Journal of Computer and Communications, 2022, 10, 132-138
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

Using U-Net to Detect Buildings in Satellite Images

Eric Wang[1], Dali Wang[2]

[1]Department of Computer Science, Stanford University, Stanford, USA
[2]Environmental Sciences Division, Oak Ridge National Laboratory, Oak Ridge, USA
Email: ericw553@stanford.edu, wangd@ornl.gov

How to cite this paper: Wang, E. and Wang, D. (2022) Using U-Net to Detect Buildings in Satellite Images. *Journal of Computer and Communications*, **10**, 132-138.
https://doi.org/10.4236/jcc.2022.106011

Received: May 2, 2022
Accepted: June 27, 2022
Published: June 30, 2022

Abstract

This report presented a method that uses deep computing and stochastic gradient descent algorithm to automatically detect building from satellite images. In this method, a convolutional neural network architecture called U-Net was trained to highlight the building pixels from the rest of the image. This method applied a binary cross-entropy loss function, used ADAM algorithm for gradient descent optimization, and adopted interaction-over-union for accuracy measurement. Continuous loss decreases and accuracy increases were observed during the training and validation. Finally, the visualization of the predicted masks from the trained model after 20 epochs proved that the U-Net model delivers over 60% Intersection over Union accuracy results for detecting buildings from satellite images.

Keywords

U-Net, Satellite Images, Computer Vision, Object Detection

1. Introduction

Satellite imagery has a myriad of uses in a variety of different technical fields, such as meteorology, oceanography, fishing, agriculture, regional planning, education, intelligence and warfare. This study uses satellite images for building detection. There are several reasons to focus on the topic of satellite imagery and its relationship to building representation: 1) it can help us to plan out what new buildings can go where, and how each current building fits in an ecosystem of other surrounding buildings, 2) it can be applied to a city building simulation, where I can have a holistic visualization of each section of the city, to find ways to improve on the current infrastructure, and 3) it can help with risk detection

and management (*i.e.* natural disaster planning).

There is an expansive amount of literature on how to detect buildings from satellite imagery using traditional approaches. [1] [2] developed feature-based approaches to characterize and detect buildings. [3] presented a region-based technique for building detection. [4] proposed a model to compute the contours of buildings. Additionally, people have applied deep computing technologies for building detection very recently. [5] proposed a scheme with guided filters for efficient building detection from satellite images using deep learning. [6] presented a method for automatic airport detection in remote sensing images using convolutional neural networks. Finally, [7] presented a method that used R-CNN network methods for building detection in remote sensing images.

2. Method

2.1. Satellite Data and Mask Generation

Most satellite images come with a special data format (such as GEOTIFF and HDF [8] [9]), and require special knowledge as well as a geospatial library (such as GDAL [10]) to process it. In order to help accomplish this task without extenal software dependecy, we first converted the original data from GEOTIFF into the common numpy format with the channel first option. The resulting dataset contains 16 images with a core area of 625 × 625 pixels, as well as an additional 92 padding pixels outside of the core box, making the total dimension a 809 × 809 pixel image. Each image contains 4 channels: the cyan, magenta, yellow, and key (black) color maps. Furthermore, binary masks for these images (where 1 represents a building pixel, and 0 represents anything else) were manually created. These images and masks were randomly split into three parts: 12 images/masks grouped as the training dataset, 2 images/masks as the validation dataset, and the final 2 images/masks as the testing dataset. An example of the input image and its corresponding masks are illustrated in **Figure 1**.

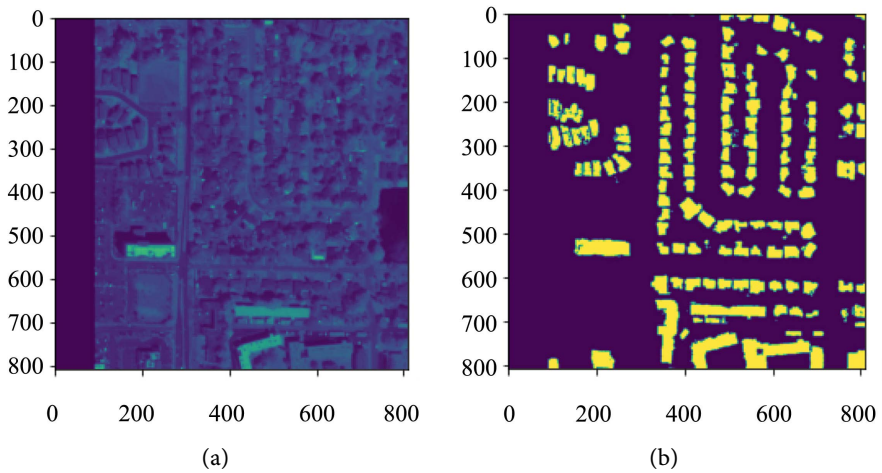

Figure 1. An example of a training images and its building mask. (a) A training image example; (b) a ground-truth mask for the training image.

2.2. Building Detection with U-Net

This study adopted a special network, called U-Net, to detect buildings. In this section, the U-Net architecture was briefly introduced and was followed by an illustration of the general workflow for training, validation, and testing. Several technical components, such loss functions, accuracy measurements, and optimization algorithms were also explained in this section.

2.2.1. U-Net Model Architecture

Our primary segmentation tool, U-Net [11] is an architecture originally used for biomedical image segmentation, but has become the most popular architecture for many type of semantic segmentation. It is composed of many of what is known as a skip connection, a connection from the early parts of the network to its later parts, with information being transferred over. This allows us to bring over lost information that was previously located in the early layers, and it allows us to get a better idea of the network as a whole. It assigns each skip connection from the first convolution to the last, and meeting in the middle, giving the architecture its signature "U" shape. In this study, a PyTorch implementation of U-Net is adopted [12].

2.2.2. Workflow of Building Detection

The workflow chart of this study was presented in the the following **Figure 2**. At first, a U-Net model was randomly initialized to take the input images from the training dataset and produce a series of masks. Then, these masks are compared with the labelled masks for loss function calculation. Next, these loss functions are used in the backpropagation procedure to adjust the weights of U-Net with a gradient decent algorithm. During the training process, the intermediate training result of the U-Net was used on the images in the validation dataset to produce validation masks. These masks are then compared with associated masks to show the model prediction accuracy. After training and validation, the final U-Net

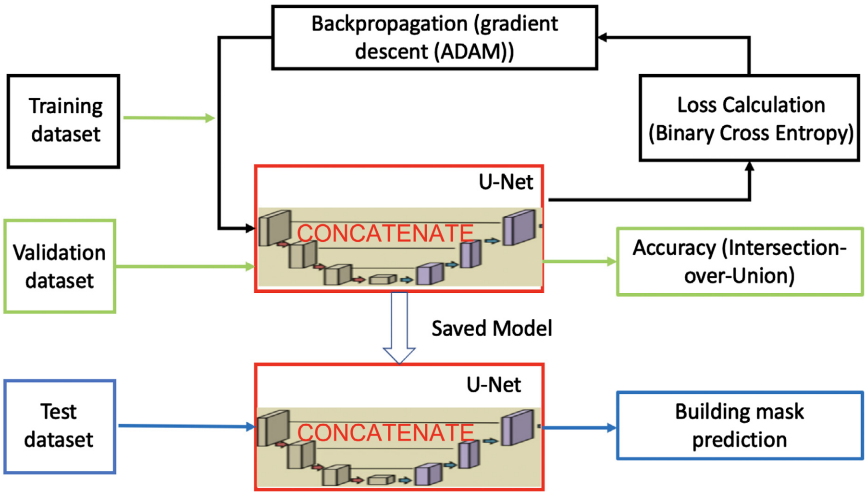

Figure 2. Workflow of building detection with U-Net architecture.

model was saved into disk and a test utility was created to load the model, take the images of testing dataset, and produce masks that show building pixels.

2.3. Loss Function

Our indicator for loss during this project was Binary Cross Entropy [13], or BCE for short. It is a loss function used to classify binary (yes/no, A/B, 0/1) tasks. As such, it is represented by Loss Equation (1):

$$-\frac{1}{N}\sum_{i=1}^{N} y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i) \tag{1}$$

where $\hat{y}_i$ and y_i are the output scalar value and the target value of the ith term, respectively, and N is the number of scalars within the output data for the model. Our purpose for using BCE is to allow quick processing and classification of our training examples, as it is equivalent to maximum likelihood estimation fitting, guaranteeing consistency and statistical efficiency.

2.4. Accuracy Measurement

For accuracy prediction, the Intersection over Union (IoU) metric, a 0 (least overlap) to 1 (most overlap) scale metric, was used to determine the amount of similarity between the predicted masks and the ground truth masks. **Figure 3** illustrates an IoU as the ratio of the two bounds' overlap over the total areas of the two bounds.

2.5. Optimization Procedure

Our method adopted Adaptive Moment Estimation, or ADAM, for stochastic gradient descent optimization. ADAM optimization relies on the first and second moment of gradient to update its learning rates. It has an increased cost, due to

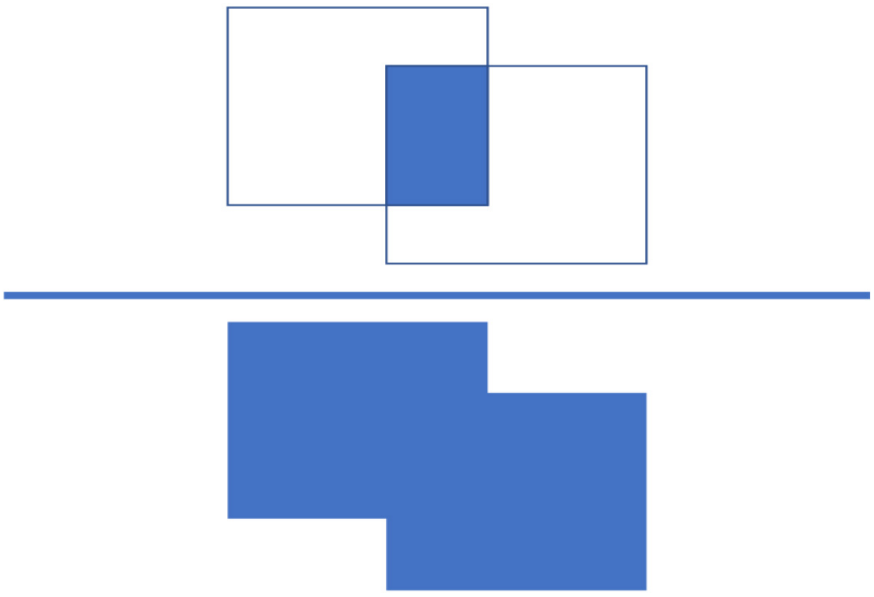

Figure 3. Illustration of Intersection-over-Union.

requiring the calculation of the second derivative, but with the added benefit of converging in circumstances that standard gradient descent may not, as it is invariant to gradient rescaling.

Mathematically, by defining the estimates of m_t and v_t as the mean and the uncentered variance of the gradients of current mini-batch g_t, respectively, as well as the decay rates as β_1 and β_2, ADAM can be represented as following Equations (2):

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \end{aligned} \tag{2}$$

This gives a decaying average for both that allows the gradient descent to proceed and eliminate more jagged routes to the intended destination.

3. Experiment and Result

3.1. Software Packages and Computer Configuration

Python 3.8 is the main program environment. Several packages were added, including PyTorch, Click, and NumPy for code development, as well as matplotlib for visualizations. PyCharm and Jupiter Notebook were crucial for debugging and arduously testing the code. All the code development and experiments were conducted on a 2020 16' MacBook Pro with an 8-core i9 Processor and 16GB DDR4 Memory.

3.2. Training and Validation Results

After running through 20 epochs (30 seconds per image/2.5 hours for the entire training) with the training data, a few insights were discovered as to the loss and accuracy of the masks as a whole.

The average of the 12 BCE values at each epoch was used to determine the training loss over time. There was a significant decreasing negative rate over time, as shown in the first graph of **Figure 4**.

As to the Validation Accuracy over time, the average of the 2 IoU values at each epoch were use to measure our prediction. Due to the high variance of the

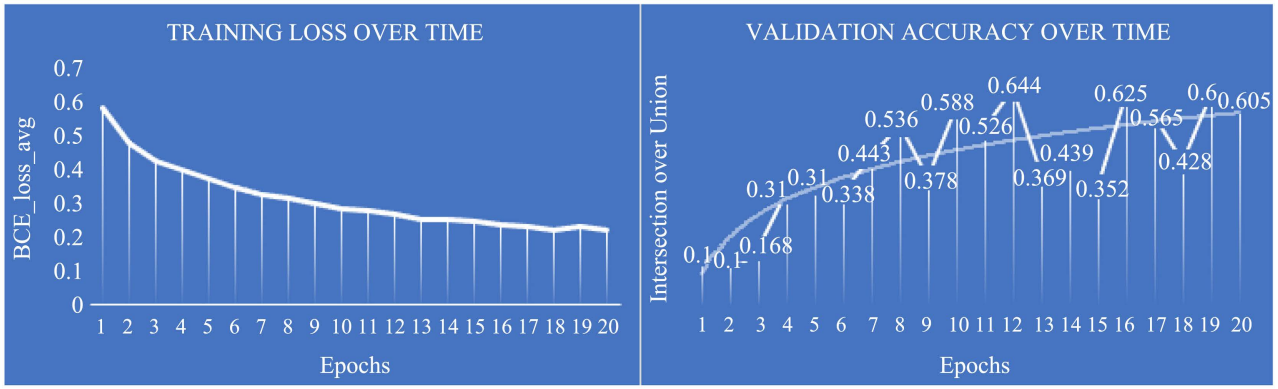

Figure 4. The performance of U-Net training and validation.

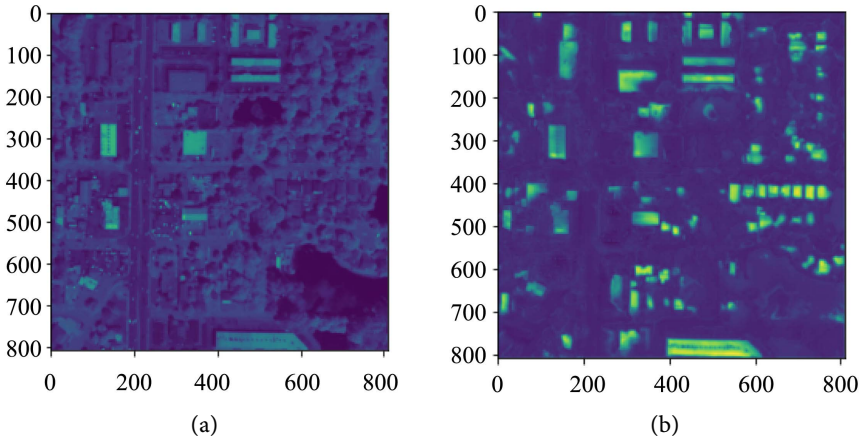

Figure 5. An example of a final test result from the model. (a) A test image example; (b) a predicted mask from test image.

data, a logarithmic trend line was plotted to show the general increase of the accuracy over time (Figure 4, second graph).

After training and validation, the final model was saved to a separate folder, called saved models.

3.3. Testing Result

Through the previous training and validation, we were able to get a well trained model to run on the test images. Figure 5 shows the result of the model on a test image. The general result seems to accurately find the buildings, leading us to conclude that our trained model was a good fit.

4. Discussion and Future Work

In this project, a U-shaped convolutional network was used to detect building pixels from satellite images with the help of the Python Pytorch package on a laptop computer. A common optimization algorithm (*i.e.* ADAM) was used for training with a fixed learning rate, and the Intersection-over-Union index was used to measure the accuracy. The result is generally satisfactory, but there is certainly more room to improve, especially as the IoU data tended to still fluctuate largely even after 20 epochs. Currently, it takes very long time to train the model, and most importantly, the training process drained my computer battery rapidly even as it was connected to the power source the entire time. For future work, I would like to look into the use of a graphics processing unit (GPU) to accelerate the deep learning calculation. I will also look to explore the impact of different learning rates on model prediction accuracy.

5. Data and Code Availability

The code related to this report is publicly available in GitHub at https://github.com/Ericw553/sat_detect. All the training, validation, and testing data is also available per request by email.

Acknowledgements

Sincere thanks to the members of JAMP for their professional performance, and special thanks to managing editor Hellen XU for a rare attitude of high quality.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Weidner, U. and Förstner, W. (1995) Towards Automatic Building Extraction from High-Resolution Digital Elevation Models. *ISPRS Journal of Photogrammetry and Remote Sensing*, **50**, 38-49. https://doi.org/10.1016/0924-2716(95)98236-S

[2] Liu, W. and Prinet, V. (2005) Building Detection from High-Resolution Satellite Image Using Probability Model. *Proceedings of* 2005 *IEEE International Geoscience and Remote Sensing Symposium*, **6**, 3888-3891.

[3] Cui, S., Yan, Q., Liu, Z. and Li, M. (2008) Building Detection and Recognition from High Resolution Remotely Sensed Imagery. *Proceedings of the XXIst ISPRS Congress*, **37**, 411-416.

[4] Theng, L.B. (2006) Automatic Building Extraction from Satellite Imagery. *Engineering Letters*, **13**, 255-259.

[5] Xu, Y., Wu, L., Xie, Z. and Chen, Z. (2018) Building Extraction in Very High Resolution Remote Sensing Imagery Using Deep Learning and Guided Letters. *Remote Sensing*, **10**, 144. https://doi.org/10.3390/rs10010144

[6] Chen, F., Ren, R., Van de Voorde, T., Xu, W., Zhou, G. and Zhou, Y. (2018) Fast Automatic Airport Detection in Remote Sensing Images Using Convolutional Neural Networks. *Remote Sensing*, **10**, 443. https://doi.org/10.3390/rs10030443

[7] Bai, T., Pang, Y., Wang, J., Han, K., Luo, J., Wang, H., Lin, J., Wu, J. and Zhang, H. (2020) An Optimized Faster r-cnn Method Based on Drnet and Roi Align for Building Detection in Remote Sensing Images. *Remote Sensing*, **12**, 762. https://doi.org/10.3390/rs12050762

[8] Ritter, N., Ruth, M., Grissom, B.B., Galang, G., Haller, J., Stephenson, G., Covington, S., Nagy, T., Moyers, J., Stickley, J., *et al.* (2000) Geotiformat Specication Geotirevision 1.0. *SPOT Image Corp*, **1**, 154-172.

[9] Folk, M., Heber, G., Koziol, Q., Pourmal, E. and Robinson, D. (2011) An Overview of the HDF5 Technology Suite and Its Applications. Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases, pp. 36-47.

[10] Warmerdam, F. (2008) The Geospatial Data Abstraction Library. In: Hall, G.B. and Leahy, M.G., Eds., *Open Source Approaches in Spatial Data Handling*, Springer, 87-104. https://doi.org/10.1007/978-3-540-74831-1_5

[11] Ronneberger, O., Fischer, P. and Brox, T. (2015) U-Net: Convolutional Networks for Biomedical Image Segmentation. *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Springer, Cham, 234-241.

[12] Charles, P. (2018) Unet: Semantic Segmentation with Pytorch. https://github.com/milesial/Pytorch-UNet

[13] Rubinstein, R.Y. and Kroese, D.P. (2004) The Cross-Entropy Method: A United Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning. Springer, Berlin.