Delineation of Road Networks Using Deep Residual Neural Networks and Iterative Hough Transform

Pinjing Xu$^{1[0000-0002-3637-6101]}$ and Charalambos Poullis$^{1[0000-0001-5666-5026]}$

Immersive and Creative Technologies Lab, Department of Computer Science and Software Engineering, Gina Cody School of Engineering and Computer Science, Concordia University, Montreal, Quebec, Canada
www.theICTlab.org

Abstract. In this paper we present a complete pipeline for extracting road network vector data from satellite RGB orthophotos of urban areas. Firstly, a network based on the SegNeXt architecture with a novel loss function is employed for the semantic segmentation of the roads. Results show that the proposed network produces on average better results than other state-of-the-art semantic segmentation techniques. Secondly, we propose a fast post-processing technique for vectorizing the rasterized segmentation result, removing erroneous lines, and refining the road network. The result is a set of vectors representing the road network. We have extensively tested the proposed pipeline and provide quantitative and qualitative comparisons with other state-of-the-art based on a number of known metrics.

Keywords: road network extraction · residual neural networks · semantic segmentation

1 Introduction

The automatic extraction of road networks from remote sensor imagery has long been a challenge not just to the GIS but also the computer vision communities. The vast variations in the road functions [e.g. rural, urban, highways, etc], colors [e.g. dirt road, asphalt] , shapes [e.g. winding mountain roads, straight highways], and sizes, make it an extremely challenging task. Recent attempts using deep learning techniques have shown promising results [14], [12], the majority of which reformulate the problem as a pixel classification problem and employ semantic segmentation techniques. Although this is useful in some cases, the majority of the applications employing road network data, e.g autonomous driving, GIS, etc, require that the network is in vector form; and in fact, it is this vectorization or linearization of the road network pixels that is perhaps one of the most challenging tasks.

In this paper we present a novel approach for extracting road networks in vector form. A deep convolutional neural network based on the SegNeXt architecture is trained to classify road pixels in satellite images of urban areas.

This architecture offers a reduced number of parameters and high localization accuracy therefore eliminating the need for the typical refinement of the segmentation results using MRF-based techniques. Furthermore, a novel loss function is proposed which provides better results than the typical loss functions used. The segmentation result is then vectorized and refined using a fast post-processing technique. During the post-processing, linear road segments are extracted using an iterative patch-based Hough transform technique which tracks the segments from one patch to the other. Next, a refinement process ensures that the nearby linear road segments are connected together to form a larger road network, and conflicting/overlapping parallel segments and other small segments are removed. The final result of the proposed technique is a road network in vector form which can be readily used in any GIS-based application.

Paper Organization. The paper is organized as follows: Section 2 provides a brief overview of the state-of-the-art in the area. In Section 3 we present an overview of the system. The network architecture is presented in detail in Section 4 including the training and validation tests. Section 4.4 explains the post-processing refinement process which converts the classified road pixels into the final road network vectors. Finally, Section 5 shows the comparisons of the proposed pipeline and the current state-of-the-art, and Section 6 presents the conclusion and future work.

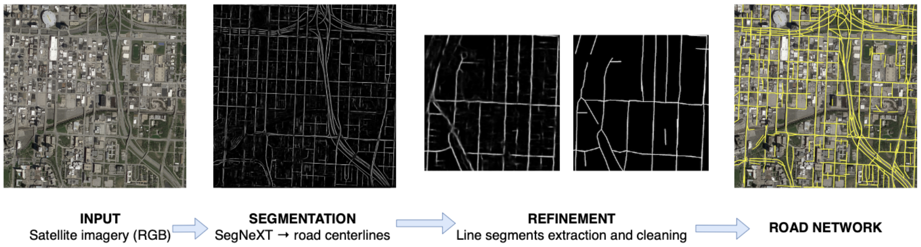

INPUT
Satellite imagery (RGB)
SEGMENTATION
SegNeXT → road centerlines
REFINEMENT
Line segments extraction and cleaning
ROAD NETWORK

Fig. 1: System overview. The input RGB image is processed using SegNeXt and results in a grayscale classification image of road and non-road pixels. The classification image is then divided into patches which are further processed. The refinement process involves an iterative application of patch-based Hough transforms which results in a set of extracted lines. Erroneously extracted lines resulting from misclassification are removed, and nearby lines are either connected (if not parallel) or suppressed (if parallel). The result is a set of vectors representing the road network in the input image shown in yellow overlaid on the input image.

2 Related Work

Below we provide a brief overview of the state-of-the-art in the area.

For many years the majority of the road extraction techniques relied solely on procedural approaches[6][1][7]. Of the most recent, is the work in [8] where the authors propose a technique for extracting the urban roads from satellite images by using orientation histograms and morphological profile features to guide a binary partition tree, thus achieving a higher accuracy.

Perhaps one of the first works on training deep neural networks for extracting large-scale road networks from satellite images is the work presented in [11]. The authors present a network trained on a challenging dataset with large context (i.e. larger sized patches containing a large number of urban features) in order to better differentiate between what is a road vs a non-road pixel.

Recently, more studies are following the semantic segmentation approach[4], [18], [15], [9]. The authors in [3] propose a fully convolutional network based on the U-Net family architecture with pre-trained ResNet-34 as the encoder. They optimize a loss function which combines the binary cross entropy and the intersection over union. During the test phase they report that data augmentation helps improve the prediction results even further.

In [17] the authors present a network which combines the ResNet and U-Net architectures to address the road network extraction. Their network employs skip connections within the residual units and between the encoding and decoding paths of the network to facilitate propagation of information, and also reduce the number of the generic U-Net's parameters by 75%.

The authors in [10] propose a semantic segmentation technique based on the ResNet architecture consisting of an encoder that compresses the image into a small feature map, and a fully convolutional decoder for generating the segmentation output. They also propose a post-processing technique for refining the segmentation result. Their approach relies on heuristics to connect and refine the roads (i.e. gap distances, accumulated error resulting from the first network's classification) which in some cases makes it hard to get complete and well optimized road networks.

At the time of writing this manuscript the best performing road network extraction technique is RoadTracer presented in [2]. The authors follow a new paradigm in which a CNN is used as a decision tool for tracing the road network in the image instead of using the neural network for semantic segmentation. Their approach has the pre-condition that the starting point lies on a road otherwise the tracing fails. Problems also arise in cases where tracing the road network runs out of points to process e.g. happens often when a bridge is reached. In this case, the result will be partial and disjoint from the entire road network and hard to recover the missing parts.

3 System Overview

The input to our system is an RGB image which is fed forward into a deep autoencoder network (SegNeXt) with aggregated residual transformations. The network outputs a semantic segmentation of the image in the form of a grayscale image in which each pixel is classified into a *road* or *non-road* classes. The classification image is then divided into patches which are further processed. During the refinement process, an iterative patch-based Hough transform is applied. Extracted lines are tracked from one patch to the other. Erroneously extracted lines resulting from misclassification are removed, and nearby lines are either connected (if not parallel) or suppressed (if parallel). The result is a set of vectors representing the road network in the input image. Figure 1 summarizes the system overview.

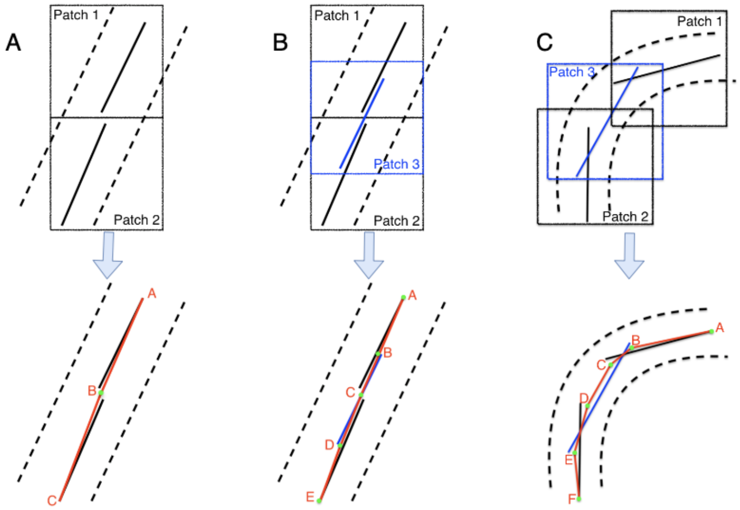


Fig. 2: Refinement process. Three cases are considered. A: no overlap between two windows, move the two nearby end points to the average location to connect these two line segments; B: 50% overlapping between windows, in addition to merging nearby end points of line segments, the merging of nearby points-to-lines is also needed; C: extract a curve by merging nearby lines. Smaller windows size and larger overlapping area will yield smoother result.

4 Network Architecture

The network architecture resembles that of a SegNeXt network [5] but with grayscale output and is shown in Figure 3. The network consists of three deep convolutional encoders and corresponding number of decoders with feed-forward links and cardinality-enabled residual-based building blocks. In each residual block the input data is split into multiple groups onto which different kernels are applied. A dilation of 2 is applied during convolution to introduce more spatial context. The feed-forward links from the encoders to the decoders help to retain high frequency information and improve the boundary delineation resulting in a smoother segmentation result therefore eliminating the need for any subsequent post-processing with conditional random fields (CRF), etc. In [16] these cardinality-enabled residual-based blocks used in shallow networks were shown to surpass in terms of performance other deeper CNNs. Thus, using these blocks the network can be shallower resulting in a smaller number of trainable network parameters therefore making the training process more effective.

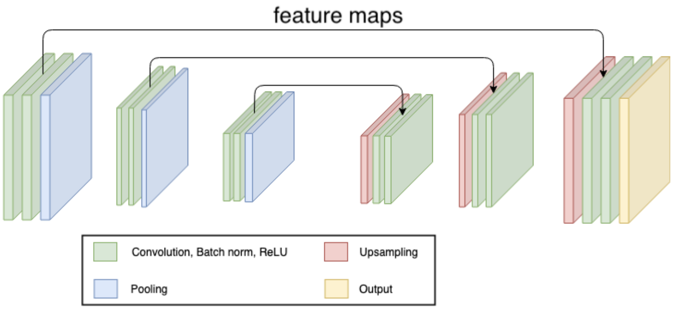


Fig. 3: SegNeXt-variant architecture

4.1 Dataset

For the training, validation and testing of the network we follow a similar approach as in [10]. A large number of satellite images acquired from Google Maps with resolution of 4096×4096 and ground sampling density of $60\frac{cm}{pixel}$ are used. These images are randomly selected in the $24km^2$ surrounding area of the GPS locations of 40 major cities. Training and validation is performed using images of 25 of these cities, and testing is performed on the images of the remaining 15 cities. Thus, there are no images of the same city between the training and testing datasets. The ground truth data is road center line extracted from the OpenStreetMap[13], width of road lines in ground truth images is 10 pixels.

4.2 Training

The network was trained on the images of the 25 cities. In order to maximize the training dataset we decided not to use a validation set during training but rather calculate the loss based after each epoch on three randomly selected patches from the training set. The training took 48 hours on a single NVIDIA GTX 1080Ti with an adaptive learning rate. We have used Keras API (with Tensorflow as backend engine) for the development of the network and the code will be made available as open source.

Input. The input to the network is a batch of 32 patches of size 200×200. Patches are selected using random sampling in order to ensure appropriate coverage.

Data augmentation. We apply a series of different data augmentation operations on the input patches. A histogram equalization is first applied to all patches in order to reduce possible high contrast resulting from the sun which appears as deep black shadows. Next, a number of transformations is performed consisting of random rotations in the range of $[0, \frac{\pi}{2}]$ degrees, scaling up/down by up to 70%, and random flipping on the vertical/horizontal axis.

Loss function. Perhaps the most widely used loss functions when dealing with a classification problem are the (a) Mean square error(MSE), and (b) Intersection over Union(IoU). However, due to the characteristics of MSE (takes the sum of a patch but ignores the positional relationships), it tends to result in a lot of noise in the segmentation output, and often yields bad performance. On the other hand, the use of an IoU loss function results in many gaps in the results as it was also recently reported in [2]. To address the aforementioned limitations, we propose a new loss function which comprises of both MSE and the inverse of IoU, and combines them as follows,

$$L = MSE \times \frac{union}{intersection} \tag{1}$$

Intuitively, the MSE is good at indicating whether a pixel is road or non-road, and the inverse of the IoU helps to reduce the noise.

4.3 Testing

During testing, we run the network on the image with a sliding window, with a window size of 200 × 200 and a step size of 100. Instead of thresholding the semantic segmentation result similar to many other semantic segmentation techniques, we remove the noise and extract roads by applying Hough transform to extract line segments in each window based on the network predictions. Since we have overlaps between the sliding windows, extracted line segments may not agree in different windows. Thus, a few more steps are needed to refine the result and get a clean road network.

4.4 Post-processing Refinement

Figure 2 shows all possible cases handled by our refinement process and what the resulting line segments will be. For a simple case, if there's no overlap between two windows (patches), extracted Hough lines will be like case A with no crossing over or overlap on one another. In this case, we merge all nearby line end points by moving them to their average location. Next, we consider the case of overlapping patches similar to case B in the Figure 2. If the extracted Hough lines on two patches do not "agree" with each other, intersection or misalignment will occur. To address the misalignment issue, we perform the following steps:

1. Merge all nearby line end points similar to case A;
2. For each line end point, search around itself for nearby lines, if there is a line **ab** passing by this end point, merge this end point into the line in the following steps:
   - break the line **ab**;
   - find the middle point **p** between the end point and the line;
   - move the end point to **p**;
   - connect the two end points **a** and **b** with point **p**, forming two new lines **ap** and **pb**

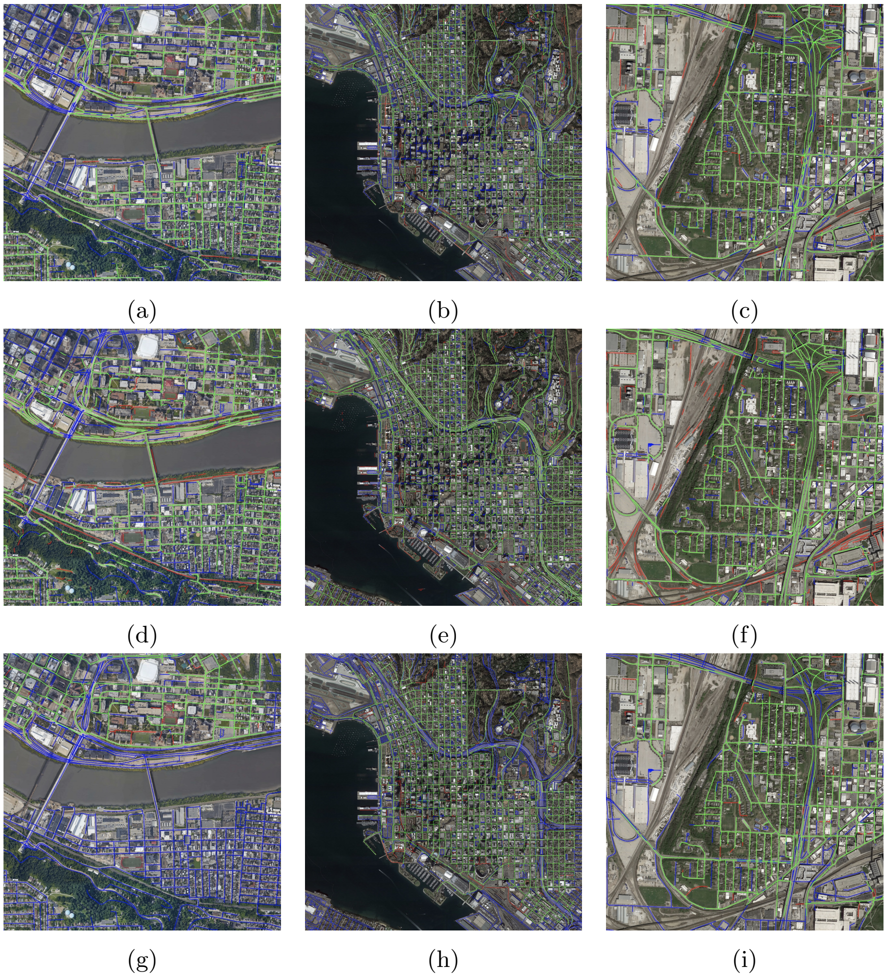


Fig. 4: Comparison between the proposed technique (a, b, c), DeepRoadMapper [10] (d, e, f), and RoadTracer [2] (g, h, i) for the cities of Pittsburgh (left column), San Diego (middle column), and Kansas City (right column). Green: true positives. Red: false positives. Blue: false negatives. Ground truth: OpenStreetMap [13]. The full resolution comparison results for all 15 cities and the source code can be downloaded from `http://theictlab.org/lp/2019Re_X/`

This procedure is repeated on all line end points in the image which results in all misaligned lines being removed. An advantage of this procedure is that Hough transform cannot extract curved roads from the network prediction but by extracting short line segments on each patch a curved road can be approximated as a set of piece-wise linear segments connected to each other. By merging nearby points-to-points (e.g. case A) and points-to-lines (e.g. case B), we can reconstruct a curved road or a circle with Hough lines (e.g. case C).

Iteratively looking through the entire image space could be a time consuming process, but since we are using sliding window technique during testing, only neighbours of current patch are in the searching range to merge nearby points and connect line segments. Meanwhile, in most cases, only one or two line segments will be found on road patches. Thus, this searching and merging progress is actually very fast.

Table 1: F1 Score, IoU and Junction metrics on 15 test cities. [2] RT: RoadTracer, [10] DRM: DeepRoadMapper (implementation provided in [2])

	Ours			**[2] RT**			**[10] DRM**		
City	**F1**	**IoU**	**Junction**	**F1**	**IoU**	**Junction**	**F1**	**IoU**	**Junction**
Amsterdam	0.28	0.16	0.16	0.01	0.01	0.01	0.22	0.13	0.04
Boston	0.71	0.55	0.58	0.67	0.51	0.74	0.77	0.62	0.66
Chicago	0.58	0.41	0.41	0.69	0.52	0.77	0.68	0.51	0.51
Denver	0.71	0.56	0.57	0.69	0.53	0.73	0.46	0.30	0.35
Kansas City	0.82	0.69	0.70	0.76	0.61	0.82	0.85	0.74	0.76
Los Angeles	0.68	0.51	0.51	0.73	0.57	0.79	0.73	0.58	0.61
Montreal	0.73	0.57	0.55	0.78	0.63	0.80	0.69	0.53	0.56
New York	0.51	0.34	0.35	0.73	0.57	0.84	0.42	0.26	0.29
Paris	0.59	0.42	0.26	0.67	0.51	0.71	0.41	0.26	0.31
Pittsburgh	0.71	0.55	0.57	0.41	0.26	0.48	0.69	0.58	0.57
Salt Lake City	0.75	0.60	0.65	0.73	0.58	0.79	0.58	0.41	0.47
San Diego	0.72	0.56	0.62	0.66	0.49	0.77	0.79	0.65	0.72
Tokyo	0.38	0.24	0.11	0.56	0.39	0.60	0.42	0.27	0.34
Toronto	0.69	0.53	0.48	0.76	0.61	0.74	0.79	0.65	0.69
Vancouver	0.41	0.26	0.25	0.65	0.49	0.70	0.45	0.29	0.29
Average	0.63	0.47	0.45	0.63	0.49	0.69	0.60	0.45	0.48

5 Evaluation

As of writing this manuscript the state-of-the-art in the area is considered to be the work presented in [2]. The authors have shown that they outperform all previously top performers in road extraction. Hence, we use the custom *junction metric* proposed by them in [2] and the well-known Intersection over Union (IoU) metric to evaluate our work and compare our results. The junction metric involves measuring the precision and recall based on the detected junctions in the inferred map. Furthermore, we report on additional metrics typically used in road extraction such as completeness, correctness, precision, recall, and F1 score.

Table 1 shows the comparison between the proposed approach and the two state-of-the-art RoadTracer [2] and DeepRoadMapper [10]. The F1 Score and IoU metrics shown are for the 15 test cities. As it can be seen, our method outperforms the DeepRoadMapper on the overall test set in both F1 score and IoU metrics. Our technique also surpasses the RoadTracer in accuracy on at least half of the cities. We attribute this to the fact that our approach initially results in a very high number of classified roads which the refinement process then prunes down leading to a lower error rate than the other techniques.

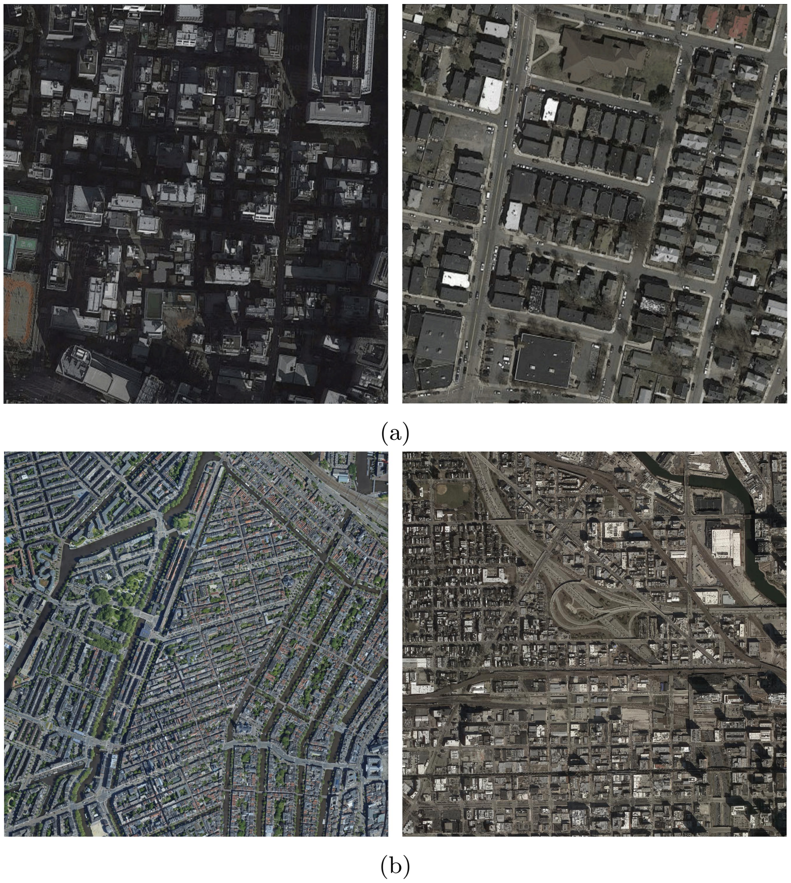
In terms of the junction metric, the results show that both semantic segmentation methods (ours and DeepRoadMapper) have the same level of performance in detecting the junctions, whereas the RoadTracer performs better because it seldom misclassifies road pixels around junctions.

As shown in the results shown in Figure 4, the road network resulting from our proposed method has relatively high completeness factor, and higher continuity than the results of DeepRoadMapper for the same areas. The images of some of the cities in the test dataset such as Tokyo and Amsterdam exhibit considerably different characteristics when compared to the images of other cities in the training set. As shown in Figure 5, the building density in Tokyo is much higher than other cities and the roads are narrower, therefore tall buildings produce shadows which occlude large parts of the roads. Amsterdam on the other hand has a different color temperature (tone). Both of these examples are seldom seen in the dataset during the training process, hence their presence in the testing dataset results in a higher misclassification rate. This is also evident from the reported metrics shown in Table 1.

Table 2: F1 score, IoU on 15 test cities with and without post-processing. [10] DRM: DeepRoadMapper (implementation provided in [2]). DRM (w/ PP): DeepRoadMapper, but replace its own post-processing with our post-processing. w/ PP: with our post-processing. w/o PP: without our post-processing

	Ours (w/ PP)		**Ours (w/o PP)**		**DRM [10]**		**DRM (w/ PP)**	
City	**F1**	**IoU**	**F1**	**IoU**	**F1**	**IoU**	**F1**	**IoU**
Amsterdam	0.28	0.16	0.26	0.15	0.22	0.13	0.21	0.12
Boston	0.71	0.55	0.62	0.45	0.77	0.62	0.80	0.67
Chicago	0.58	0.41	0.44	0.29	0.68	0.51	0.74	0.58
Denver	0.71	0.56	0.64	0.47	0.46	0.30	0.55	0.38
Kansas City	0.82	0.69	0.78	0.64	0.85	0.74	0.88	0.79
Los Angeles	0.68	0.51	0.57	0.39	0.73	0.58	0.77	0.63
Montreal	0.73	0.57	0.69	0.52	0.69	0.53	0.74	0.59
New York	0.51	0.34	0.41	0.26	0.42	0.26	0.42	0.27
Paris	0.59	0.42	0.30	0.18	0.41	0.26	0.42	0.27
Pittsburgh	0.71	0.55	0.59	0.42	0.69	0.58	0.75	0.60
Salt Lake City	0.75	0.60	0.72	0.56	0.58	0.41	0.64	0.47
San Diego	0.72	0.56	0.64	0.47	0.79	0.65	0.83	0.71
Tokyo	0.38	0.24	0.08	0.04	0.42	0.27	0.42	0.26
Toronto	0.69	0.53	0.67	0.51	0.78	0.65	0.83	0.71
Vancouver	0.41	0.26	0.35	0.21	0.45	0.29	0.47	0.31
Average	0.63	0.47	0.52	0.37	0.60	0.45	0.63	0.49

Table 2 shows the effect on the performance of the proposed post-processing method. A set of experiments were conducted to determine how the iterative Hough transform post-processing affects the accuracy and completeness of the



(a)

(b)

Fig. 5: Cities exhibiting different characteristics i.e. patterns, building densities, building heights, etc. The most commonly occurring city pattern/density in the training dataset looks like Boston (a)-right and Chicago (b)-right. Unique cases appearing in the test dataset none similar to which were seen by our network during training such as Tokyo (a)-left and Amsterdam (b)-left. Tokyo (a)-left is shown at the same zoom-level as Boston (a)-right; has much higher road and building density, roads are narrower, tall buildings produce shadows which occlude large parts of the roads. Amsterdam (b)-left is shown at the same zoom-level as Chicago (b)-right. The majority of the images used in training have similar color temperature (tone) as Boston and Chicago; in contrast Amsterdam has more green and gray areas.

extracted road network. First, we applied our pipeline on the aforementioned 15 cities with- and without- the proposed post-processing. Furthermore, we applied our post-processing method on the results of DeepRoadMapper by replacing its own post-processing steps. As it can be seen from the reported metrics the proposed post-processing method has a significant and positive effect on the evaluation results. Specifically using our network, in Tokyo where the network performs the worst the F1 score increased by 30% and IoU increased by 19% when using our post-processing method, while the overall F1 score increased by 10%, and IoU increased by 9%. For DeepRoadMapper, the average performance improved by 4% on both F1 score and IoU after substituting their post-processing method with ours. It should be noted that DeepRoadMapper uses a post-processing method which relies on training yet another deep neural network to recover the missing segments and connect the gaps in the raw segmentation result. The authors indicate that the training of this second network takes at least a day to reach a good performance score. Thus, our iterative Hough transform method is not only improving the overall performance, but also takes less time to perform the task. All measurements shown in Table 2 are based on the F1 score and IoU metrics.

6 Conclusion

We presented a novel approach for road extraction. Uniquely, the proposed approach leverages cardinality-enabled neural networks with feed forward links in order to achieve high accuracy in the semantic segmentation. The classification result is then further processed using a novel post-processing refinement process which iteratively applies a Hough-transform on a per-patch basis which results in a set of linear segments. The segmented are further refined by connecting nearby segments together and removing erroneous segments resulting from misclassification. We compared our approach with state-of-the-art techniques and we have shown that it can produce on average comparable results and in some cases better. We also compared the post-processing techniques and showed our proposed iterative Hough-transform post-processing method brings significant improvements for semantic segmentation results.

Acknowledgement

This research is based upon work supported by the Natural Sciences and Engineering Research Council of Canada Grants DG-N01670 (Discovery Grant) and DND-N01885 (Collaborative Research and Development with the Department of National Defence Grant).

References

1. Barzohar, M., Cooper, D.B.: Automatic finding of main roads in aerial images by using geometric-stochastic models and estimation. In: Computer Vision and Pattern Recognition, 1993. Proceedings CVPR'93., 1993 IEEE Computer Society Conference on. pp. 459–464. IEEE (1993)
2. Bastani, F., He, S., Abbar, S., Alizadeh, M., Balakrishnan, H., Chawla, S., Madden, S., DeWitt, D.: Roadtracer: Automatic extraction of road networks from aerial images. In: Computer Vision and Pattern Recognition (CVPR) (2018)

3. Buslaev, A., Seferbekov, S., Iglovikov, V., Shvets, A.: Fully convolutional network for automatic road extraction from satellite imagery. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (2018)
4. Cheng, G., Wang, Y., Xu, S., Wang, H., Xiang, S., Pan, C.: Automatic road detection and centerline extraction via cascaded end-to-end convolutional neural network. IEEE Transactions on Geoscience and Remote Sensing **55**(6), 3322–3337 (2017)
5. Forbes, T., Poullis, C.: Deep autoencoders with aggregated residual transformations for urban reconstruction from remote sensing data. In: 2018 15th Conference on Computer and Robot Vision (CRV). pp. 23–30. IEEE (2018)
6. Hinz, S., Baumgartner, A.: Automatic extraction of urban road networks from multi-view aerial imagery. ISPRS Journal of Photogrammetry and Remote Sensing **58**(1-2), 83–98 (2003)
7. Hu, J., Razdan, A., Femiani, J.C., Cui, M., Wonka, P.: Road network extraction and intersection detection from aerial images by tracking road footprints. IEEE Transactions on Geoscience and Remote Sensing **45**(12), 4144–4157 (2007)
8. Li, M., Stein, A., Bijker, W., Zhan, Q.: Region-based urban road extraction from vhr satellite images using binary partition tree. International Journal of Applied Earth Observation and Geoinformation **44**, 217–225 (2016)
9. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3431–3440 (2015)
10. Máttyus, G., Luo, W., Urtasun, R.: Deeproadmapper: Extracting road topology from aerial images. In: The IEEE International Conference on Computer Vision (ICCV) (2017)
11. Mnih, V., Hinton, G.E.: Learning to detect roads in high-resolution aerial images. In: European Conference on Computer Vision. pp. 210–223. Springer (2010)
12. Noh, H., Hong, S., Han, B.: Learning deconvolution network for semantic segmentation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1520–1528 (2015)
13. OpenStreetMap contributors: Planet dump retrieved from https://planet.osm.org . `https://www.openstreetmap.org` (2017)
14. Sherrah, J.: Fully convolutional networks for dense semantic labelling of high-resolution aerial imagery. arXiv preprint arXiv:1606.02585 (2016)
15. Singh, Suriya; Batra, A.P.G.T.L.B.S.P.M.J.C.V.: Self-supervised feature learning for semantic segmentation of overhead imagery. In: BMVC (2018)
16. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1492–1500 (2017)
17. Zhang, Z., Liu, Q., Wang, Y.: Road extraction by deep residual u-net. IEEE Geoscience and Remote Sensing Letters (2018)
18. Zhou, L., Zhang, C., Wu, M.: D-linknet: Linknet with pretrained encoder and dilated convolution for high resolution satellite imagery road extraction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 182–186 (2018)