for better visualization. In b) we compare reconstructions generated by different models, confirming the relative increase in performance observed in table 1. In c) we visualize the histogram of StreetMover distances between graphs in the test set and reconstructions using GGT. We notice how in half of the cases the reconstructions are very accurate, with a StreetMover distance lower than 0.010. In the right tail of the population, few failure cases contribute to a mean StreetMover distance much higher than the median. The failure cases most frequently happen for complex graphs with very cluttered edges, as shown in the seventh column of examples in a).