Rail Network Detection from Aerial Imagery using Deep Learning

Mehrdad Salehi
Apple Inc
msalehi@stanford.edu

Yonghong Wang
Apple Inc
yhwang99@stanford.edu

Abstract

Having an accurate and up-to-date rail network data is the foundation of any mapping application that supports public transportation. Traditionally, creating such a network needs manual digitization of satellite imagery and building the network with human intervention. The goal of this project is to segment the rail network from aerial imagery using Deep Learning. In this project, we study different techniques to classify and segment rail networks in the aerial imagery. We implement ConvNets as well as combination of ConvNets and DeConvNets to classify and segment the rail network. We also share the insights that we derived from analyzing the results.

1. Introduction

Having an accurate and up-to-date rail network data is the foundation of any mapping application that supports public transportation. Traditionally, creating such a network needs manual digitization of satellite imagery and involves human intervention. The goal of this project is to segment the rail network from aerial imagery using Deep Learning. This project has three main contributions: First, we classified images into two classes of rail and non-rail using a network of 7 ConvNets and three fully connected nets; second we did a coarse-level segmentation using ConvNets by reducing dimensionality of images to the label space, we also improved the results dramatically by using ConvNets and DeConvNets, and finally we did a fine-level segmentation by using ConvNets and DeConvNets.

2. Data Preparation

For this project, we needed both aerial imagery and label dataset. Aerial imagery data can be downloaded from providers such as USGS. For label dataset, we used an open source mapping data platform called Open Street Map (OSM) where we obtained labels for our learning algorithm as well as for evaluation.

Figure 1. Example of images, two top images do not have rail track, two bottom images have rail tracks in them

2.1. Aerial Imagery

Aerial imagery should be converted to tiles so that the input data have a uniform size. We followed a standard tiling systems used in the mapping industry. The image size for classification and coarse segmentation are 60x60 meters. For the fine segmentation, we used 30x30 meter images as the data instance. As for the pixel size, in coarse segmentation, each pixel is 3.5x3.5 meters, while for the fine segmentation we used the actual pixel of the image that is about 10cm.

2.2. Label Dataset

In order to prepare the labels we processed OSM rail network for the entire USA. We used Spark, Scala, and JTS and ran our Spark job on a cluster of machines. For the 3.5x3.5 meter tiles, it took about one hour to create the label tiles. This time was longer for 10cm labels, as we used a different technique using Python on a single machine. Figure 1 shows some examples in the dataset.

3. Related Work

We used neural network to classify and segment rail network from satellite images. So we first discuss the work on image classification and segmentation. Image classification is the task of assigning one label to each input image from a fixed set of categories. Krizhevsky et al., in [7] used ConvNets and dropout (5 layer CNNs and 3 layer fully connected layers), and achieved 15.3% top-5 test error rate in ILSVRC 2010 challenge which is significantly better than 26.2% top-5 test error rate by the 2nd place. This marks the turning point for large-scale object recognition, and since then deep neural networks outperforms traditional machine learning methods. Simonyan et al., in [16] further improved the top-5 test error rate to 6.8%. They used deep and narrow CNN filters (14 CNN layers, 3 FC layers). Szegedy et al., in [17] introduced GoogLeNet, a 22 layers deep network with efficient inception module and no FC layers. They achieved 6.7 % top 5 erorr and won 2014 ILSVRC challenge. In 2015, [4], He et al., proposed Residual Network and improved the top-5 test error to 3.57% and surpasses human performance. Using Residual Network, they can use very deep network (152 layers). Zagoruyko et al., in [18] introduced wide residual networks (WRNs) and show that they are far superior than deep residual networks in both efficiency and accuracy. They achieved new state-of-the-art results on CIFAR, SVHN, COCO and significant improvements on ImageNet.

Image segmentation is the task of partitioning image into multiple segments, and each segment belongs to one category. Pohlen et al., in [14] proposed a network architecture for semantic segmentation in street scenes. It used ResNet encoder/decoder architecture and residuals remain at the full input resolution throughout the network. He et al., in [3] introducted Mask R-CNN method to efficiently detect objects in an image and generate high-quality segmentation mask for each object. They extends Faster R-CNN method, [15], by adding a branch for predicting an object mask at the same time. Mask R-CNN outperforms all exsisting, single-model entries on every task, and won COCO 2016 challeng.

Mnih et al., in [13] proposed a large-scale learning approach to detecting roads using a neural network. They used much larger labelled datasets, and introduced a post-processing method to significantly improve the results. Instead of predicting each pixel is road or not, they predict a small block of pixels is road or not, and they are able to get around 87% test accuracy. Acar et al., in [19] using classification algorithms to detect road from satelite images. They used a method consisting of two stages. They first preprocess the image by using greyscale transformation and thresholding process and obtained a binary image, then they use K-nearest neighbours and Naive Bayes classifiers on images by using colour features. They got around 70% to 80 % recall and about 33% to 40 % precision in detecting if 15 x 15 pixel block is part of a road. Kahraman et al., in [6] studied road detection from satellite images using neural networks. The used Multilayer Perceptron (multilayer neural network). For every pixel, they used 27 features from 3 x 3 x 3 pixels around it, and predict if the pixel is part of a road. They were able to get accuracy between 87% and 93% for different settings. Maboudi et al., in [9] introduced a multi-stage object-based approach for road extraction from VHR satellite images. They use Edge-preserving guided filtering to improve the segmentation quality. They also proposed a skeleton based linearity index called SOLI. They have achieved more than 84% accuracy for the datasets they studied. Marmanis et al., in [12], proposed a deep learning approach to semantic segmentation of high resolution aerial images. They used Fully Convolution Networks (FCNs) and aggressive deconvolution with recycling of early network layers, and finally converted into a pixelwise classification at full resolution. They achieved 86% accuracy on their dataset.

They are very few research in detecting and segmenting rail network from satellige images. Maqsood et al., in [11] developed an algorithm to detect railway track from images taken from Google maps. They studied only a small region instead of whole state, country or world. They combined edge detection and line detection algorithms to detect railway.

To make deep neural network work, tuning hyper parameters places a crutial role. Bengio in [1], discussed various techniques to train deep neural network and tune hyper parameters.

4. Aerial Image Classification

The goal of this part of the project was to classify whether a satellite image contains a rail track or not. The classifier should discriminate an image that contains a rail track from other ones. The input image is a 256 x 256 x 3 where each image tile is 60m x 60m. We are interested to get a label for each tile whether it contains a rail track or not.

4.1. Network Architecture

We implemented a 10 layer convolutional neural network with 7 ConvNets and 3 fully connected networks. After each ConvNet (except the last one), we applied a max pooling layer. Also, we applied a batch normalization layer after each layer. Table 1 summarizes the architecture of the network.

4.2. Training and Hyper Parameter Tuning

We trained a network with 7 convolutional layers and 3 fully connected layers (See Table 1). First we tried larger

Layer	Size
Conv 1	3 x 3 x 96
Max-pooling	2 x 2
Conv 2	3 x 3 x 128
Max-pooling	2 x 2
Conv 3	3 x 3 x 192
Max-pooling	2 x 2
Conv 4	3 x 3 x 192
Max-pooling	2 x 2
Conv 5	3 x 3 x 128
Max-pooling	2 x 2
Conv 6	3 x 3 x 128
Max-pooling	2 x 2
Conv 7	3 x 3 x 128
FC 1	128 x 128
FC 2	128 x 64
FC 3	64 x 2

Table 1. Network architecture for image classification

number of channels (e.g., 1024), but training on this network was slow and even crashed because of insufficient memory. Then we reduced the number of channels from 1024 to 192 or 128. This enabled us to train the model. Except for the last one, after each convolutional layer we used a max-pooling layer to reduce the dimensionality. Also, we applied a batch normalization layer after each convolutional layer. The ReLu non-linearity was applied as the activation function.

At the first run, we had a small training dataset consisting of ten thousand images. Using these images we trained the network and obtained 84% accuracy on the test dataset. Then we created more labels of around 100 thousand images and also added learning rate decay to the model with step size 1000 and decay of 0.96. We trained the model with other decay rates and steps sizes (e.g., step size of 10000 and decay 0.1 and also step size 1000 with decay 0.5). Finally, the best accuracy that we obtained on the test set after 1 epoch was 96%. At the end, we increased the number of epochs to 5 and as a result we obtained 98% accuracy on the test set.

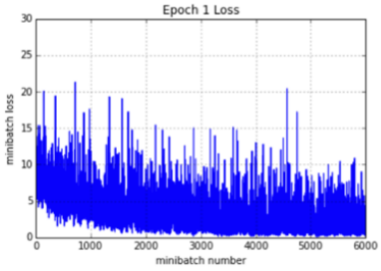
The bottom Figure 2 shows the progress of loss value over one epoch. We used mini batches of size 16. When we increased the size of mini batches to 32, the GPU ran out of memory.

4.3. Analysis of Results and Insights

We obtained 98% test accuracy over a test dataset of size 17000 of balanced images for the classification. The top of Figure 2 show some examples of the classification output.

We observed interesting insights for the cases where the predications do not match the labels. For example, in left

Label: rail, Prediction: Rail Label: no rail, Prediction: no rail



Classification loss over one epoch of 100 thousand images.

Figure 2.

There is an unused rail track in this image covered by grass. This image is labeled as rail, but the classifier does not recognize rail track and predicts it as no rail.

Since the bridge looks very similar to rail tracks, classifier predicts this image as a rail.

Figure 3.

of Figure 3, the image was labeled as rail but the classifier predicts it as no rail. The reason is that the rail track is buried under the grass and only a small part of it is partially visible and it is likely unused. This shows that the imagery and label dataset are not temporally in sync.

In the other example (right of Figure 3), a bridge looks very similar to a rail track and the classifier mistakenly marked it as rail.

5. Course-level Aerial Image Segmentation

The second phase of this project was to segment rail tracks on aerial images where the labels are at lower resolution than the actual pixel size, hence we call it coarse-level segmentation. Similar to classification phase, images are

A 60 x 60 aerial images and the low-resolution labels (3.5 m x 3.5 m). The red tiles show the location of rail tracks.

Figure 4.

Layer	Size
Conv 1	7 x 7 x 96
Max-pooling	2 x 2
Conv 2	5 x 5 x 128
Max-pooling	2 x 2
Conv 3	3 x 3 x 128
Max-pooling	2 x 2
Conv 4	3 x 3 x 256
Max-pooling	2 x 2
Conv 5	3 x 3 x 128
Max-pooling	2 x 2
Conv 6	3 x 3 x64
Max-pooling	2 x 2
Conv 7	1 x 1 x 2

Table 2. ConvNets Network architecture for coarse-level image segmentation

60m x 60m aerial images. However, at this phase our goal was to segment each 3.5m x 3.5m of the image. We used 3.5m as the resolution size because we were able to create large amount of labels at this resolution. Figures 4 shows the resolution of labels (red tiles) for a 60m x 60m images. Each image 60m x 60m includes 16 x16 labels.

5.1. Network Architecture

Since each image has 16 x 16 labels, we defined a convolutional neural network that reduced the dimension of the input image (256 x 256) to the dimension of labels (16 x 16). We defined a network that consists of 7 ConvNets. Table 2 summarizes the network architecture. We also defined a DeConvNets, with 15 layers ConvNets, and 8 layers DeConvNets, each layer followed by Batch Normalization. Input is 256 x 256 x 3, output is 16 x 16 x 2. Table 3 summarizes the network architecture.

Layer	Size	Step
Conv 1	3 x 3 x 32	1
Conv 2	3 x 3 x 64	2
Conv 3	3 x 3 x 64	1
Conv 4	3 x 3 x 128	2
Conv 5	3 x 3 x 128	1
Conv 6	3 x 3 x 128	1
Conv 7	3 x 3 x 256	2
Conv 8	3 x 3 x 256	1
Conv 9	3 x 3 x 256	1
Conv 10	3 x 3 x 256	2
Conv 11	3 x 3 x 256	1
Conv 12	3 x 3 x 256	1
Conv 13	3 x 3 x 256	2
Conv 14	8 x 8 x 2048	1
Conv 15	1 x 1 x 2048	1
DeConv 1	8 x 8 x 256	1 (valid)
DeConv 2	3 x 3 x 256	1
DeConv 3	3 x 3 x 256	1
DeConv 4	3 x 3 x 128	2
DeConv 5	3 x 3 x 64	1
DeConv 6	3 x 3 x 32	1
DeConv 7	3 x 3 x 16	1
DeConv 8	3 x 3 x 2	1

Table 3. ConvNets and DeConvNets Network architecture for coarse-level image segmentation

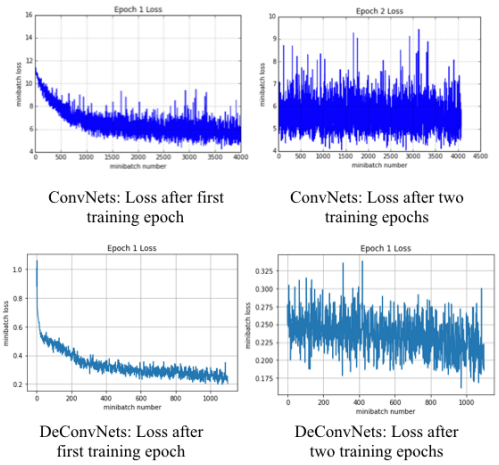
5.2. Training and Hyper Parameter Tuning

We started training the model with the same images that we used to train the classification part (Section 4), but almost all the image were predicted as no rail. This happened because the data was imbalanced (each negative example of classification phase contains no positive pixel but 256 negative ones). Next, we trained using only images that include rail network (removed negative examples). After this update, the results were slightly improved but still we were missing large number of rail tracks. Finally, we included only images that, out of 256 coarse labels, have at least 30 pixels labeled as rail. This change improved the models performance significantly. We trained the network with multiple learning rate decays and step sizes. Eventually the final hyper parameters were learning rate decay of 0.96 and step size of 1000. We used 78000 images for training the network.

We trained both networks for two epochs. Figures 5 shows the training loss after the first and second epochs.

5.3. Analysis of Results and Insights

For the ConvNets model, coarse-level segmentation, we obtained an overall precision of 85%, recall of 46% and 59.7% F1 for the positive examples (rail tracks). For the



ConvNets: Loss after first training epoch

ConvNets: Loss after two training epochs

DeConvNets: Loss after first training epoch

DeConvNets: Loss after two training epochs

Figure 5.

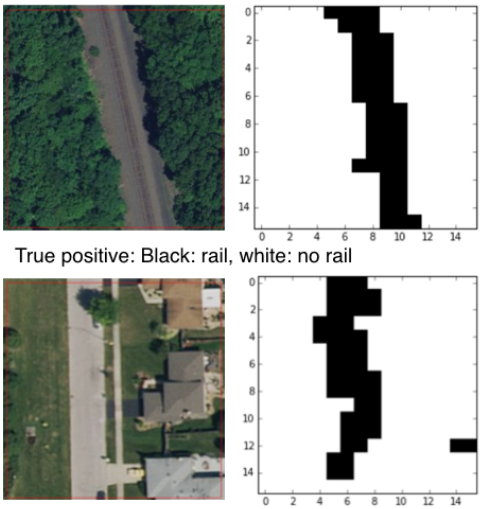
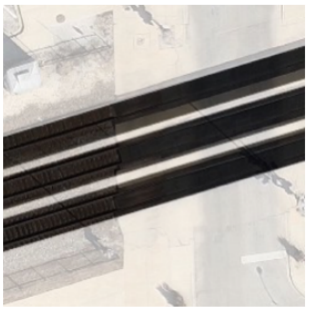


Figure 6.

ConvNets and DeConvNets model, coarse-level segmentation, we obtained an overall precision of 84.%, recall of 77.8% and 81.0% F1 for the positive examples (rail tracks). We observed interesting insights by analyzing the predictions.Figures 6 shows satellite images and predicted labels for them. The classifier sometimes gets confused between road and rail network and segment road as rail (Figures 6). This example shows that we might need to add more negative examples including roads.



Overlapping satellite images and corresponding labels created using OSM rail network data

Figure 7.

Layer	Size
Conv 1	7 x 7 x 32
Max-pooling	2 x 2
Conv 2	5 x 5 x 64
Max-pooling	2 x 2
Conv 3	3 x 3 x 96
Conv 4	1 x 1 x 96
DeConv 1	3 x 3 x 64
DeConv 2	3 x 3 x 32
DeConv 3	3 x 3 x 2

Table 4. Network architecture for fine-level image segmentation

6. Fine-level Aerial Image Segmentation

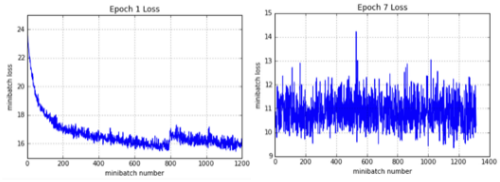
The final phase of this project was to segment rail tracks at individual pixel level. To make the predictions more accurate, we used higher resolution images at this stage where each images covers 30 x 30 meter. We created label dataset by intersecting OSM rail tracks with a 30 x 30 meter grid. Figure 7 shows the aerial image overlapping with the generated label from OSM data.

6.1. Network Architecture

We used a mixture of ConvNet-DeConvNet to segment images. First we reduced the dimensionality of images from 256 x 256 to 32 x 32 using ConNets and then increased dimensionality from 32 x 32 to 256 x 256 using DeConvNet. Table 4 summarizes the architecture of the network.

6.2. Training and Hyper Parameter Tuning

As the first attempt to train the model, instead of using the tick lines covering rail tracks (Figure 7), we used the rail tracks centerline (a thin line). However such a dataset is hugely imbalanced. We tried to fix this by using a weighted softmax cross entroy loss, and tried different weights for each dataset. The results were not promising (we got below



Loss after first training epoch Loss during the 7th epoch

Figure 8.

10% recall). Keeping the same strategy, we increased the amount of training data from 20000 to 40000 and trained for 10 epochs, but the results did not improve.

Finally, we used the training data as tick lines that cover the entire width of rail tracks (see Figure 7). This strategy improved the results so that even after the first epoch the classifier was able to approximately detect rail tracks. We changed the learning rate decay and decay step and eventually chose 0.96 and 1000 respectively. Figure 8 show the loss values during epoch 1 and 7.

6.3. Analysis of Results and Insights

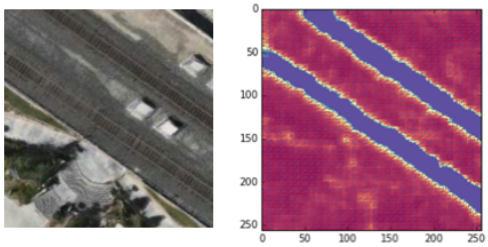
For the fine-level segmentation, we obtained an overall precision of 68% and recall of 70% for the positive examples (rail tracks). By analyzing the prediction results and labels, we were able to reason about the specific pattern that we see in the results.The top of Figure 9 shows an aerial imagery that contains two rail tracks as well as the predicted results. Blue indicates predicted rail tracks and the red highlights no rail pixels. By analyzing the bottom of Figure 9, we can conclude that those parts of rail tracks that were visible in the imagery were detected. However, those parts that are under the bridge or covered by the trees shadow were not detected. In addition, the tiny elevated regions on the bridges median that look like the elevated part of rail tracks are predicted as rail.

7. Future Work

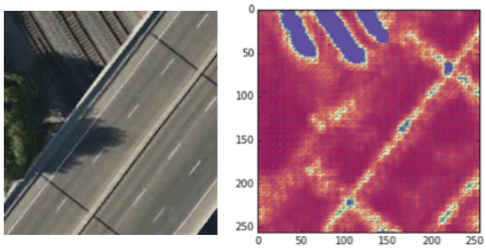
As for the future work, we will use ResNets with DeConvNets, Wide ResNets [18] and Mask R-CNN [3]. We are also interested in training ensembled networks, and then use Dark Knowledge method which was co-invented by Hinton et al., [5], to train a single network based on original labels, and soft labels created by ensembled models.

8. Acknowledgements

We thank Yi Cao for help with providing some labels for each pixel of the satellite images at zoom 20 (30m X 30m). We also use sample codes from Stanford Computer Vision CS231n course [8], NLP course CS224n [10], and referenced sample code in Github, [2]



Input image and probabilities of predicted labels for pixels: Blue: rail, Red: no rail



Input image and predictions: Rail tracks are partially detected because the main part of them are under the bridge. Also, that part of rail tracks that is covered by tree shadows are not detected.

Figure 9.

We are thankful for CS231 teaching stuff for this wonderful class.

References

[1] Y. Bengio. Practical recommendations for gradient-based training of deep architectures. *arXiv*, 2012. arXiv:1206.5533.

[2] fabianbormann. Tensorflow-deconvnet-segmentation. https://github.com/fabianbormann/Tensorflow-DeconvNet-Segmentation/commits/master.

[3] K. He, G. Gkioxari, P. Dolla, and R. Girshick. Mask r-cnn. *arXiv*, 2017. arXiv:1611.08323.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[5] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv*, 2015. arXiv:1503.02531.

[6] I. Kahraman, M. K. Turan, and I. R. Karas. Road detection from high satellite images using neural networks. *International Journal of Modeling and Optimization*, 5(4):304–307, 2015.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. pages 1097–1105, 2012.

[8] F. Li, A. Karpathy, J. Johnson, S. Yeung, and A. TAs. Cs231n: Convolutional neural networks for visual recognition, 2017. http://cs231n.stanford.edu.

[9] M. Maboudi, J. Amini, M. Hahn, and M. Saati. Road network extraction from vhr satellite images using context

aware object feature integration and tensor voting. *Remote Sensing*, 8(637), 2016.

[10] C. Manning, R. Socher, and A. TAs. Cs224n: Natural language processing with deep learning, 2017. https://web.stanford.edu/class/cs224n/.

[11] M. Maqsood, A. Javed, and N. Majeed. A novel algorithm for railway tracks detection using satellite imagery. *International Journal of Computer Applications*, 64(14):13–17, 2013.

[12] D. Marmanis, J. D. Wegner, S. Galliani, K. Schindler, M. Datcu, and U. Stilla. Semantic segmentation of aerial images with an ensemble of cnns. In *ISPRS Annals of Photogramme- try, Remote Sensing and Spatial Information Sciences*, number 3, pages 473–480, 2016.

[13] V. Mnih and G. Hinton. Learning to detect roads in high-resolution aerial images, 2010.

[14] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe. Full-resolution residual networks for semantic segmentation in street scenes. *arXiv*, 2016. arXiv:1703.06870v2.

[15] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv*, 2015. arXiv:1506.01497.

[16] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv*, 2014. arXiv:1409.1556.

[17] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[18] S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv*, 2017. arXiv:1605.07146.

[19] afak Altay Aar and afak Bayr. Road detection using classification algorithms. *Journal of Computers*, 10(3):147–154, 2015.