Satellite Image Segmentation for Building Detection using U-net

Guillaume Chhor, *Computational and Mathematical Engineering,* Cristian Bartolome Aramburu, *Mechanical Engineering,* and Ianis Bougdal-Lambert, *Aeronautics and Astronautics*
{gchhor, cbartolm, ianisbl} [at] stanford.edu

***Abstract*—Automatically detecting buildings from satellite images has a lot of potential applications, from monitoring movements of populations in remote areas to evaluating the available surface to implant solar panels on roofs. We develop a Convolutional Neural Network for the extraction of buildings from satellite images, adapted from a U-net originally developed for biomedical image segmentation. We train our model on satellite images and on ground-truth labels extracted from OpenStreetMap. We show that our model achieves a reasonable level of accuracy, though slightly lower than state-of-the-art, and outline some ideas for further improvements.**

I. Introduction

BEING able to detect new buildings directly from satellite images is especially useful in regions where populations move very rapidly (because they are nomadic, displaced or migrating), as well as in remote and extent areas where the census of these new buildings is often done laboriously by hand and quickly outdated. It can also find an important application in the assessment of building damages following a natural disaster, allowing the formulation of an adequate response in targeted areas. Finally, it can be of great use to solar panels manufacturers who want to evaluate the available roof surface in a particular area.

This project addresses the broader issue of semantic segmentation of satellite images by aiming at classifying each pixel as belonging to a building or not. We developed a Convolutional Neural Network suitable for this task, inspired from the U-net [7]. We trained our model on a set of two-dimensional satellite images. The corresponding labels were binary masks, ie. two-dimensional matrices with ones for pixels where a building was present, zeros otherwise. Given a satellite image as input, our network was then able to output a corresponding predicted binary mask.

This project was also Guillaume's project for his CS230 Deep Learning class. As image segmentation was revolutionized by the use of deep convolutional neural networks and is one of the most complex task in computer vision, this project was also a perfect fit for this class. We had the chance to benefit from the advice and the close supervision of the CS230 teaching staff who helped us solve our issues and find new directions of research.

II. Related Work

Using image segmentation for automatic building detection in satellite images is a pretty recent field of investigation. To that respect, only a few articles have been published on that topic. We can cite [1], [2] and [3]. Essentially, the first two articles use SVM algorithms to extract buildings from very high resolution (VHR) satellite images. The base-case accuracy reached in each case is respectively 74% and 83%. The last paper presents a list of results on the recently released INRIA Aerial Image dataset, using different architectures for image segmentation such as Fully Convolutional Networks (FCN) [4] and SegNet [5]. Contrary to ours, the INRIA dataset consisted of very high resolution images.

The U-net – a specific type of FCN – has received a lot of interest for the segmentation of biomedical images using a reduced dataset, but has proven to be also very efficient for the pixel-wise classification of satellite images [6]. We mainly built upon [7] for the design of our model. In their paper, the authors develop a U-Net specifically dedicated to biomedical image segmentation. Another paper [8] extended the U-net model to the segmentation of 3D medical images, but this was beyond the scope of our project. In both cases, the authors obtained accuracies of about 75%, depending on the test sets.

III. Dataset and Features

A. Data selection and train/dev/test splitting

We followed the advice of Drew Bollinger – a developer at DevelopmentSeed we met at the GISDay@Stanford – and collected our data using MapBox API for OpenStreetMap (OSM). OSM is an open-source mapping platform where anyone can edit maps, adding features such as roads or buildings. We ran the DevelopmentSeed Python API for Mapbox on Docker containers to scrap a dataset of satellite images, along with several corresponding layers (buildings, rivers, roads and woods) with pixel-labeled images for each one of these classes. As we were only interested in the extraction of buildings, we only downloaded the "buildings" layer. We were able to acquire 34547 RGB images of size $256 \times 256 \times 3$ from the territory of France, along with their corresponding RGB building mask.

B. Preprocessing

In order to train our model on a pre-built network, as we will see further on, we first had to resize our images and label masks to a $224 \times 224 \times 3$ size. As the labels we extracted from OSM only consisted of the "buildings" layer, there was no need for an entire RGB channel, so we converted the label images to a binary mask of size $224 \times 224 \times 1$ with 1 for pixels labeled as "buildings", 0 otherwise. Below is one of the satellite images from our dataset, along with its corresponding mask:

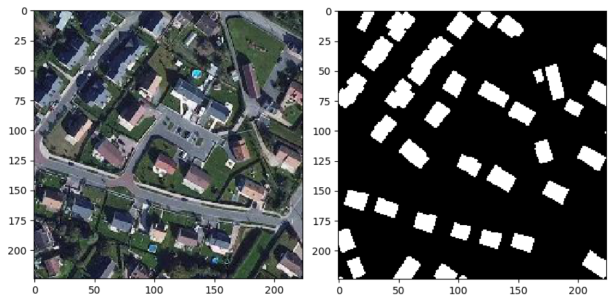


Fig. 1. Satellite image and corresponding mask with buildings identified in white.

One advantage of our dataset is that the images were labeled by humans, resulting in a quite good accuracy. However, a non-negligible proportion of satellite images were not up-to-date with the most recent masks, as new buildings had already been added by users. Some images were also blurry. Overall, this likely decreased the performance of our model – as it had partially learned on mislabeled and/or blurry training images – and led to a suboptimal performance at test time.

Unfortunately, the training was not progressing properly at first, because of the class imbalance. Indeed, most of our images had far more non-building pixels than building pixels. Therefore, we decided to put a threshold of 15% on the minimum ratio of building pixels per image and ended up with a dataset of 3042 images. We split that dataset according to a $70\%/15\%/15\%$ train/dev/test ratio, ie. 2129, 456 and 456 images respectively. This is a relatively small dataset, which is one of the reasons that led us to apply data augmentation, as we will see later. We then centered and normalized our images, and shuffled the dataset.

C. Data Augmentation

The U-net was shown in [7] to work well with only a limited number of training examples, provided one made heavy use of data augmentation. Accordingly, in order to maintain a reasonable amount of images, and above all to avoid overfitting by ensuring a sufficient invariance and robustness of the network, we followed [7] and applied real time data augmentation techniques to our training set. The satellite images were shifted, flipped and rotated, which allowed us to train our model on a considerably larger set of images. This task was done using the Keras framework which allowed us to augment the data in real time when feeding the network with batches. In consequence, there are no memory processes engaged.

IV. Methods

A. Architecture of the model

Instead of developing a model from scratch, we decided to use an existing model of Convolutional Neural Network for image segmentation. Namely, we turned to the U-net, originally developed for biomedical image segmentation [7]. Once trained, the network was able to output a pixel-wise binary classification (building or not) with good accuracy.

Basically, the U-net builds upon the Fully Convolutional Network [4]. A contracting path extracts features of different levels through a sequence of convolutions, ReLU activations and max poolings, allowing to capture the context of each pixel. A symmetric expanding path then upsamples the result to increase the resolution of the detected features. In the U-net architecture, skip-connections (concatenations) are added between the contracting path and the expanding path, allowing precise localization as well as context. The expanding path therefore consists of a sequence of up-convolutions and concatenations with the corresponding feature map from the contracting path, followed by ReLU activations. The number of features is doubled at each level of downsampling. A figure of the U-net taken from [7] is presented below.

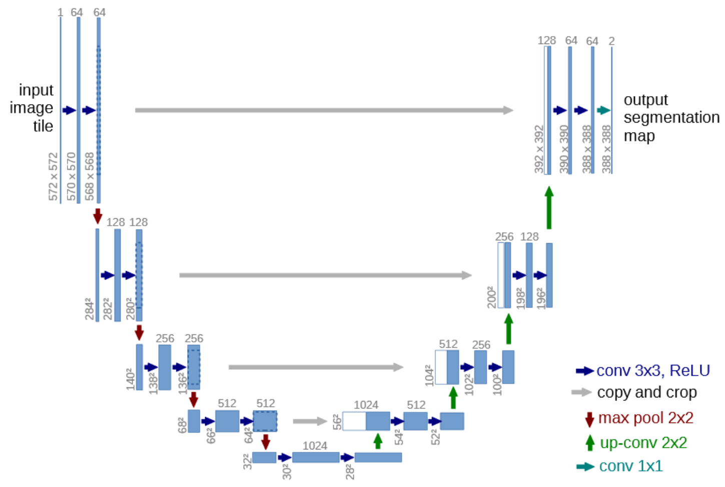


Fig. 2. Original U-net for biomedical image segmentation

B. Modifications

For this building extraction problematic, we chose to use a slightly modified version of the U-Net. We started with a version of the U-net implemented by ZFTurbo [9]. We eventually did not use the pre-trained weights from this source, as they were trained on synthetic data and did not seem to help the training. We thus trained the model from scratch, with a few modifications from the original architecture. First of, we replaced the stochastic gradient descent with the Adam

Optimizer, known to converge faster during training. Then we changed the dimensions of the input images, as the original U-net was designed for images of size 572 × 572 × 3. We also modified the padding to "same" to avoid shrinking when doing convolutions, added batch normalization after each ReLU activation to speed-up training and used a loss based on the Dice coefficient instead of the cross-entropy loss used in [7]. The loss based on the Dice coefficient is commonly used for image segmentation as it allows coping with class imbalance. We did not use dropout, as we did not see any overfitting while training the model, partly thanks to the use of data augmentation. Finally, to ease optimization and tackle vanishing gradient, we decided to get rid of the last downsampling layer of depth 1024.

V. Experiments/Results/Discussion

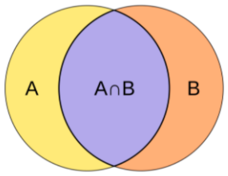
A. Metrics

As mentioned above, the metric used to evaluate the score of our training was the Dice Coefficient, also known as F_1 score:

$$Dice = \frac{2 \times |A \cap B|}{|A| + |B|} = 2 \times \frac{precision \times recall}{precision + recall} \tag{1}$$

where A is the ground truth and B the predicted label. It is very similar to the Jaccard Index, also known as IOU (Intersection over Union):

$$Jaccard = \frac{|A \cap B|}{|A \cup B|} \tag{2}$$



The loss was set to the opposite of the dice coefficient

$$Loss = -Dice \tag{3}$$

B. Training

The training was carried out on Floydhub GPUs (Tesla K80). Considering the memory available on the GPU, we did a training on mini-batches of 32 images. We went through approximately 120 epochs over a total of 6 hours. We started from a learning rate of 0.001. Each time a plateau was reached for more than 5 epochs (ie. each time the loss did not decrease for 5 epochs), we reduced the learning rate by 50%, using the Keras callback ReduceLROnPlateau. We considered the training to be complete when no progress was seen for more than 20 epochs. We can see below that there is no overfitting – partly thanks to data augmentation – as we converge towards a Dice coefficient of approximately 0.75 for both the training set and the development set.

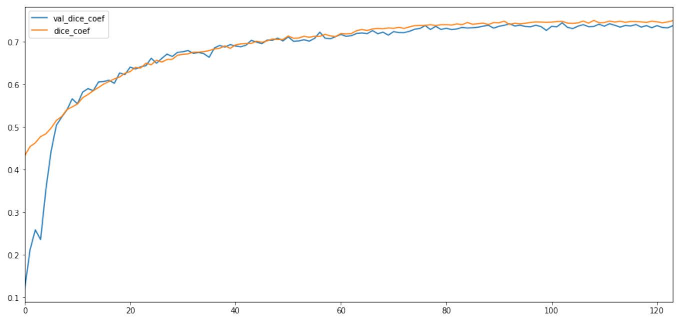
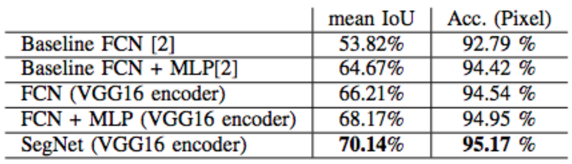


Fig. 3. Learning curve of our model during training. We plotted the dice coefficient evaluated on our training set, as well as the dice coefficient evaluated on our development set.

Before achieving a successful training, we went through many issues such as vanishing gradients and an iteration loop training/testing/tuning that was too slow. For these reasons we chose to simplify the model by removing the 1024-deep layer. The slow rate of convergence was also the reason why we had to fine-tune the learning rate while training, why we investigated good-practices such as putting the Batch Normalization layer after, instead of before, the ReLu activation, and most importantly why we put a threshold on the density of building pixels.

C. Testing and Results

Evaluating our trained model on the test set, we obtained a Dice coefficient of 0.75, and a Jaccard coefficient of 0.60. Few results are available in the literature to compare our results and we should note that the score obtained is highly dependent on the resolution of the images. However, we did find the following results [3], obtained with high resolution images provided by the INRIA [11]. The images had a resolution of 0.3m, while we estimate the resolution of our images to be at best 0.5m.



	mean IoU	Acc. (Pixel)
Baseline FCN [2]	53.82%	92.79 %
Baseline FCN + MLP[2]	64.67%	94.42 %
FCN (VGG16 encoder)	66.21%	94.54 %
FCN + MLP (VGG16 encoder)	68.17%	94.95 %
SegNet (VGG16 encoder)	**70.14%**	**95.17 %**

Fig. 4. State-of-the-art results on a dataset provided by the INRIA, using different methods.

We present below the predictions outputted by our model on some testing data:

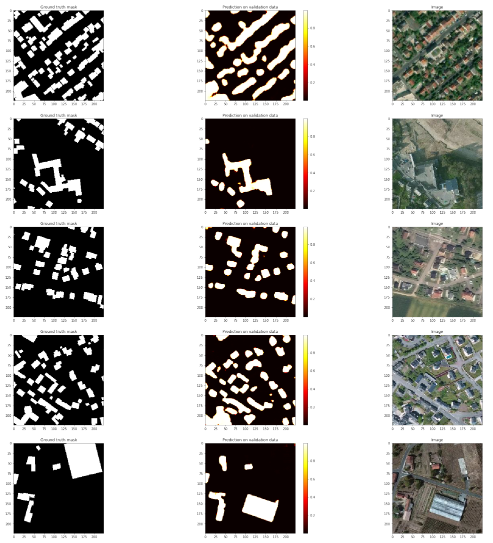


Fig. 5. Ground-truth masks, predicted masks and satellite images from our test set.

D. Discussion

We see from the results above that we achieve a precision slightly inferior to state-of-the art results using convolutional neural networks, but still reasonable (mean IOU of 0.60 vs. 0.70 in literature). We note however that our U-net performs better than the baseline FCN upon which it is built, which is encouraging.

Starting from these results, we tried to understand where the loss of accuracy originated, in order to try to improve our model. We first suspected the ratio of building pixels per images to be in cause, but we found no notable correlation between that ratio and the value of the Dice coefficient. We therefore decided to focus on the test examples that were labeled with the lowest accuracy. Bellow are some of them:

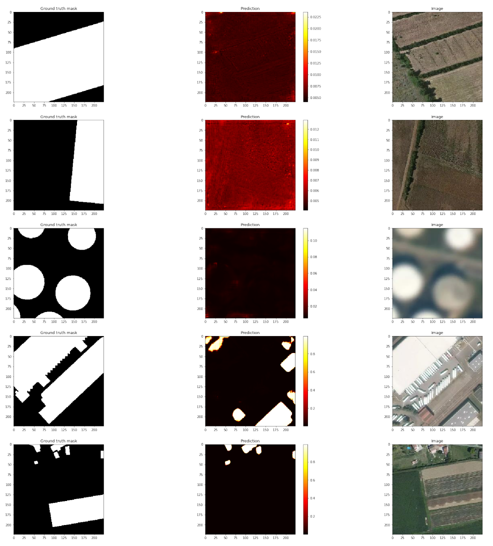


Fig. 6. Ground-truth masks, predicted masks and satellite images for some of the worst test examples.

We see that our network mainly fails on mislabeled or blurry data. Often, either the masks are wrongly defined by the users of OSM or the satellite images are not up to date. This points to an interesting application of our method: detecting those kind of errors on open-source mapping platforms like OSM to correct the layers, or update satellite images where it is more needed. However, we see by looking at the results that even for correctly classified images, there is room for improvement, notably when it comes to boundary detection.

E. Improving boundary detection

The main weakness of our model is that it fails at detecting building boundaries with great precision. It has a propensity to give us soft edges, instead of sharp and well defined ones. This is especially an issue in densely populated areas where buildings are close:

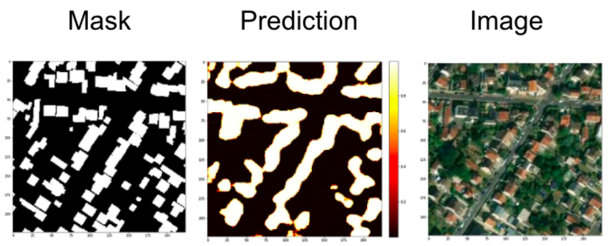


Fig. 7. Ground-truth mask, predicted mask and satellite image in a densely populated area. The algorithm is unable to make a clear separation between close houses.

We tried to find a way to improve the accuracy of classification on boundaries. The idea is to weight the inner and outer borders of the buildings. Looking at the formula of the Dice coefficient, by weighting negatively the pixel on the outer border of the building instead of just zero, we will penalize the loss strongly if the model outputs a non-zero probability on these pixels. On the contrary, we want the model to be rewarded if it classifies accurately the inner border of a building, so we weight positively the inner border (if the model outputs a high probability for this pixel, the loss will notably decrease). That last modification allows us to roughly compensate the pixels weighted negatively in the denominator.

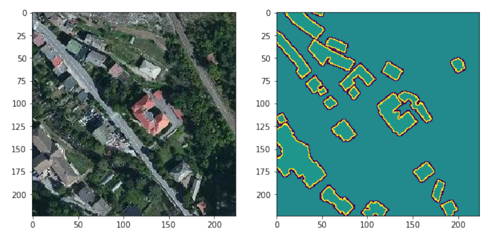


Fig. 8. The outer boundaries of buildings are weighted negatively (purple), while the inner boundaries are weighted positively (yellow).

We were unable however to quantify that improvement. While it was clear that the output was more accurate than before in terms of boundary definition, this was not reflected on the value of the dice coefficient, which actually turned out to be slightly lower. Testing the improved model on the test set, we obtained a Dice coefficient of 0.74, and a Jaccard coefficient of 0.59, when we had 0.60 previously. We present below the effect of the improvement on a few images:

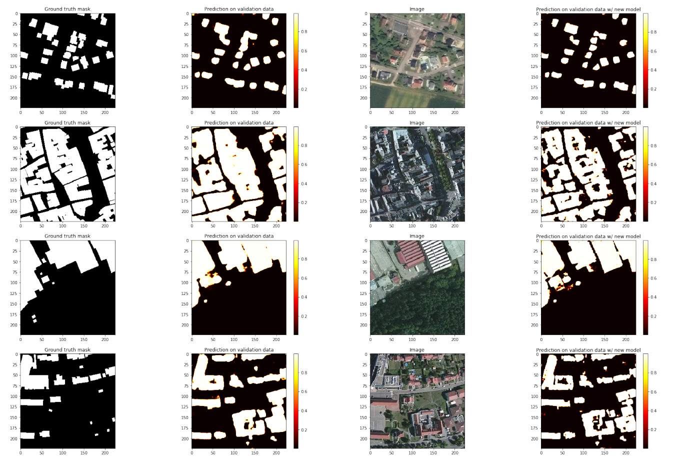

Ground truth mask
Prediction on validation data
Image
Prediction on validation data w/ new model

Fig. 9. Ground-truth, original prediction, satellite image and improved prediction. There is a clear visual improvement in boundary detection that is not reflected on the dice coefficient.

There is actually an explanation to what seems to be a paradox. If we look into the Dice coefficient, it is possible to show that a model with a loss defined by that metric would rather encompass the building pixels by a slight margin rather than not detect them. This results in undefined boundaries when the buildings are close to each other and ”blobby” predictions. It so appears that the Dice coefficient might not be a good index of performance for training a model on a segmentation task if we want to output precise boundary definitions.

VI. Future Work

From what precedes, a first thing to do would be to implement a better loss, that could take into consideration the precision on the boundary along with the accuracy on the segmentation task, and output a better IOU. This was seemingly studied by the authors of this paper [3], from which we could take inspiration.

To address the problem of precisely detecting boundaries, we could also think of only training our model to extract building boundaries, instead of the whole buildings. However this is of lower interest in terms of potential applications, and it is not sure that the algorithm would be as efficient, as it might end up detecting all straight lines in a given image. For these reasons, a more sophisticated strategy would be needed to overcome the problem of ”blobby” predictions and particularly, the one presented in [3] would be suitable for that. Bischer et. al. propose to use multi-task learning to not only rely on the original semantic term, but also on a geometric term that would incorporate the boundary information into a single loss function. The results they present show that the use of that new loss achieves the best results per location and overall, on both evaluation metrics.

Apart from solving the geometric inaccuracies of the predictions, we would also like to test the performance of the U-net on the dataset recently released by the INRIA [11]. That would allow us to evaluate the influence of image resolution on the score, and to have a more relevant comparison to the state-of-the-art results shown above. Conversely, we could evaluate other types of architectures (SegNet, One Hundred Layers Tiramisu, ...) on our own dataset.

Finally, with the aim of building a scalable predictive system which could be trained with a very large training set, we considered taking advantage of the computational resources offered by Google Cloud. We plan on using an automation system called ClusterJob (CJ), in combination with Elasticluster, to run the experiments in a self-designed SLURM cluster in the Cloud. CJ builds reproducible computational packages that are easy to reuse and share with others [10]. On the other hand, Elasticluster is a command line tool that facilitates the creation, management and setup of computing clusters hosted on cloud infrastructures like Amazon’s Elastic Compute Cloud EC2 or Google Compute Engine.

VII. Individual Contributions

During the course of this project we mainly worked together to discuss data acquisition, data pipeline, choice of architecture, implementation,... Cristian focused specifically on the use of TensorFlow and on finding good sources of satellite images. Ianis investigated the possibility of exporting data from different APIs - especially Earth Engine and Mapbox. Guillaume concentrated on finding the most suitable U-net model to import and on implementing it on Python.

VIII. Conclusion

In this project, we examined an end-to-end approach for semantic segmentation of satellite images for building detection using few data, and with a relatively low resolution. We implemented a CNN based on the U-net architecture developed by Ronneberger et al. in [7] and used the MapBox API in OpenStreetMap to collect the datasets of interest. We used the high-level python API Keras to implement our model and facilitate data augmentation in order to improve its robustness. Our proposed approach achieved a reasonable accuracy, though slightly lower than state-of-the-art results published in the literature using other CNN architectures, and without post-processing. We then proposed a way to tackle the imprecise semantic segmentation of boundaries outputted by our model. Building upon this work and with the aim of preserving semantic boundaries, we plan to extend our network with further geometric cues and a uncertainty weighted multi-task loss, inspired by [3].

Acknowledgment

The authors would like to thank Drew Bollinger from DevelopmentSeed who helped us acquire our dataset, and Kian Katan, Ramtin Keramati & Younes Mourri, the teaching staff of CS230, for their precious help.

References

[1] M. Vakalopoulou, K. Karantzalos, N. Komodakis, N. Paragios, *Building Detection in Very High Resolution Multispectral Data with Deep Learning Features*, Nov 2015

[2] O. Benarchid and N. Raissouni, *Support Vector Machines for Object Based Building Extraction in Suburban Area using Very High Resolution Satellite Images, a Case Study: Tetuan, Morocco*, IAES International Journal of Artificial Intelligence (IJ-AI), Vol. 2, No. 1, March 2013, pp. 43 50.

[3] Benjamin Bischke, Patrick Helber, Joachim Folz, Damian Borth, Andreas Dengel, *Multi-Task Learning for Segmentation of Building Footprints with Deep Neural Networks*, arXiv:1709.05932v1 [cs.CV], September 2017.

[4] J. Long, E. Shelhamer and T. Darrell, *Fully Convolutional Networks for Semantic Segmentation*, University of Berkeley, Proceedings of the IEEE, 2015.

[5] V. Badrinarayanan, A. Kendall, R. Cipolla, *SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation*

[6] https://www.kaggle.com/c/dstl-satellite-imagery-feature-detection

[7] O. Ronneberger, P. Fischer, T. Brox, *U-Net: Convolutional Networks for Biomedical Image Segmentation*, Computer Science Department and BIOSS Centre for Biological Signalling Studies, University of Freiburg, Germany, 2015.

[8] J. Patravali, S. Jain, S. Chilamkurthy, *2D-3D Fully Convolutional Neural Networks for Cardiac MR Segmentation*, Qure.AI, 2017.

[9] ZFTurbo, https://github.com/ZFTurbo

[10] Hatef Monajemi, David L. Donoho and Victoria Stodden, *Making Massive Computational Experiments Painlessly*, Proc. of IEEE Intl. Conf. on Big Data, Dec 2016.

[11] INRIA Aerial Image Labeling Dataset, https://project.inria.fr/aerialimagelabeling/