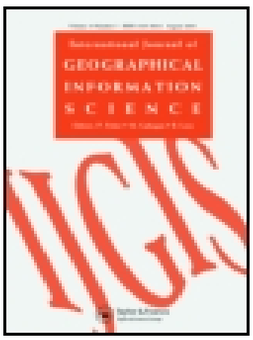

International Journal of Geographical Information Science

ISSN: 1365-8816 (Print) 1362-3087 (Online) Journal homepage: https://www.tandfonline.com/loi/tgis20

A locally-constrained YOLO framework for detecting small and densely-distributed building footprints

Yiqun Xie, Jiannan Cai, Rahul Bhojwani, Shashi Shekhar & Joseph Knight

To cite this article: Yiqun Xie, Jiannan Cai, Rahul Bhojwani, Shashi Shekhar & Joseph Knight (2019): A locally-constrained YOLO framework for detecting small and densely-distributed building footprints, International Journal of Geographical Information Science, DOI: 10.1080/13658816.2019.1624761

To link to this article: https://doi.org/10.1080/13658816.2019.1624761

INTERNATIONAL JOURNAL OF GEOGRAPHICAL INFORMATION SCIENCE
https://doi.org/10.1080/13658816.2019.1624761

Taylor & Francis
Taylor & Francis Group

RESEARCH ARTICLE

A locally-constrained YOLO framework for detecting small and densely-distributed building footprints

Yiqun Xie[a], Jiannan Cai[a], Rahul Bhojwani[a], Shashi Shekhar[a] and Joseph Knight[b,c]

[a]Department of Computer Science and Engineering, University of Minnesota - Twin Cities, Minneapolis, MN, USA; [b]Department of Forest Resources, University of Minnesota-Twin Cities, St. Paul, MN, USA; [c]Remote Sensing and Geospatial Analysis Laboratory, University of Minnesota - Twin Cities, St. Paul, MN, USA

ABSTRACT

Building footprints are among the most predominant features in urban areas, and provide valuable information for urban planning, solar energy suitability analysis, etc. We aim to automatically and rapidly identify building footprints by leveraging deep learning techniques and the increased availability of remote sensing datasets at high spatial resolution. The task is computationally challenging due to the use of large training datasets and large number of parameters. In related work, You-Only-Look-Once (YOLO) is a state-of-the-art deep learning framework for object detection. However, YOLO is limited in its capacity to identify small objects that appear in groups, which is the case for building footprints. We propose a LOcally-COnstrained (LOCO) You-Only-Look-Once framework to detect small and densely-distributed building footprints. LOCO is a variant of YOLO. Its layer architecture is determined by the spatial characteristics of building footprints and it uses a constrained regression modeling to improve the robustness of building size predictions. We also present an invariant augmentation based voting scheme to further improve the precision in the prediction phase. Experiments show that LOCO can greatly improve the solution quality of building detection compared to related work.

ARTICLE HISTORY
Received 29 April 2018
Accepted 23 May 2019

KEYWORDS
Building detection; deep learning; locally constrained; YOLO; remote sensing

1. Introduction

Building footprints are among the most visually conspicuous features in urbanized areas, and they provide valuable information to a variety of domains such as urban planning and city management (Chau *et al*. 2007, Kubota *et al*. 2008, Joshi and Kono 2009), census (e.g. population estimation) Wu et al. 2018, solar energy suitability analysis (Rylatt *et al*. 2001, Compagnon 2004), environmental science (Locke *et al*. 2014), etc. For example, distribution of buildings can help urban researchers or city administrators monitor urbanization and population expansion, or help environmental scientists to study the physical conditions (e.g. air flow, heat accumulation) of different city zones. Coverage of building rooftops can also help energy specialists to estimate the current solar potential in the areas of interest. In face of natural disasters or emergencies, up-to-date building catalogs are important for decision makers to accurately locate the affected people and

CONTACT Yiqun Xie xiexx347@umn.edu

properties. Despite the valuable information contained in building footprint data, many cities in the US and around the globe still do not have a good and up-to-date coverage of buildings in their data catalogs.

With advancements in remote sensing techniques, satellite imagery (e.g. visual bands) has been collected at broad scales (e.g. national level). Moreover, those datasets are becoming available at high spatial resolution and temporal frequency. For example, the National Agriculture Imagery Program (NAIP) (National Agriculture Imagery Program 2018) at the US Farm Service Agency provides national-scale one-meter resolution imagery on an annual basis. The availability of such broad-scale datasets opens up a great opportunity for detecting building footprints and making timely updates of existing building footprint catalogs.

Given a collection of satellite imagery (visual bands) in a set S of spatial regions (e.g. zones in several cities), and existing building footprint data in a subset $S' \subset S$, we aim to automatically identify building footprints in $S - S'$ using a deep learning model trained from the data in S'. The two objectives are solution quality (e.g. precision and recall) and computational efficiency. The constraint is that the spatial resolution of the satellite imagery must be sufficiently high (e.g. two meters) in order to maintain feasibility.

Detection of building footprints has three major challenges. First, it is difficult to tell the class of a single pixel (e.g. building or non-building) without examining neighboring pixels in a local context. For example, a pixel on a road can share very similar visual band values with a pixel on a building roof-top. The appropriate neighborhood size to consider often varies from small (e.g. 10 meters) to large building footprints (e.g. 200 meters). Second, building footprints are often densely distributed in urban areas. A 500 m by 500 m region, for example, may contain hundreds of residential houses in the United States, and some of the buildings may be separated by only a few pixels in an image. Third, deep learning frameworks demand high computational power due to the large size of the training datasets and the large number of parameters.

Prior to the advent of deep learning, object detection algorithms most commonly employed traditional image processing techniques to construct manual features, and then feed them into a machine learning classifier (e.g. support vector machine) to predict the objects. The same strategies were used in building detection. The earlier methods mainly focused on detecting a few large building footprints in a single aerial image. Lin and Nevatia (1998) used fragmented linear features (e.g. roof boundaries) to construct one or several cubes to represent the three-dimensional large buildings in a skewed aerial image. Similarly, the Hough transform was used to identify the linear features in an image, followed by a semi-automatic scheme to annotate a few large buildings in the scene (Cui *et al*. 2008). Besides linear features, critical feature points (e.g., SIFT) were also employed to identify the locations of buildings in a delineated urban area (Sirmacek and Unsalan 2009). Beyond these geometric features, zonal features were explored to assist building detection. Cohen *et al*. (2016) used a combination of several image processing techniques to construct manual features of building shadows, and was able to recover the sizes of well-separated large buildings based on shadow characteristics. In addition, multi-spectral imagery was also used to build local HOG (histogram of oriented gradients) features, which are fed into a support vector machine to identify the pixels of building footprints (Konstantinidis *et al*. 2017).

In recent years, deep learning has revolutionized computer vision with its greatly improved performance over traditional image processing methods (Krizhevsky *et al*. 2012, Razavian *et al*. 2014, LeCun *et al*. 2015). Specifically, convolutional neural networks (CNN) (Goodfellow *et al*. 2016, Shin *et al*. 2016), which are locally connected deep neural networks, have become the new state-of-the-art in image classification and labeling. For remote sensing images, CNN has demonstrated improved ability in feature construction, compared to the traditional manual features (e.g., HOG) (Nogueira *et al*. 2017). CNN has also been applied to classify the land use and land cover themes of aerial image patches with high precision (Romero *et al*. 2016, Kussul *et al*. 2017). However, as shown above, the main objective of CNN is to predict the theme of an input image patch (e.g., building scene, non-building scene), and it does not output the actual locations and sizes of the objects in an image. Thus, CNN is not well-suited for building footprint detection in its original structure. To extend CNN's capability, You-Only-Look-Once (YOLO) frameworks (Redmon *et al*. 2016, Redmon and Farhadi 2017) were proposed to predict the bounding boxes of objects in an image through additional regression modeling. Taking advantage of the end-to-end learning scheme, the YOLO framework directly outputs all the bounding boxes of objects in its final network layer, and avoids the need for a separately-designed preprocessing phase (e.g. image patching). As indicated by its name, YOLO needs only one pass of an image through the network to get all the results, which significantly improves its performance during the prediction phase. While YOLO is a state-of-the-art framework for object detection, it is still limited in finding small objects appearing in groups (Redmon *et al*. 2016, Redmon and Farhadi 2017). In the building detection problem, small and densely distributed buildings are commonly observed in urban areas, making it difficult for YOLO to correctly recognize and separate out individual buildings in remote sensing data. Besides object-based YOLO and its variations, semantic segmentation is another branch of CNN which focuses on pixel-level classification. Through per pixel label predictions, recent work has tried to use semantic segmentation (Zhao *et al*. 2017, Chen *et al*. 2018, Yuan 2018) to obtain exact boundaries of building footprints. Semantic segmentation methods require exact boundaries of building footprints as training data, which may introduce more noises considering the pixel-level match between the footprint data and the imagery (e.g. tree canopies covering a corner of a building). In addition, the accuracy of semantic segmentation methods are often evaluated at pixel-level, which does not necessarily reflect the solution quality at the object-level.

In this work, we focus on the object-based branch of CNN. Specifically, we address the limitations of YOLO by proposing a LOcally-COnstrained (LOCO) You-Only-Look-Once framework to identify small and densely distributed building footprints in remote sensing datasets. Our design decisions in LOCO are theoretically grounded on an analysis of the "receptive field" (Dumoulin and Visin 2016, Luo *et al*. 2016) in convolutional neural networks, which helps explain why a deep learning architecture may or may not be suitable for the building detection problem. In addition, we propose a constrained regression modeling for bounding box construction in the output layer to improve the robustness of building size prediction. To further improve the precision of detections, we present an invariant augmentation based voting scheme which can be applied without any changes in the training process.

To validate the proposed techniques, we compared the solution quality of YOLO, LOCO and two intermediate frameworks using identical training and testing datasets. We also tested four semantic segmentation schemes such as PSPNet (Zhao *et al*. 2017) and DeepLabV3+ (Chen *et al*. 2018) to compare the object-based and pixel-based CNN branches. The results showed that the LOCO framework can significantly boost F1 scores (a statistical measure combining precision and recall) of detection results. We also presented the visualizations of detections from the candidate frameworks at multiple scales to highlight the improvements.

1.1. Scope and outline

This paper investigates bounding-box based deep learning frameworks for building footprint detection using the visual bands of satellite imagery. Specifically, we study a common challenging scenario where an image may contain hundreds of small and densely distributed buildings.[1] The goal of this work is not to replace the role of human experts in building cataloging. Instead, we consider the output building footprints as suggestions which will go through manual inspection before being archived in building catalogs. In other words, the role of deep learning is to assist human experts in the process of data generation.

The rest of the paper is organized as follows. Sec. 2 introduces the key concepts and formally defines the problem, Sec. 3 presents the proposed LOcally-COnstrained (LOCO) You-Only-Look-Once framework, Sec. 4 summarizes the experiment results, Sec. 5 discusses challenges and opportunities, and Sec. 6 concludes the key contributions of the paper with future work.

2. Problem formulation

We first introduce the following key concepts before formal problem definition:

Features: Features are variables that are used to predict target values (e.g. class labels) in a machine learning algorithm. In this problem, features are the visual bands and their derivatives in a satellite image.

Minimum Orthogonal Bounding Rectangle (MOBR): For a two-dimensional polygon, its MOBR satisfies three conditions: (1) it is rectangular in shape; (2) it is orthogonal in direction; and (3) it covers the entire polygon with a minimum area. In this problem, we use MOBRs to approximate locations and sizes of building footprints.

Ground truth: A collection of geo-located MOBRs of building footprints in the study area. Part of the ground truth is used for training, and the rest is used to test the solution quality. Training data and test data must be mutually exclusive.

Threshold for spatial resolution: A constraint on the spatial resolution of satellite imagery. It requires that the resolution is sufficiently high (e.g. smaller than 2-meter) to make it feasible for detecting building footprints. An input imagery is not valid if it fails the threshold.

Threshold for spatial extent: It defines the minimum size of an input image, and is used (1) to reduce the number of building footprints that intersect with the boundary of a spatial region, and (2) to allow large building footprints to be contained inside the

extent. For example, if a spatial region is of size 100 m by 100 m, then its area is not sufficient to cover buildings of larger sizes (e.g. parking ramp, hospital, mall).

With the key concepts, the building detection problem is formally defined as follows:

Inputs:

– Satellite imagery (visual bands) in a set S of spatial regions (e.g. zones in several cities);
– Ground truth data in a set S' of spatial regions, where $S' \subset S$;
– A threshold for minimum spatial extent: t_e;
– A threshold for minimum spatial resolution: t_r.

Output:

– Building footprints for the set of spatial regions in $S - S'$;

Objectives:

– Solution quality (e.g. precision, recall, F1 score);
– Computational efficiency;

Constraints:

– The extent (i.e. width, height) of each spatial region satisfies t_e;
– The spatial resolution of input remote sensing data satisfies t_r.

3. Locally constrained deep learning framework

In this section, we propose a LOcally COnstrained (LOCO) You-Only-Look-Once framework to identify small and densely distributed building footprints in remote sensing datasets. Here "small" is relative to the spatial extent covered by an input image. The size of buildings can vary greatly based on their purpose. For example, in the United States, a typical size of residential houses in an urban area could be less than 15 meters, whereas a large office complex, shopping mall or a grocery store could span hundreds of meters. Thus, the spatial extent of an image needs to be at least the same as most of the largest buildings in a target area. If the spatial extent is too small (e.g. 100 m by 100 m), then an image may be entirely taken up by just a part of a building roof. In addition, the larger the spatial extent of an image, the smaller the portion of building footprints that are clipped at the boundaries. In this work, we set the minimum spatial extent t_e (Sec. 2) to 400 m.

Our proposed LOCO framework is based on the You-Only-Look-Once (YOLO) framework (Redmon *et al.* 2016, Redmon and Farhadi 2017). In Sec. 3.1, we first demonstrate the limitations of YOLO for building detection through both detailed analysis and intuitive examples. These analyses are used to inform the design of our LOCO framework in Sec. 3.2. In Sec. 3.3, we propose a constrained regression modeling scheme to improve the robustness of building size estimation. Finally, Sec. 3.4 presents a voting scheme (test time augmentation) to further eliminate false detections and improve the precision of results.

3.1. A spatial angle: network architecture and building detection

In this section, we take a spatial perspective, and using the concept of "receptive field" (Dumoulin and Visin 2016, Luo *et al.* 2016), we analyze the spatial information propagation in YOLO to better understand its effects on building detection. This analysis will then be used to inform the architecture design of our proposed LOCO framework.

3.1.1. Spatial information propagation: receptive field in CNN

In the human eye, the receptive field of a sensory neuron refers to the visual field that affects the firing of the neuron. By analogy, the receptive field of a cell in a CNN layer defines the spatial extent in the input space that contributes to the value of the cell (Dumoulin and Visin 2016).

For example, in the input layer, the receptive field of all cells is just the cells themselves. If we have a convolutional layer of size 3×3, then after one convolution, the receptive field of each output cell becomes 3×3.

To compute the exact receptive field of each cell, we have to additionally consider the location of the cell, the dimensions of the input, as well as the padding method being used. As an example, the receptive field of a cell that is close to a border of the input can be limited by its distance to the border, and that limit is often not symmetric along each dimension. Since such details are not necessary for a general understanding of spatial information propagation in a CNN, here we only consider kernel size k and stride s in the calculation. The resulting size of the receptive field is then the maximum size of receptive fields of all elements in the output. The size of a receptive field at the l^{th} layer is calculated as:

$$W_l = \max\{D_0, (k_l - 1) \cdot \prod_{i=1}^{l} s_i + W_{l-1}\} \tag{1}$$

where W_l is the side length of the receptive field at the l^{th} layer, D_0 is the side length of the input (or the 0^{th} layer), k^l is the kernel size of the l^{th} layer, and s_i is the stride of the i^{th} layer.

The process of spatial information propagation, as described by the receptive field, has profound implications on building detection.

3.1.2. Information propagation effects on building detection

While increasing the depth of a neural network can improve its expressive power (Cybenko 1989), the increased sizes of receptive fields may not always be beneficial for the learning process. In scenarios such as small building detection, a large receptive field can actually introduce a large volume of noise which may hurt the training.

In the following, we first analyze the receptive fields of two YOLO frameworks, full YOLO, which has 24 layers, and tiny YOLO, which has 15. Figure 1(a,b) show how the receptive field grows as an input flows through the layers.

By the last layer, full YOLO and tiny YOLO's receptive fields have grown to 374×374 and 318×318, respectively. In addition, starting from layer 19 in full YOLO and layer 11 in tiny YOLO (highlighted in Figure 1), the kernel size of convolutional operations becomes 96×96 (pixels in the original input image).

In YOLO frameworks, a common input image size is 416 rows $\times$ 416 columns. Figure 2 visualizes the expansion of the receptive field on an everyday image and a satellite image.

The ability to expand the receptive field of an image is very beneficial for detecting objects in everyday images because the objects are often quite large in proportion to the pixels they cover in the frame. For example, the cow in Figure 2 covers a substantial portion of the image frame. In such cases, it is meaningful to construct "large" features with an increased number of convolutional and pooling layers. By contrast, the buildings

ID	Layer type	Size (row × column)	Size in original input pixels	Receptive field
1	Convolutional	3 × 3	3 × 3	3 × 3
2	Max-pool	2 × 2	2 × 2	4 × 4
3	Convolutional	3 × 3	6 × 6	8 × 8
4	Max-pool	2 × 2	4 × 4	10 × 10
5	Convolutional	3 × 3	12 × 12	18 × 18
6	Convolutional	1 × 1	4 × 4	18 × 18
7	Convolutional	3 × 3	12 × 12	26 × 26
8	Max-pool	2 × 2	8 × 8	30 × 30
9	Convolutional	3 × 3	24 × 24	46 × 46
10	Convolutional	1 × 1	8 × 8	46 × 46
11	Convolutional	3 × 3	24 × 24	62 × 62
12	Max-pool	2 × 2	16 × 16	70 × 70
13	Convolutional	3 × 3	48 × 48	102 × 102
14	Convolutional	1 × 1	16 × 16	102 × 102
15	Convolutional	3 × 3	48 × 48	134 × 134
16	Convolutional	1 × 1	16 × 16	134 × 134
17	Convolutional	3 × 3	48 × 48	166 × 166
18	Max-pool	2 × 2	32 × 32	182 × 182
19	Convolutional	3 × 3	96 × 96	246 × 246
20	Convolutional	1 × 1	32 × 32	246 × 246
21	Convolutional	3 × 3	96 × 96	310 × 310
22	Convolutional	1 × 1	32 × 32	310 × 310
23	Convolutional	3 × 3	96 × 96	374 × 374
24	Convolutional	1 × 1	32 × 32	374 × 374

(a) Full YOLO

ID	Layer type	Size (row × column)	Size in original input pixels	Receptive field
1	Convolutional	3 × 3	3 × 3	3 × 3
2	Max-pool	2 × 2	2 × 2	4 × 4
3	Convolutional	3 × 3	6 × 6	8 × 8
4	Max-pool	2 × 2	4 × 4	10 × 10
5	Convolutional	3 × 3	12 × 12	18 × 18
6	Max-pool	2 × 2	8 × 8	22 × 22
7	Convolutional	3 × 3	24 × 24	38 × 38
8	Max-pool	2 × 2	16 × 16	46 × 46
9	Convolutional	3 × 3	48 × 48	78 × 78
10	Max-pool	2 × 2	32 × 32	94 × 94
11	Convolutional	3 × 3	96 × 96	158 × 158
12	Max-pool*	2 × 2	64 × 64	190 × 190
13	Convolutional	3 × 3	96 × 96	254 × 254
14	Convolutional	3 × 3	96 × 96	318 × 318
15	Convolutional	1 × 1	32 × 32	318 × 318

*This max-pool layer only replaces the value of a cell by the maximum at its local 2 x 2 region, but does not reduce the resolution

(b) Tiny YOLO

Figure 1. Expansion of receptive fields in full and tiny YOLO. The third and fourth columns both show the size of the convolutional kernel and pooling window at the corresponding layer. The difference is that in the third column, the size is measured by the number of grid cells at the current layer, whereas in the fourth it is measured by the number of pixels in the original input image. The underlined sizes highlight the increases in kernel size caused by the max-pooling layers.

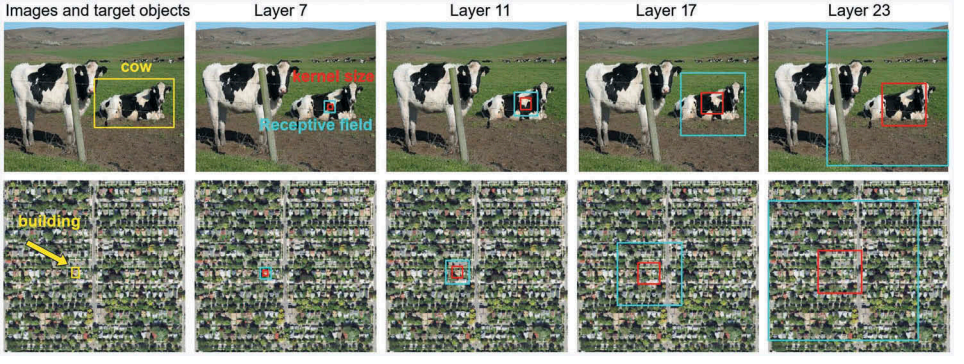

Figure 2. Visualization of growth in kernel size and receptive field in YOLO. The yellow bounding boxes represent two example target objects in the images (i.e. a cow and a building). The red and blue boxes show the kernel sizes and receptive fields at corresponding layers. For objects in everyday images, the large kernel size and receptive field are meaningful and necessary to provide the information needed for detecting the object. However, for buildings in remote sensing images, the large extent introduces a huge volume of noise and increases the difficulty of learning. (Best viewed in color).

in satellite images are often very small relative to the frame size. A typical house in a US residential area might measure 8×15 meters. For one-meter resolution imagery (e.g. NAIP), that only covers less than 0.07% of a 416×416 image frame. This means that if the receptive field covers the entire image frame, then most of the information being considered is actually irrelevant or noise. Thus, in these scenarios, it is no longer meaningful to have a large receptive field.

In Figure 3, we mark two example locations (blue crosses), one on an everyday image and one on a satellite image. The locations are about 50–100 pixels away from the centers of the objects (red crosses), and are within the final receptive field of each image (e.g. layer 23 in Figure 2). We can see that the information is very helpful to determine the boundary of the cow object in Figure 3(a) but is irrelevant for the target building object in Figure 3(b). In addition, the information at the upper blue cross in Figure 3(b) contains information about several buildings other than the target one. If that information is actually used to determine the size of the target building, the estimation may significantly deviate from the true size. Indeed, such information introduces noise into the learning process, and it may become a difficult task for the model to identify the small amount of information that is useful.

3.2. LOCO network architecture

Using the spatial information propagation analysis and its effects on building detection, here we present our LOcally-COnstrained (LOCO) You-Only-Look-Once framework. In the LOCO framework, "locally constrained" means that the receptive field will be constrained to a smaller local region by the network architecture design.

LOCO aims to leverage existing large-scale high quality datasets, such as the previously described US NAIP imagery, which has one-meter resolution, and is collected and integrated annually on a national scale. It is also important that images in the datasets have a sufficiently large spatial extent to allow large buildings to be contained.

Our LOCO framework targets one-meter high-resolution data, and each image input is 416 $\times$ 416 in size, corresponding to a 416-meter by 416-meter spatial extent. Since the desired receptive fields for small and large buildings are very different (e.g. dozens vs. hundreds of pixels), our overall network structure includes a bifurcation of tasks as shown in Figure 4. Specifically, the upper branch is for small building detection and the lower branch is for large building detection. The two branches require separate training on different ground truth data (i.e. small and large buildings). To split building footprints into "small" and "large" groups, we used 32 meters as the threshold for the side-length of a building's bounding box. The threshold was determined by clustering all the sizes of building footprints in our data. Other thresholds could be used as long as

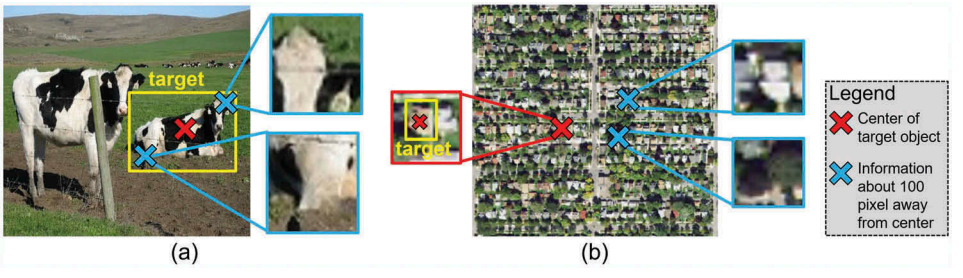

Figure 3. Usefulness of information that is far away from an object in (a) an everyday image and (b) a satellite image. The target objects are marked by the yellow bounding box, and their centers are marked by the red crosses. The information is meaningful to estimate the size of the cow object but becomes noise for estimating the target building size. (Best viewed in color).

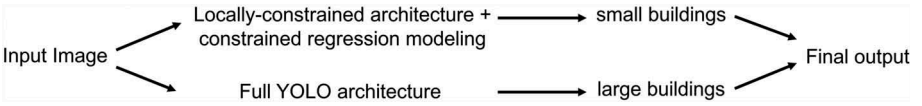

Figure 4. The overall deep learning framework of LOCO. The architecture has two branches. The upper branch uses the proposed locally-constrained design, which is based on the spatial propagation of information. The lower branch uses the full YOLO architecture. In a building detection task, the upper and lower layers are responsible for detecting small and large building footprints, respectively.

they fit the receptive fields of the two branches in the architecture (Sec. 3.2.1 and 3.2.2). In addition, this hard-thresholding scheme can potentially be improved by soft-thresholding through future research.

3.2.1. Small and densely distributed building footprints

Our design decision is based on the spatial information propagation analysis and the spatial properties of the input (i.e. 416 m by 416 m at one-meter resolution). According to Figure 1(a), full YOLO has 24 layers in total, which contributes to its large receptive field. Even in the 15-layer tiny YOLO, the receptive field reaches 318×318 in size.

Since tiny YOLO has the smaller receptive field, we use it as the base of our LOCO framework. As shown in Figure 1(b), most of the expansion in the receptive field of tiny YOLO is introduced by the last two max-pooling layers. Thus, in our architecture we skip the down-resolution operations from the last two max-pooling layers. In other words, the new max-pooling layer only replaces a cell's value by the maximum value at its local 2×2 region but does not reduce the resolution (i.e. same as max-pooling* in Figure 1(b)). Table 1 shows the difference in convolutional kernel sizes and receptive fields among full YOLO, tiny YOLO, and LOCO.

LOCO architecture is designed to avoid the majority of the non-relevant information in the receptive field and thus reduce the difficulty of learning for small buildings. Since small buildings form the dominating landscape in many urban areas, we expect this specialized design for small building detection to improve the precision of a major proportion of all buildings.

3.2.2. Large building footprints

For large building footprints, we may need information from a large spatial extent (e.g., side length of hundreds of pixels) in the corresponding input image in order to determine their locations and sizes. This scenario becomes very similar to object detection in everyday images, so it is appropriate to use network architectures that create a large receptive field. Thus, for the lower branch in Figure 4, we use the full YOLO architecture to detect large building footprints and this completes the general framework of LOCO.

Table 1. Convolutional kernel sizes and receptive fields at different layers.

	Kernel size				Receptive field			
Layer ID	7	13	19	24	7	13	19	24
Full YOLO	12	48	96	96	26	102	246	374
Tiny YOLO	24	96	–	–	38	254	–	–
LOCO-small	24	24	–	–	38	110	–	–

3.3. Constrained modeling of bounding box regression

YOLO uses regression modeling to determine the location and sizes of objects (see Appendix A for details). Here we aim to improve the robustness of YOLO's regression modeling for small building detection.

In the final output layer of YOLO (e.g., a coarse 13×13 grid pooled from an original 416×416 image), the cell that contains the center of an object is responsible for predicting four parameters related to its location and size: the center coordinates c_x and c_y, width l_w, and height l_h. As discussed in Appendix A, the final values of the four parameters are not directly given by the final layer outputs, but rather are derived from them using Equation (A1). Here we focus on the size parameters: width l_w and height l_h. According to Equation (A1), their values are derived using an exponential function (e.g., for width it is $l_w = anchor_w \cdot e^{p_w}$, where p_w is a final layer output). For objects in everyday images, the use of an exponential function is helpful to cover a wide-range of box sizes. However, small buildings have a much smaller range of sizes, so having that wide range of coverage may increase model sensitivity. For example, with the exponential function, the range of sizes considered in prediction is $(0, +\infty)$. By contrast, in our LOCO framework, the actual range of sizes of small buildings is (0,32).[2] Thus, we propose a constrained regression modeling to limit the range of predicted object dimensions to improve the robustness and ease of learning:

$$\begin{aligned} l_w &= w_{max} \cdot \sigma(p_w) \\ l_h &= h_{max} \cdot \sigma(p_h) \end{aligned} \tag{2}$$

where $\sigma(\cdot)$ is the sigmoid function, p_w and p_h are values from the output layer of the neural network, and w_{max} and h_{max} are the maximum width and height used for small buildings. In our design (Sec. 3.2), both values are set to 32.

Since the range of outputs of sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ is (0,1) for any input x, the ranges of l_w and l_h are limited to (0, w_{max}) and (0, h_{max}). Figure 5 shows a stability comparison between Equation (A1) and Equation (2), in which changes in l_w and l_h are evaluated after a small error is injected into the network predictions p_w and p_h. The smaller the changes in l_w and l_h, the more robust the modeling. From Figure 5 we can see that our constrained modeling is less sensitive to changes in p_w and p_h, making the learning process easier for the deep neural network.

Total number of predictions: Intuitively, it may seem that YOLO is not suited for small object detection due to the limited number of grid cells (i.e. 13×13) in its output layer. However, with the use of anchor boxes (Appendix A), we can see that YOLO can actually detect multiple objects in a grid cell. For example, using YOLO's default 5 anchor boxes, we can get 5 objects per cell. This means that in each image, we can get as many as $13 \times 13 \times 5 = 845$ detections, which should be sufficient for small objects (e.g., building footprints in remote sensing images). Thus, the number of grid cells is not a theoretical limitation of YOLO in the case of small objects. In Sec. 3.1, we showed that over-expansion of spatial information propagation may decrease YOLO's performance on small object detection.

Note that in our new regression modeling, Equation (2) no longer requires anchor boxes. While each cell in this scheme can have at most one detection, the LOCO

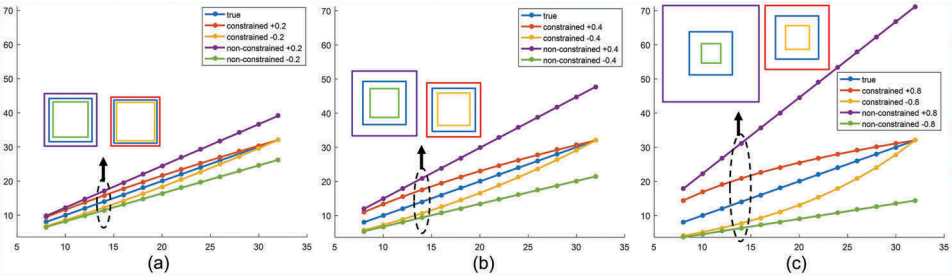

Figure 5. Robustness comparison: sensitivity to errors. The X axis is the true size and Y axis is the predicted size. We introduce a small error ϵ into the true network predictions p_w^* and p_h^*, and evaluate the changes in l_w and l_h computed by non-constrained modelings (Equation (3) with anchor size 16×16) vs. constrained modeling (Equation (A1)). We set $\epsilon = \pm 0.2, \pm 0.4, \pm 0.8$ in (a), (b) and (c), respectively. To better visualize the differences in sizes, we draw the true and predicted boxes for size 14. We can see that the boxes constructed by the constrained regression modeling are less sensitive to the introduced errors.

architecture for small buildings has an increased number of grid cells in the final layer (i.e. 52×52). Thus, it can detect as many as 2704 objects, which is more than the 845 in YOLO and should be sufficient for small objects.

3.4. *Invariant augmentation based voting scheme*

So far we have completed our discussion related to the learning (or training) phase in the proposed LOCO framework. In this section, we present a post-processing technique that is only related to the prediction phase. The idea is based on test time augmentation, which aims to generate multiple detections of the same object and then use all the detections to determine the final parameters of the object.

There exists a variety of ways to augment an image. For building detection, we found that some augmentations can significantly impact detection results. For example, rotation is a common augmentation strategy. When an image is rotated at an angle that is not a multiple of 90°, empty areas (e.g., black pixels with 0 values) will be introduced in the result. This is because an image is always stored as a rectangular matrix (with depth), which is also the required input format for convolutional neural networks. Thus, once an image is rotated with an angle that is not a multiple of 90°, empty spaces are automatically created in the four corners. In our tests, such empty areas had a huge impact on the detection results, causing many missing detections. In addition, such rotations also require a re-sampling of the pixel values. For example, with a 45° rotation, the pixel grid will become tilted (i.e. non-orthogonal directions). Thus, to store it as an image we need to re-sample the values to an orthogonal grid. To avoid such effects, we use an invariant augmentation scheme:

Definition 3.1. ***Invariant augmentation.*** An image augmentation is invariant if: (1) it maintains the same spatial extent of the original image; (2) it maintains the same values (e.g., R,G,B values) of the original image at every spatial location in the image; and (3) it is not the same as the original image. Here "spatial location" means a real-world spatial location represented by a pixel in the image but not the row and column ID of the pixel.

According to Def. 3.1, invariant augmentation does not miss, add or update any values to the spatial locations in the original image, so information-wise it is equivalent to the original image. Lemma 3.2 shows that there exist seven and only seven invariant augmentations.

Lemma 3.2. *There are seven invariant augmentations. Denote X as the original image,* $F_h(\cdot)$ *as a horizontal flipping operation,* $F_v(\cdot)$ *as vertical flipping, and* $R_{90}(\cdot)$ *as a* 90° *rotation operation. The seven invariant augmentations are: (1)* $F_h(X)$, $F_v(X)$, $F_v(F_h(X))$, $R_{90}(X)$, $F_h(R_{90}(X))$, $F_v(R_{90}(X))$, $F_v(F_h(R_{90}(X)))$.

Proof. First, we prove the seven operations are invariant. A flipping operation does not change the spatial extent or values of an image since it just reverses the order of rows or columns. Similarly, a 90° rotation only switches rows and columns. Since each of these transposition operations is invariant, their combinations are also invariant. Then, we show that only these seven transpositions are invariant. There are five categories of transposition: ″crop″, ″translate″, ″resize″, ″rotate″ and ″flip″. ″Crop″ and ″translate″ operations inevitably change the spatial extent of the original image. For ″resize″ and ″rotate″ (with a degree that is not a multiple of 90°), the values may change due to the need for resampling. Finally, the seven invariant augmentations cover all possible combinations of ″flip″ and ″rotate″ (by multiples of 90°) operations.

Using the seven invariant augmentations as well as the the original image itself, we can obtain a set of eight equivalent images to use in the voting process. Then, we apply the trained model to get all detections from the eight images and project the detections back to map coordinates (i.e. overlaying detections of the same object). **Existence voting**: Existence is a qualitative decision, so we use majority-voting to determine if a detected object is likely to exist. Specifically, if we have more than four detections at the same location, then we accept the object's existence; otherwise we reject it. **Location and size voting**: Location and size are quantitative decisions, so we choose the median c_x, c_y, l_w, and l_h values among all detections.

Note that the training images should be augmented the same way to improve the robustness. For example, the directions of shadows may be an important feature used by deep learning models. Augmenting the training data in the same way allows the model to see the shadows in different directions (e.g., north vs. south) which can help improve the detection accuracy on augmented test images.

Our goal is to use this invariant augmentation based voting scheme to improve the precision of the proposed framework. The intuition is that a ″non-object″ (e.g., ″non-building″) may be mistakenly detected in one of the augmentations, but the probability of it being mistakenly detected in a majority of images is comparatively lower, assuming that the precision of the model is greater than 50%.

4. Validation

The major goal of our experiments was to show that the proposed LOCO framework can address the limitations of the YOLO framework (i.e. not well-suited for detecting small objects appearing in groups (Redmon *et al*. 2016, Redmon and Farhadi 2017)) and thus

improve its performance on building detection. We additionally included semantic segmentation in the experiments to provide a general comparison between object-level and pixel-level deep learning methods for the task of building detection, although it is not the main focus of the paper.

4.1. Candidate methods

Since our aim is to explore deep learning frameworks for building footprint detection, we compared eight deep learning methods in the evaluation:

YOLO-13: The original full YOLO framework with 13 rows and columns in its final output layer.

YOLO-52: A partial LOCO framework. In the complete LOCO framework, the architecture is separated into two branches to consider the different characteristics of small and large buildings. Here YOLO-52 does not apply the branching. Instead, it uses the small building framework for all buildings. The final layer has 52 rows and 52 columns. Note that in this case we do not apply the constrained regression modeling (Sec. 3.3) since it is only for small building footprints.

LOCO-single: The LOCO framework without the voting scheme presented in Sec. 3.4. It constructs output directly from the input itself.

LOCO-vote: The complete LOCO framework with the voting scheme based on invariant augmentation.

PSPNet (binary): The Pyramid Scene Parsing Network (PSPNet) (Zhao *et al*. 2017), a semantic segmentation model for pixel-level classification. Pixels in its training data are labeled in binary values, i.e. 1 for building pixels and 0 for non-building pixels.

PSPNet (distance): PSPNet with training data labeled with the signed distance function (Yuan 2018). The distance at each pixel is calculated as its distance to the nearest building boundary. The signs of distances are negative inside building footprints and positive outside. The signed distances are grouped into 39 classes (i.e. 19 positive, 19 negative and 1 zero) to label the pixels.

DeepLabV3+ (binary): Google's latest and best-performing open semantic segmentation framework. The binary version is trained with binary labeling.

DeepLabV3+ (distance): DeepLabV3+ trained with signed distance labeling (Yuan 2018).

4.2. Training and testing

The training datasets consist of two parts: (1) the satellite images; and (2) the ground truth building footprints. Both datasets were from free and publicly available sources. Since Minnesota, US is an important region for our potential application, we collected our datasets from multiple sites within five different cities in the state of Minnesota using US NAIP imagery. The building footprint datasets were obtained from the state government's public resources. We used data from all sites in four of the cities as the training data, and data from the fifth city as the test data. The test city is more than 150 kilometers away from the nearest training city. This ensured that the model did not pre-observe the landscape in the test city during the training phase. We used this setup to

approximate a general real-world scenario where some cities maintain regulated routines of spatial data collection and digitization but others may not.

As we discussed in Sec. 3, the LOCO framework takes input images of size 416 × 416, corresponding to a 416 m by 416 m region at one-meter resolution. Thus, the imagery was split into 416 × 416 chunks to feed into the network. The building footprints were preprocessed into minimum orthogonal bounding boxes (MOBR, Sec. 2) to use in the training phase. Since our dataset did not have good coverage of building footprints that are smaller than 50 m^2 (e.g., small storage sheds in back yards), we did not consider those footprints in the experiments and removed them from both training and testing.

Overall, our training dataset contained 5185 images of size 416 m × 416 m, leading to a total area of about 900 square km. There were 172,096 ground truth building footprints (above 50 m^2) in the 5185 zones.

Since semantic segmentation frameworks perform pixel-based classification instead of object-level detection such as YOLO, they require exact boundaries of building footprints to generate the training data. However, in our targeted application area (Minnesota), the number of exact building boundaries we had was not large enough. In other words, many of the training samples in our target application area are well-suited for object based methods (e.g., LOLO and YOLO) but not appropriate for pixel level segmentation. Thus, to increase the amount of training data with exact boundaries for semantic segmentation frameworks, we included an extra public Massachusetts building dataset (Massachusetts Buildings Dataset 2013). The number of building footprints added was 125,220. The satellite images we used with these building footprints were also collected from NAIP imagery to keep better consistency with our test dataset.

Finally, to facilitate the training process, we adopted the transfer learning scheme and used pre-trained parameters on everyday objects in the shallow layers for all the methods.

4.3. Solution quality evaluation

The standard metrics used for evaluation were: (1) precision ($\frac{|\{Detections\}\cap\{GroundTruth\}|}{|\{Detections\}|}$); (2) recall ($\frac{|\{Detections\}\cap\{GroundTruth\}|}{|\{GroundTruth\}|}$); and (3) F1 score ($\frac{2}{precision^{-1}+recall^{-1}}$). Precision is the percentage of detections that are true. Recall is the percentage of true buildings successfully detected. The F1 score is the harmonic mean of precision and recall, which unites the two measures for direct comparisons. The numbers of building footprints in Area 1 to Area 5 in our test city were: 1831, 1699, 2741, 1917 and 2746. Tables 2, 3 and 4 show the precision, recall and F1 scores of the eight candidate frameworks in the five test areas. To better visualize the trends, the tables' corresponding charts are shown in Figure 6.

4.3.1. Comparison between LOCO and YOLO

As we can see, YOLO-13 did not perform very well on detecting building footprints that are small and densely distributed in the remote sensing datasets. Based on our analysis in Sec. 3.1, one major cause of this could be the excessive size of the receptive field created by the full YOLO framework.

Table 2. Precision for building detection.

Framework	Area 1	Area 2	Area 3	Area 4	Area 5
PSPNet (binary)	71.06%	66.55%	68.22%	62.57%	62.90%
DeepLabV3+ (binary)	73.95%	73.06%	66.25%	68.52%	64.41%
PSPNet (distance)	63.77%	61.87%	62.58%	56.89%	55.66%
DeepLabV3+ (distance)	67.18%	66.97%	66.73%	62.69%	59.51%
YOLO-13	46.99%	46.76%	45.87%	49.43%	42.31%
YOLO-52	63.69%	63.64%	66.05%	61.76%	58.51%
LOCO-single	80.39%	75.78%	76.93%	74.73%	72.86%
LOCO-vote	91.40%	89.54%	90.00%	91.30%	89.63%

Table 3. Recall for building detection.

Framework	Area 1	Area 2	Area 3	Area 4	Area 5
PSPNet (binary)	66.96%	67.75%	69.21%	68.44%	60.52%
DeepLabV3+ (binary)	64.39%	69.51%	68.52%	69.69%	62.20%
PSPNet (distance)	67.12%	77.40%	72.31%	73.08%	62.05%
DeepLabV3+ (distance)	54.72%	60.80%	65.27%	66.56%	57.54%
YOLO-13	43.97%	39.61%	44.62%	52.43%	39.44%
YOLO-52	76.24%	71.39%	73.66%	79.55%	76.69%
LOCO-single	73.84%	68.33%	72.05%	77.36%	72.40%
LOCO-vote	71.27%	62.57%	71.69%	76.11%	70.79%

Table 4. F1 scores for building detection.

Framework	Area 1	Area 2	Area 3	Area 4	Area 5
PSPNet (binary)	68.95%	67.14%	68.71%	65.37%	61.69%
DeepLabV3+ (binary)	68.84%	71.24%	67.36%	69.10%	63.29%
PSPNet (distance)	65.40%	68.77%	67.09%	63.98%	58.69%
DeepLabV3+ (distance)	60.32%	63.74%	65.99%	64.57%	58.51%
YOLO-13	45.43%	42.89%	45.23%	50.89%	40.82%
YOLO-52	69.40%	67.30%	69.65%	69.54%	66.38%
LOCO-single	76.97%	71.86%	74.41%	76.02%	72.63%
LOCO-vote	80.09%	73.66%	79.81%	83.02%	79.11%

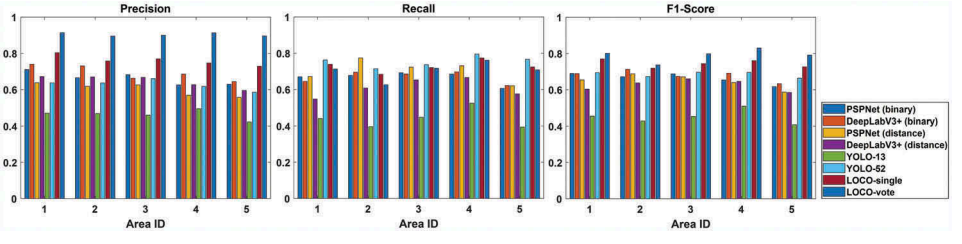

Figure 6. Comparison of precision, recall and F1 score. LOCO-vote is able to significantly improve the precision of detections while maintaining a similar level of recall. As a result, LOCO-vote achieves the highest F1 scores among the 8 methods. The accuracy of semantic segmentation is similar to YOLO-52.

According to Table 2, YOLO-52 has an improved precision rate (10%-20%) over YOLO-13. As an intermediate architecture between original YOLO-13 and LOCO, YOLO-52 has a reduced receptive field, which avoids most of the noise and makes it easier to construct useful features. However, since YOLO-52 does not have the bifurcation structure in LOCO, its single-sized receptive field is not large enough for large building footprints. Also, trying to learn small and large buildings together may potentially

introduce confusion into feature construction during training. Thus, compared to LOCO-single and LOCO-vote, these characteristics of YOLO-52 lead to a 10%-20% reduction in precision. On the other hand, there is also a potential disadvantage of the bifurcation structure in LOCO. For example, the upper branch (Figure 4) does not include large building footprints in its training data, so it needs to penalize detections of the large building footprints. As a result, if some small buildings share similar features with some large buildings, they may also get penalized during training. Thus, compared to YOLO-52, LOCO-single has a tendency to detect fewer building footprints (i.e. 1%-4% lower recall) in the input images (Table 3). Overall, comparing LOCO-single and YOLO-52, the former is able to achieve a significantly higher precision (i.e. 10%-20%) without giving much up in recall (i.e. 1%-4%).

For LOCO-single and LOCO-vote, we can see in Table 2 that the voting scheme does make a great difference in reducing the number of false detections and brings another 10% to 20% improvement in precision. Similarly, it is interesting to see that the voting scheme is able to maintain the same level of recall (Table 3) while gaining significant improvement on precision. In general, the recall of LOCO-vote is about 1% to 2% lower than that of LOCO-single, but it has a further 10% to 20% boost in precision. This asymmetry in the precision-recall trade-off shows the potential value of the voting scheme. It is worth mentioning that in Area 2, LOCO-vote exhibits a 5% decrease in recall. While this is still smaller than the 15% increase of precision, it calls for future work to further analyze and improve the robustness of the voting scheme.

Finally, in regards to both precision and recall, Table 4 shows the F1 scores of the candidate methods. Since there is often a trade-off between precision and recall, the F1 score is a standard statistical measure that combines them into a single statistic for direct comparisons. The trend is that LOCO-single and LOCO-vote consistently achieve higher F1-scores in the test areas compared to YOLO-13 and YOLO-52. Within the two groups, YOLO-52 outperforms YOLO-13 and LOCO-vote outperforms LOCO-single.

4.3.2. Comparison between LOCO and semantic segmentation

Semantic segmentation focuses on pixel-level classification. For the task of building detection, it aims to find the exact boundary of building footprints using segments created by predicted labels on pixels (e.g., (Yuan 2018)). However, as shown in Figure 9, currently it is still difficult for semantic segmentation to well-approximate actual building boundaries at meter high resolution, especially for small building footprints.

Accuracy of semantic segmentation is often evaluated at the pixel-level. For detection of objects such as buildings, however, good accuracy at the pixel-level does not necessarily reflect good accuracy at the object-level (e.g., two adjacent buildings being connected by a small amount of misclassified pixels). In order to evaluate the ability of semantic segmentation for building detection, we converted the individual segments into individual objects (bounding boxes). This also allowed us to use a consistent evaluation metric for the comparisons among LOCO, YOLO and semantic segmentation. Theoretically, if the results of semantic segmentation can accurately delineate building boundaries at the pixel level (i.e. finer scale), their corresponding bounding boxes (i.e. at coarser scale) should also remain accurate.

According to the evaluation statistics in Tables 2, 3 and 4, our proposed LOCO framework had about 5%-20% improvement in F1-scores compared to semantic segmentation. Within the group of semantic segmentation methods, overall the best

performance was achieved by DeepLabV3+ with binary labels.[3] It achieved similar statistics as YOLO-52, which were largely better than the original YOLO-13 baseline (about 20% improvements in F1-Score).

For both PSPNet and DeepLabV3+, the results were slightly better with binary labels than signed distance labels in terms of F1-scores (except PSPNet in Area 2). It is interesting that for PSPNet the precision decreased but the recall improved with signed distance labeling whereas for DeepLabV3+ both precision and recall dropped. Note that there could be a variety of ways to implement the signed distance function, such as different numbers of classes used to represent the distance intervals or different ways to divide the distances (e.g., equally). In our implementation, we tried to apply the same strategy used in (Yuan 2018), which had 128 classes on imagery with 0.3-meter resolution (we used 39 classes with 1-meter resolution). The effects of such modeling are beyond the scope of this paper but may be worth further exploring.

4.3.3. Comparison across small and large building footprints

Since the goal of the paper is to improve detection accuracy for smaller buildings, here we show separate precision, recall and F1-scores for small and large buildings. Since we used 32-meter as the threshold for the side length of bounding boxes when separating training data for small and large building footprints, the same 32-meter is used here to split small and large buildings for evaluation.

Figures 7,8 show the precision, recall and F1-scores for small and large buildings, respectively. Since some test areas did not have a sufficient number of large buildings, we merged the detected large buildings across the five areas to better calculate the statistics. There were in total 181 large buildings. Note that in the US the number of small buildings is in general far greater than the number of large buildings.

The results for small building detection are almost the same as for small and large buildings combined, which shows that the overall improvements achieved by LOCO can be represented by the improvements on detecting small buildings. For large buildings, the precision, recall and F1-scores for all the methods decreased. Specifically, the F1-scores dropped about 15–25% for LOCO-single and LOCO-vote, 30% for YOLO-52, 5–10% for YOLO-13 and 40–50% for semantic segmentation. Comparing the methods, LOCO still maintained the highest F1-scores among the methods, where LOCO-single was more balanced between precision and recall and LOCO-vote was higher on precision and lower on recall. Semantic segmentation methods suffered from very low precision due to many false positives which formed large segments in their classification

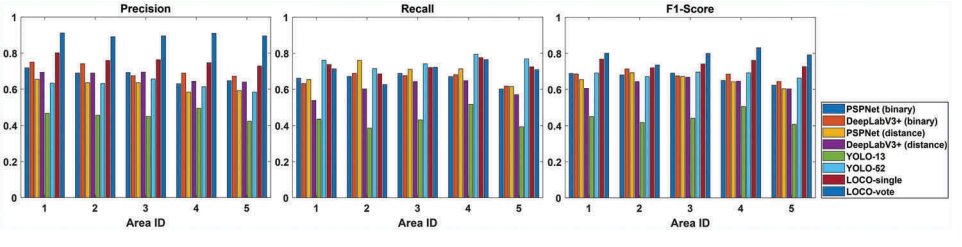

Figure 7. Comparison of precision, recall and F1-scores for small buildings. The results are very similar to the statistics for all buildings.

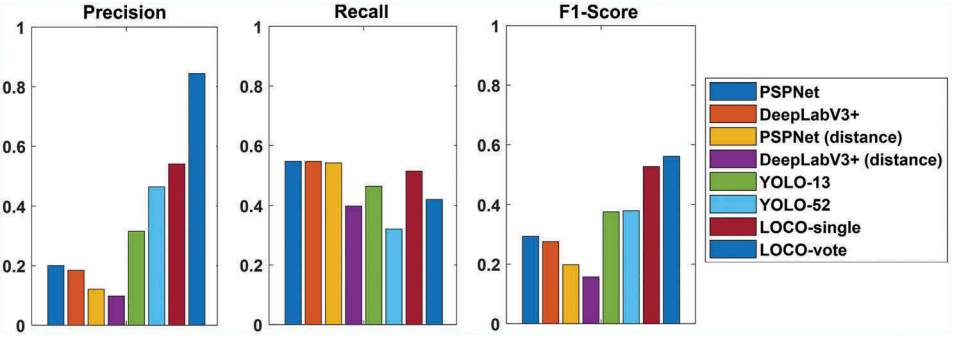

Figure 8. Comparison of precision, recall and F1-scores for large buildings. Compared to small buildings, the accuracy on large buildings decreased for all methods.

Figure 9. (a) shows a comparison of results in a local region: Ground truth in blue, LOCO and YOLO-13 in yellow, and DeepLabV3+ and PSPNet in red. The dashed boxes in orange and purple are two zoom-in windows, whose results are shown in (b). Visually we can see that the results of LOCO are in general closer to the ground truth compared to YOLO, and that it is still challenging for semantic segmentation to approximate the exact building boundaries.

results. The lower precision, recall and F1-scores for large buildings might be due to their smaller quantity in the US (e.g., fewer samples for both training and testing) and the greater variety between their styles. In contrast, small buildings (e.g., residential houses) normally follow certain standard designs and are much greater in quantity. From a different perspective, the smaller quantity of large buildings (at least in the US) may

potentially mitigate the lower accuracy of detection methods because a smaller number makes large buildings relatively faster to digitize when manual corrections and refinements are needed.

4.3.4. Qualititive comparison with visualization

Figure 9 provides a visualization of the detection results overlaid on top of the remote sensing images. In the figures we use LOCO-vote's results to represent the final outputs of our proposed work. For PSPNet and DeepLabV3+, we used their binary-label versions, which had better F1 scores over the versions with distance labels. Figure 9(a) shows the detailed results in local areas. To better visualize the differences, Figure 9(b) highlights the results in two zoomed-in windows, which correspond to the orange and purple dashed boxes in Figure 9(a). We can see that YOLO-13 results contain many missed and false detections, and LOCO results have a better visual match with the building footprints in the ground truth. For semantic segmentation, we can see that currently it is still difficult for these methods to satisfiably detect the exact boundaries of building footprints at meter-high resolution. These errors also affected the accuracy of building detection at coarser scales (i.e. when the segments were converted to bounding boxes of buildings).

5. Discussion

While this work shows the potential value of deep learning in geospatial object detection, there are still many challenges to be addressed and opportunities to be explored. Our discussion focuses on the following eight aspects.

Remote sensing images vs. everyday images: For remote sensing datasets, we found that the results of deep learning frameworks, while very encouraging, are still not as satisfying as they are for objects in everyday images. The difficulty may come from various aspects: (1) Building objects are often very small and have significantly fewer number of pixels compared to objects in everyday images. This coarse-resolution introduces more uncertainty when deciding the boundary of a building footprint; and (2) The training data of building footprints are mostly directly converted from existing feature or vector layers (e.g., GeoJSON, Shapefile), which are not really modified based on the exact imagery. This commonly leads to many types of noise, such as tree coverage, shadows, alignment (e.g., registration, camera angles), re-construction, etc. Such noise may have more impacts on the performance of pixel-level (i.e. finer scales) methods.

Two detection paradigms: Object-level and pixel-level deep learning frameworks are two paradigms that are related to object detection. The objects in pixel-level frameworks are created by grouping same-class pixels into connected segments. While pixel-level frameworks have been shown to produce near-approximations of exact boundaries of objects in everyday images, their results on building footprints are still not very satisfying. Thus, rather than considering the two paradigms as competing and mutually-exclusive, we envision an integrated method to potentially make the building detection problem easier. Our conjecture is that, once we have a good bounding box of a building footprint, it becomes easier to find the exact boundary of the footprint. Figure 10 shows some very preliminary results that demonstrate the idea. Each pair of images in Figure 10 shows a clipped building image patch using the detected bounding box and an

unsupervised segmentation of the pixels (only the largest segments were kept). The example results show that the problem becomes easier once we have good bounding boxes on the buildings, and we can potentially train a simpler pixel-level classification model to improve the boundary delineation. In addition, since semantic segmentation performs classification at pixel-level, it requires high-quality data of exact building boundaries in training. However, such data may not always exist in different applications, so in such scenarios it could also be helpful to start with the bounding box paradigm (e.g., require less manual efforts) and then move to fine-scale boundary delineation. We should also note that exact boundaries of buildings are not always needed in all types of applications (e.g., building counting and distribution analysis), so this information needs to be considered when selecting a suitable method.

Angle-aware bounding box: Like most bounding-box based methods, the proposed LOCO framework detects the orthogonal bounding boxes of objects. Since building footprints are not necessarily aligned with the orthogonal directions, it is meaningful to further tighten the bounding boxes by making the framework angle-aware. There are multiple strategies to achieve this goal and we discuss two of them. The first is to integrate with deep learning schemes that can predict rotation angle (e.g., (Liu *et al*. 2017a), (Liu *et al*. 2017b)). This will require new designs of network architecture and re-training. The second is to unsupervised augmentation Xie *et al*. (2018) (e.g., rotation-vector based or context-based) to estimate the angles of building footprints. For example, roads can be used as the context for building angle estimation. Unsupervised augmentation does not require training data to have angle information.

Interpretability: A critical issue and challenge in deep learning is its interpretability. This "black-box" nature of deep learning still needs to be addressed or reduced (Castelvecchi 2016). A first step might be to create good visualizations of the convolutional kernels for geospatial objects.

Flexibility: Currently, the LOCO framework is designed to work with well-integrated, standardized, one-meter resolution and large-scale (e.g., national) remote sensing imagery. This potentially reduces its flexibility when imagery with different standards are used. For example, NAIP imagery, which is one of the highest-quality open data at large scale, is collected by different contractors in different states in different years (NAIP Contractor Map 2009, 2011), leading to different imaging platforms and practices. Different states may also have different standards on location accuracy (NAIP Information Sheet 2015), which may further introduce inconsistency. More research are needed to understand the effects of different imaging practices on the performance

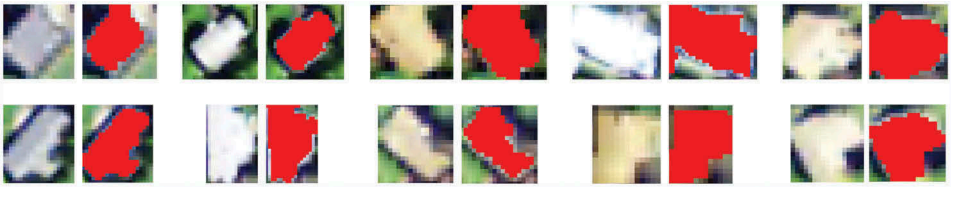

Figure 10. Preliminary results of pixel-level segmentation on image patches of individual buildings extracted using detected bounding boxes. The examples show that it might be easier to approximate exact boundaries of building footprints once their bounding boxes are available.

and flexibility of deep learning models. In addition, currently we use a hard threshold (e.g., 32 meter) to separate small and large building footprints. Soft-thresholding (e.g., fuzzy) or other branching techniques can be explored to improve the flexibility of LOCO. Finally, we may also explore a more flexible architecture that uses different resolutions and sizes of input images to detect buildings of different sizes.

Geographies: The main focus of this study was to show that the proposed analysis, designs and techniques can improve the ability of deep learning frameworks to detect small and densely distributed building footprints. We are not yet at the stage of providing a universally trained model that is general enough to use in all types of regions and landscapes. In fact, the effects of different geographies on the model's performance are not yet clear, and will likely require a significant amount of future research to study in detail.

Scale: Urban landscapes can differ significantly from one geographic area to another. For example, residential houses in the US are generally more sparse than those in Southeast Asia due to a lower population density. While the buildings may share many similar visual features, training a model with all landscapes together could potentially introduce more confusion than consensus. A similar situation in global agriculture monitoring (GEOGLAM 2018) led to the common use of locally-calibrated models to improve solution quality. Thus, future research is needed to determine the appropriate and applicable scales for deep learning models (e.g., state, country, continent or global level).

Integration: Due to the limitations of CPU and GPU memory, input images of convolution deep neural networks are mostly hundreds of pixels in size. This requires that every remote sensing image be clipped into 416×416 chunks. After detection, the results are then merged together. The main issue here is uncertainty. Unlike a normal merge operation on "true" data, the outputs from a model contain both missing and false detections, especially on the boundary of an image where objects are split into parts. For this reason, in our network design, we used a relatively large extent (e.g., 416 m by 416 m) to reduce the portion of buildings on the boundary while also allowing large buildings to be contained within the frame. In future work, it is necessary to study an integration scheme under the uncertainty of detection errors in deep learning frameworks.

6. Conclusions and future work

We investigated the use of deep learning in geospatial object detection using remote sensing datasets, and proposed a LOcally-COnstrained (LOCO) You-Only-Look-Once framework to detect small and densely distributed building footprints in satellite images. The design of LOCO is based on the analysis of spatial information propagation in convolutional deep neural networks. The LOCO framework avoids the excessive receptive fields in YOLO frameworks to reduce noise and facilitate learning for small building footprints. We further presented an invariant augmentation based voting scheme to improve the precision of detections. The voting scheme is part of the prediction phase, so it does not require any change in the training process and can be directly applied with minimal effort. Through experiments, we showed that LOCO can greatly improve F1 scores compared to related work. The results were also visualized to qualitatively validate the detections of LOCO.

In future work, we will continue exploring the new research opportunities discussed in Sec. 5. We are especially interested in further investigating the use of pixel-level learning models in image clips of individual buildings (Figure 10) to try to improve the accuracy on boundary delineation.

Notes

1. "small" is relative to the spatial range covered by the image, e.g., 500 meter by 500 meter.
2. 32 can be replaced by another value in other problems.
3. Note that at pixel-level, this model achieved high F1-scores (about 85% to 95%) on our validation dataset, which is a very good performance (e.g., (Yuan 2018)) based on the criteria of segmentation.

Acknowledgments

We would like to thank the reviewers and the members of the spatial computing research group at the University of Minnesota for their helpful comments on improving the quality of the paper. We also thank Kim Koffolt for improving the readability of this article.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the Advanced Research Projects Agency - Energy, U.S. Department of Energy [DE-AR0000795]; U.S. Department of Defense [HM0210-13-1-0005, HM1582-08-1-0017]; U.S. National Science Foundation [1737633, 0940818, 1029711,1541876, IIS-1218168, IIS-1320580]; U.S. Department of Agriculture [2017-51181-27222]; U.S. National Institute of Health [KL2 TR002492, TL1 TR002493, UL1 TR002494]; OVPR Infrastructure Investment Initiative, University of Minnesota; Minnesota Supercomputing Institute (MSI), University of Minnesota.

Notes on contributors

Yiqun Xie is a PhD candidate in the Department of Computer Science and Engineering at the University of Minnesota – Twin Cities. His research focuses on spatial data science, including spatial data mining, machine learning and optimization.

Jiannan Cai is a visiting PhD student in the Department of Computer Science and Engineering at the University of Minnesota – Twin Cities. He is also a PhD candidate in the Department of Geo-informatics at the Central South University, China. His research focuses on spatial data mining, spatial statistics and GIS.

Rahul Bhojwani is a master's student in the Data Science program at the University of Minnesota – Twin Cities. His research focuses on spatial data science and machine learning.

Prof. Shashi Shekhar is a McKnight Distinguished University Professor in the Department of Computer Science and Engineering at the University of Minnesota – Twin Cities. He is an IEEE Fellow and AAAS Fellow. Earlier, he served as the President of the University Consortium for GIS (UCGIS), and on many National Academies' committees. His research focuses on spatial data science, spatial databases and GIS.

Prof. Joseph Knight is an Associate Professor in the Department of Forest Resources, and the Director of the Remote Sensing and Geospatial Analysis Laboratory at the University of Minnesota – Twin Cities. His research focuses on geospatial science, remote sensing and GIS.

References

Castelvecchi, D., 2016. Can we open the black box of ai? *Nature News*, 538 (7623), 20. doi:10.1038/538020a

Chau, K.-W., *et al*., 2007. Determining optimal building height. *Urban Studies*, 44 (3), 591–607. doi:10.1080/00420980601131902

Chen, L.-C., *et al*., 2018. *Encoder- decoder with atrous separable convolution for semantic image segmentation*. arXiv preprint arXiv:1802.02611.

Cohen, J.P., *et al*., 2016. Rapid building detection using machine learning. *Applied Intelligence*, 45 (2), 443–457. doi:10.1007/s10489-016-0762-6

Compagnon, R., 2004. Solar and daylight availability in the urban fabric. *Energy and Buildings*, 36 (4), 321–328. doi:10.1016/j.enbuild.2004.01.009

Cui, S., *et al*. (2008), Building detection and recognition from high resolution remotely sensed imagery. *In*: *Proceedings of the XXIst ISPRS Congress*, Beijing, China, 411–416.

Cybenko, G., 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2 (4), 303–314. doi:10.1007/BF02551274

Dumoulin, V. and Visin, F., 2016.*A guide to convolution arithmetic for deep learning*. arXiv preprint arXiv:1603.07285.

GEOGLAM. Available from: http://www.geoglam.org. [Accessed May 2019].

Goodfellow, I., *et al*., 2016. *Deep learning*. Vol. 1. Cambridge: MIT press.

Joshi, K.K. and Kono, T., 2009. Optimization of floor area ratio regulation in a growing city. *Regional Science and Urban Economics*, 39 (4), 502–511. doi:10.1016/j.regsciurbeco.2009.02.001

Konstantinidis, D., *et al*., 2017. Building detection using enhanced hog–lbp features and region refinement processes. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10 (3), 888–905. doi:10.1109/JSTARS.2016.2602439

Krizhevsky, A., Sutskever, I., and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. *In*: *Advances in neural information processing systems*. Lake Tahoe, NV, USA, 1097–1105.

Kubota, T., *et al*., 2008. Wind tunnel tests on the relationship between building density and pedestrian-level wind velocity: development of guidelines for realizing acceptable wind environment in residential neighborhoods. *Building and Environment*, 43 (10), 1699–1708. doi:10.1016/j.buildenv.2007.10.015

Kussul, N., *et al*., 2017. Deep learning classification of land cover and crop types using remote sensing data. *IEEE Geoscience and Remote Sensing Letters*, 14 (5), 778–782. doi:10.1109/LGRS.2017.2681128

LeCun, Y., Bengio, Y., and Hinton, G., 2015. Deep learning. *Nature*, 521 (7553), 436. doi:10.1038/nature14539

Lin, C. and Nevatia, R., 1998. Building detection and description from a single intensity image. *Computer Vision and Image Understanding*, 72 (2), 101–121. doi:10.1006/cviu.1998.0724

Liu, L., Pan, Z., and Lei, B., 2017a. *Learning a rotation invariant detector with rotatable bounding box*. arXiv preprint arXiv:1711.09405.

Liu, Z., *et al*. (2017b), Rotated region based cnn for ship detection. *In*: Image Processing (ICIP), 2017 IEEE international conference on. Beijing, China, 900–904.

Locke, D.H., *et al*., 2014. Urban environmental stewardship and changes in vegetative cover and building footprint in new york city neighborhoods (2000–2010). *Journal of Environmental Studies and Sciences*, 4 (3), 250–262. doi:10.1007/s13412-014-0176-x

Luo, W., *et al*., 2016. Understanding the effective receptive field in deep convolutional neural networks. *In*: *Advances in neural information processing systems*. Barcelona, Spain, 4898–4906.

Massachusetts Buildings Dataset, 2013. https://www.cs.toronto.edu/~vmnih/data/. [Accessed Nov 2018].

NAIP Contractor Map, 2009. https://www.fsa.usda.gov/Assets/USDA-FSA-Public/usdafiles/APFO/Others/2009-contractor.gif.[Accessed Dec 2018].

NAIP Contractor Map, 2011. https://www.fsa.usda.gov/Internet/FSA_Image/2011_contractors.jpg. [Accessed Dec 2018].

NAIP Information Sheet, 2015. https://www.fsa.usda.gov/Internet/FSA_File/naip_info_sheet_2015.pdf. [Accessed Dec 2018].

National Agriculture Imagery Program. https://www.fsa.usda. gov/programs-and-services/aerial-photography/imagery-programs/naip-imagery/. [Accessed May 2019].

Nogueira, K., Penatti, O.A., and Dos Santos, J.A., 2017. Towards better exploiting convolutional neural networks for remote sensing scene classification. *Pattern Recognition*, 61, 539–556. doi:10.1016/j.patcog.2016.07.001

Razavian, A.S., *et al*. (2014), Cnn features off- the-shelf: an astounding baseline for recognition. *In*: Computer vision and pattern recognition workshops (CVPRW), 2014 IEEE conference on. Columbus, OH, USA, pp. 512–519.

Redmon, J., *et al*. (2016), You only look once: unified, real-time object detection. *In*: Proceedings of the IEEE conference on computer vision and pattern recognition. Las Vegas, NV, USA, 79–788.

Redmon, J. and Farhadi, A. (2017), Yolo9000: better, faster, stronger. *In*: 'Computer vision and pattern recognition (CVPR), 2017 IEEE conference on. Honolulu, HI, USA, 6517–6525.

Romero, A., Gatta, C., and Camps-Valls, G., 2016. Unsupervised deep feature extraction for remote sensing image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 54 (3), 1349–1362. doi:10.1109/TGRS.2015.2478379

Rylatt, M., Gadsden, S., and Lomas, K., 2001. Gis-based decision support for solar energy planning in urban environments. *Computers, Environment and Urban Systems*, 25 (6), 579–603. doi:10.1016/S0198-9715(00)00032-6

Shin, H.-C., *et al*., 2016. Deep convolutional neural networks for computer-aided detection: cnn architectures, dataset characteristics and transfer learning. *IEEE Transactions on Medical Imaging*, 35 (5), 1285–1298. doi:10.1109/TMI.2016.2528162

Sirmacek, B. and Unsalan, C., 2009. Urban-area and building detection using sift keypoints and graph theory. *IEEE Transactions on Geoscience and Remote Sensing*, 47 (4), 1156–1167. doi:10.1109/TGRS.2008.2008440

Wu, S.S., Wang, L., and Qiu, X., 2018. Incorporating gis building data and census housing statistics for sub-block-level population estimation. *The Professional Geographer*, 60 (1), 121–135. doi:10.1080/00330120701724251

Xie, Y., *et al*. (2018), An unsupervised augmentation framework for deep learning based geospatial object detection: a summary of results. *In*: Proceedings of the 26th ACM SIGSPATIAL international conference on advances in geographic information systems. Seattle, WA, USA, 349–358.

Yuan, J., 2018. Learning building extraction in aerial scenes with convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40 (11), 2793–2798. doi:10.1109/TPAMI.2017.2750680

Zhao, H., *et al*. (2017), Pyramid scene parsing network. *In*: IEEE Conf. on computer vision and pattern recognition (CVPR), Honolulu, HI, USA, 2881–2890.

Appendix A. Regression modeling in YOLO

In the YOLO framework, the original input size of each image is convoluted and pooled to a spatially-coarser grid. Thus, a typical input size of $416 \times 416 \times 3$ (row $\times$ column $\times$ channel) finally becomes $13 \times 13 \times R$ (row $\times$ column $\times$ number of predicted parameters). In this output layer, the cell that contains the center of an object is responsible for predicting its location and size.

To understand the regression modeling, first we need to understand the four parameters representing the locations and sizes of predicted bounding boxes, that is, the center coordinates c_x and c_y, width l_w and height l_h. These parameter values are not directly given in the output layer. Instead, they are further derived afterwards.

Center coordinates: Since the cell that contains the object's center is responsible for detecting it, the center coordinates of an object are modeled as offsets to the cell's top-left corner (minimum x and y). Because the center is contained by the cell, the predicted offsets are values in the range $[0, 1]$.

Width and height: YOLO models width and height using an anchor box. An anchor box represents a common size (width and height) of objects in the images. For example, one can use the average width and height of all objects to build a common size. Then, the actual width and height of any object is modeled as a scaling of the common width and height.

Equation (A1) shows the detailed derivation of the four parameters:

$$\begin{aligned} c_x &= cell_x + \sigma(p_x) \\ c_y &= cell_y + \sigma(p_y) \\ l_w &= anchor_w \cdot e^{p_w} \\ l_h &= anchor_h \cdot e^{p_h} \end{aligned} \tag{A1}$$

where $cell_x$ and $cell_y$ are the top-left coordinates of a cell; $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function (always outputs value in $(0, 1)$); $anchor_w$ and $anchor_h$ are the width and height of the anchor box; and p_x, p_y, p_w and p_h are the predicted values in the output layer of the neural network.

The overall structure of the parameters is: {obj, c_x, c_y, l_w, l_h, $class_1$, $class_2$, …, $class_C$}. The number of predicted parameters R depends on two settings: (1) the number of object classes in the problem; and (2) the number of anchor boxes. We can already see the effect of class number on the number of parameters. In terms of the anchor boxes, YOLO uses multiple anchor boxes (i.e. multiple common sizes) rather than just one, and it predicts a set of parameters for each anchor box. Thus, its parameter structure becomes: {anchor$_1$:{obj, c_x, c_y, l_w, l_h, $class_1$, $class_2$, …, $class_z$}; anchor$_2$:{obj, c_x, c_y, l_w, l_h, $class_1$, $class_2$, …, $class_z$}; …; anchor$_A$:{obj, c_x, c_y, l_w, l_h, $class_1$, $class_2$, …, $class_z$}}. The reason for having multiple anchor boxes is to address the scenario where a grid cell in the output layer contains multiple centers of overlapping objects. In YOLO, the number of anchor boxes is set to 5. With this multi-anchor-box modeling, each grid cell can predict as many as 5 boxes instead of only one. The total number of parameters predicted by each grid cell is then: $R = (5 + number\_class) \cdot number\_anchor$.