Global Scale Data Management for Building Detection from Satellite Images

David Yang
dzyang@fb.com
Spatial Computing, Facebook
USA

Yao-Yi Chiang
Spatial Sciences Institute,
University of Southern California
USA
yaoyic@usc.edu

Xiaoming Gao, Xiang Gao,
Saikat Basu
[gaoxm,ethanssr,saikatbasu]@fb.com
Spatial Computing, Facebook
USA

ABSTRACT

The capability to automatically extract objects, such as building outlines, from satellite imagery at global scale can create comprehensive map data that support many applications. Existing approaches for object detection from satellite imagery focus on algorithm development for the machine learning model but often overlook the important data management component. This paper presents an in-production data management system, a useful key enabling an efficient data pipeline for training, inferencing, and postprocessing for building outline detection from satellite imagery on a global scale. The case study showcases the data management system's performance for building outline detection in four countries.

KEYWORDS

imagery recognition, building detection, spatial data management, global scale

ACM Reference Format:
David Yang, Yao-Yi Chiang, and Xiaoming Gao, Xiang Gao, Saikat Basu. 2021. Global Scale Data Management for Building Detection from Satellite Images. In *SIGMOD '21: ACM SIGMOD/PODS International Conference on Management of Data, June 20–25, 2021, Xi'an, Shaanxi, China.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Real-time, accurate, and complete map data is the key to the success of many applications, such as autonomous driving cars, humanitarian missions, and natural disaster management (e.g., [8]). For example, map data of road networks can help governments at all levels to plan, measure, and improve operations and guide policies (e.g., measuring the distance to health facilities from individual house locations in Malawi [9]). Building footprints data allow population density estimation (e.g., [14]) and natural disaster management. Map data also have huge commercial value. For example, with detailed point-of-interest (POI) information, map data allow people to search for events, job listings, and POIs based on their locations and, in turn, recommend venues of interest (e.g., [2]).

Unfortunately, traditional survey methods for generating authoritative map data are time-consuming, require a considerable amount of resources, and often cannot provide up-to-date information. Recent success in volunteered geographic information (VGI) has improved the availability of map data [13] (e.g., OpenStreetMap). However, the coverage and timeliness of map data are still insufficient for many regions around the globe, especially for areas where map data can change frequently (e.g., seasonal roads and refugee camps). Artificial Intelligence (AI) technologies, such as deep learning models, have demonstrated numerous successful computer vision tasks in image understanding, including object detection and semantic segmentation. Nevertheless, collecting a large number of labeled training data for feature detection from satellite imagery (e.g., extracting building outlines) on a global scale is still challenging. The fact that spatial non-stationarity exists in the imagery (e.g., building rooftops can vary significantly from one region to another) means that the training data should be as diversified geographically as possible, but manually creating these training data is not feasible. Using contextual data (e.g., OpenStreetMap) to collect training data automatically (e.g., [14]) can overcome this challenge. The idea is that spatial data from different sources (e.g., buildings on OpenStreetMap and imagery) can depict the same Earth entity linked by the geocoordinates of the data. Therefore, using available spatial data can help identify the same entity's likely location in another data source. For example, although building outlines from OpenStreetMap might not be complete, we can use them to annotate possible building locations in satellite imagery to create training data and then train a deep learning model for recognizing all buildings in the same area from the imagery (e.g., [14]). Similar context-based approaches exist in recognizing geographic features from scanned historical maps (e.g., [4] and [3]). When processing datasets with global coverage, this context-based approach requires careful consideration in managing large contextual datasets, automatically collected training data, satellite imagery, and the detection results.

This paper introduces a data management system with an end-to-end data pipeline for storing and querying the large-scale contextual data and imagery for the training and inferencing processes of deep learning models for building outline detection from satellite imagery. The paper describes various components in the data pipeline and demonstrates the results in four countries. The data management system is currently in use to generate building outlines on a global scale.

SIGMOD '21, June 20–25, 2021, Xi'an, Shaanxi, China

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

2 RELATED WORK

Ever since the launch of the Landsat satellites in the 1970s, there is an abundance of approaches and applications for object detection from satellite imagery (i.e., remote sensing). In addition, recent advances in deep learning neural networks have enabled high accuracy machine learning models. However, most state-of-the-art building detection approaches focus on developing and testing machine learning models for small areas or using a single set of training data for every target region (e.g., the D-LinkNet [15] and Hourglass model [1, 11] models in our system) (see [7] for a recent survey). The result is that the trained model would not work well across geographical areas. One reason is that because of the inherent spatial non-stationarity property (e.g., the visual appearance of building rooftops can vary significantly from one area to another), the training data should be (spatially) as close to the target area as possible. Using a context-based approach to generate labeled training data can overcome this difficulty, but managing the contextual data and training data for efficient training and inferencing of machine learning models is still challenging. Our system integrates a number of specialized data management components and spatial index techniques to achieve this task.

3 DATA MANAGEMENT SYSTEM FOR GLOBAL SCALE BUILDING DETECTION

This section presents our data management system for global building detection (Figure 1). The data management system supports efficient storage and query of geospatial data in the vector and raster formats. A Geospatial Indexing service, which is built on top of Hive tables, is responsible for real-time queries of the vector datasets. A Geospatial Blob Storage system, which is built on top of a key-value database for storing binary large objects (BLOB) [10], is in charge of handling the raster datasets accessible through the <x,y,z> coordinates (i.e., web map tiles).

In general, the system stores the contextual data, building polygons from OpenStreetMap (OSM), in a Hive table with spatial indexes generated from the Geospatial Indexing service. The system processes the input imagery, satellite imagery from Maxar, to generate map tiles and stores them in the Geospatial Blob Storage system. Both contextual and input data have global coverage. The Training component can access the Hive table and the Geospatial Blob Storage system to collect training data (Section 3.1) and train convolutional neural network (CNN) models (Section 3.2). The Inference component uses the trained CNN models to segment building footprints from the imagery tiles and then stores the predicted raster building footprints in the Geospatial Blob Storage system (Section 3.3). Next, the Polygonization component converts the raster building footprints into vector polygons and then stores the polygons in an output Hive table indexed with the Geospatial Indexing service (Section 3.4). Finally, the Conflation component conflates the result polygons with an existing map, such as OSM, to identify unmapped buildings (Section 3.5). These unmapped buildings can be incorporated in a map editor for human editing (Section 3.6). We describe the entire system in detail in the following sections.

3.1 Collecting Training Data

For building footprint segmentation using CNNs, we need to collect large and diversified training datasets automatically. In our current

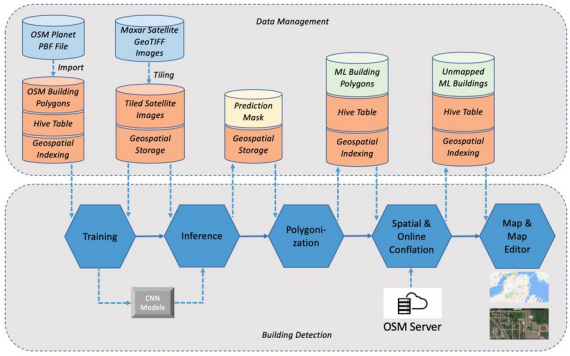

Figure 1: Architecture of the data management system for global building detection

data pipeline, the building polygons from OSM serve as the contextual data for automatic training data collection. OSM is a free, editable map covering the entire world and is built by volunteers largely from scratch. OSM is licensed under the Open Database License [12]. We first download the latest OSM PBF (Protocolbuffer Binary Format) files that contain OSM's existing map data for the entire planet, including building polygons. Then the system parses the PBF files to extract all building labels as WKT (Well-Known-Text) polygons and uploads them to a Hive table. The Geospatial Indexing service is a high-performance, multi-customer/multi-use, indexing service to enable efficient and large scale storing, caching, and querying of spatial data in Hive. The indexing service uses the common multi-level nested R-tree structure. The parent R-tree include the bounding boxes of the child R-trees, and the leaf R-tree include the building polygon data. For a bounding box query, the system first uses the parent R-tree to find all child R-trees that have an intersection with the bounding box. Then search continues recursively within the child R-tree until it reaches all related leaves, which contain the intersecting polygons.

In addition to the contextual data, the other input for training data collection contains high-resolution satellite imagery from Maxar, a collection of approximately 1,500,000 images in the GeoTIFF format with a total size of 1.5PB. These images were taken from multiple passes of satellites over a period of a few years, with specific images chosen to minimize cloud cover while also being as recent as possible. Each image covers an area of $10km$ by $10km$ on the ground. Each pixel in the images roughly represents an area of approximately $50cm$ by $50cm$. These images are rather large and using them directly for training and inference is inefficient. Our solution is to slice them into map tiles, and index the tiles by their geocoordinates (x and y) and zoom level (z) in the Geospatial Blob Storage system. This way, we can process multiple tiles in parallel on multiple machines (workers) and minimize the transmission of large files over the network. The tiled images have the same image dimension as the Bing map tile at zoom level 15, equivalent to an area of $1km$ by $1km$ on the ground. The tiled images use the (pseudo) Mercator projection and have a dimension of 2,048 by 2,048 pixels. The system stores the tiled images in a Geospatial Blob Storage system in which each tile is indexed and can be accessed using their geocoordinates and zoom level (following the convention of Bing map tiles). The underlying storage of the Geospatial Blob Storage

system is a key-value database providing a persistent, highly available, and distributed online solution for large geospatial datasets. The key represents the tile's coordinates. The system stores the mapping between the keys and the tile coordinates in a relational database for efficient tile search. The values represent the physical location of the image tile. Each dataset in the storage system can store up to 1 billion image tiles.

3.2 Training CNN Models

The Training component uses image tiles of automatically collected training data and satellite imagery in the Geospatial Blob Storage system to train two separate convolutional networks, D-LinkNet [15] and Hourglass [11] [1] models, for building detection. The D-LinkNet model is a semantic segmentation neural network built with the LinkNet encoder-decoder architecture and includes dilated convolution layers. It is as efficient in computation and memory consumption as the LinkNet model. The dilation convolution in the D-LinkNet model is a powerful tool that can facilitate building detection of different scales by enlarging the receptive field without reducing the resolution of the feature maps. The Hourglass model processes image features across all scales and consolidates them to best capture various spatial relationships associated with the buildings of different scales. The Hourglass model repeats bottom-up, top-down processing, and implements intermediate supervision. The model earns its name based on the successive steps of pooling and upsampling to produce a final set of predictions. For both networks, we use group normalization.

3.3 Inference of Building Footprints

After training, the Inference component uses the trained model to infer building footprints across the globe. The inference process is distributed over multiple graphics processing units (GPUs), and each GPU performs inferencing on the tiled images of zoom level 15 retrieved from the Geospatial Blob Storage system. The tile coordinates of the satellite images are stored in a thread-safe queue, and each GPU worker queries the queue for the next tile for processing. The queue is implemented using a relational in-memory database. The inference results, building-footprint segmentation masks, are uploaded to a new dataset in the Geospatial Blob Storage system.

3.4 Polygonization

Given the building-footprint segmentation masks from the Inference component, the next step is to convert the results in the raster format into polygons in the vector format. Here, the Polygonization component first converts the segmentation masks into a binary image using a pre-defined threshold and then vectorizes the boundary pixels of each connected component in the binary image. If a building crosses the boundary of a zoom-level-15 tile, this polygonization step would generate a partial building footprint. To reduce the number of buildings at the tile boundaries, our algorithm works on image tiles at zoom level 13, equivalent to 16 zoom-level-15 tiles. For each zoom-level-13 tile, the corresponding zoom-level-15 tiles are retrieved and stitched together to compose one large image. The resulting polygons are uploaded to a Hive table that is also indexed using the Geospatial Indexing service.

3.5 Spatial Conflation

The last step is to conflate the machine learning (ML) generated building polygons with the existing buildings on a map, with the goal of filtering out the ML-generated polygons that are already mapped. For this purpose, the Conflation component employs a distributed spatial conflation algorithm to detect overlaps between the ML-generated buildings and existing buildings stored in a Hive table.

First, the Conflation component creates a tree partitioning scheme that divides the space into non-overlapping rectangles such that the number of buildings in each leaf partition is about the same. The spatial conflation algorithm uses the leaf partitions to distribute the computing load for calculating the intersections between the ML buildings and existing buildings across multiple machines and combines the results from each machine. Each machine processes a subset of ML and existing buildings in a few partitions. After leaving out the ML building polygons overlapping with existing buildings, the algorithm uploads the unmapped ML buildings into an Hive table. The tree partitioning scheme used in this process comes from the Presto query engine, which is based on Spatial Join Techniques [6].

3.6 Online Conflation

One potential application of the ML-generated building polygons is to incorporate them into map editors so that human mappers can do postprocessing on the data. For this, the Online Conflation service is introduced in the data management system. We use an open-source map editor called RapiD as an example to illustrate how the service works. RapiD (Figure 2), is a modified version of the popular OSM editing tool, the iD editor. RapiD enables the mapping community to enjoy a fast and accurate mapping experience with pre-populated map features detected from high-resolution satellite imagery, including roads and buildings[5]. When a mapper opens RapiD and starts to edit an area, the Online Conflation service queries the ML-generated buildings within this area using the Geospatial Indexing service on top of the Hive table. Meanwhile, RapiD queries the OSM buildings from the OSM live servers for the same area. The Online Conflation service also uses an R-tree to detect and filter out the ML-generated buildings that overlap with buildings from the live OSM data. Depending on the area size, the number of buildings in the conflation result could range from a few hundred to a few thousand. Finally, the service sends the unmapped buildings to RapiD for display. The Online Conflation service can be applied to any other map editors that are capable of incorporating additional data sources, such as the map editor JOSM.

Figure 2: Open source map editor RapiD.

RapiD enables human mappers to validate the ML-generated map features against satellite images and check them into the OSM database if the features match the ground truth, or make edits on the features if post-processing is needed. Together with the Online Conflation service, RapiD releases human mappers from the time-consuming and tedious process of manual tracing of map features.

4 EXPERIMENT

This section discusses our results of global scale building detection. First, we used the tiling process to generate 153,676,895 satellite images (zoom level 15) for the entire globe from Maxar's GeoTIFF images. The process took about three weeks for 100 workers to tile the entire planet, including downloading and uploading data. We downloaded the PBF files containing a snapshot of OSM map data on 2020-09-01, with a total size of 50*GB*, from OSM. A total of 1.24 billion building labels were imported from these PBF files across 6,585,127 Bing tiles of zoom level 15. From these tiles, 545,000 tiles were selected as the training dataset based on two criteria. First, at least 200 tiles were selected from each country so that the training data would have sufficient country diversity. Secondly, the number of tiles from each country was limited to 20,000 to prevent the tiles of any single country from having an unwanted high weight in the training dataset.

The OSM and Maxar datasets were used to train the CNN models for segmenting building footprints. The satellite imagery for each zoom-level-15 tile has a size of 2048 x 2048 pixels. We randomly cropped the imagery tile to 1024 x 1024 pixels at training time to reduce the computation cost and then applied mean subtraction. We trained the networks with a batch size of 64 for 150 epochs. We used Pytorch's Distributed Training package to train the model on 64 GPUs. We used the SGD (stochastic gradient descent) optimizer with momentum = 0.9, weight decay = 0.0001, and an initial learning rate of 0.093 with step scheduler having a drop factor of 10 at epochs 30, 60, 90.

Due to the large data size, the satellite imagery and building labels would not fit into the main memory. Instead, the tile coordinates to satellite imagery and building labels was loaded into the main memory at the beginning of the training. During each minibatch training, the tile coordinates of the minibatch samples were used to fetch the satellite images from the Geospatial Blob Storage system. The tile coordinates were also converted to bounding boxes for querying the Geospatial Indexing service to fetch building labels from the Hive table. The building polygons were then rasterized to create the same size training label images as the corresponding imagery tile. This way, only the satellite imagery and training labels of the current minibatch need to be loaded into the GPU memory at a time. The model training typically took about 90 hours. Figure 3 shows a sample of the training data, the satellite imagery, and its corresponding building labels. Figure 4 shows the results obtained by the Hourglass model in a suburban area near Boston, USA.

We generated building data for four countries to showcase the data management system's performance: Pakistan, Nigeria, Ghana, and Paraguay. The inference process includes three steps, reading the satellite imagery tiles from the Geospatial Blob Storage system, applying the model on the tiled images to obtain building prediction masks, and writing the prediction masks back to the Geospatial

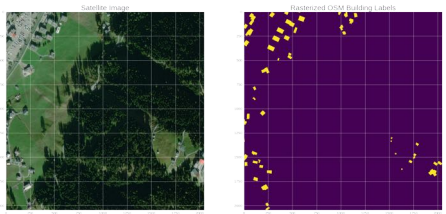

Figure 3: Left: Maxar satellite image for training; right: corresponding rasterized OSM building labels.

Figure 4: (a) The Maxar satellite image of a Boston suburban neighborhood. (b) The raw building segmentation. (c) The thresholded building segmentation. (d) The building polygons obtained from the Polygonization process.

Blob Storage system. These four countries have a total of 2,710,528 zoom-level-15 tiles, and it took about 12 hours for 200 GPU workers to complete the inference. By estimation, it would take about 40 days for 200 GPU workers to finish the entire planet.

The four countries have 169,408 zoom-level-13 tiles in total. The polygonization process distributed these tiles to 400 CPU workers and took about 27 hours to finish. By estimation, it will take roughly 85 days to polygonize the entire globe using the same amount of computational resources. The polygonization process generated 52,082,528 building polygons for the four countries. Next, the polygons were conflated against buildings in OSM using the Spaptial Conflation. The OSM snapshot used in our experiment had 419,423,031 buildings for the entire globe. The conflation process took about 9 minutes. Estimating proportionally, the process will take roughly 11.5 hours to conflate the entire globe using the same amount of computational resources. The conflation process found 6,161,415 (11.8%) ML buildings overlapping with the OSM buildings. The rest of 45,921,113 (88.2%) ML-generated buildings were not mapped before.

5 DISCUSSION AND FUTURE WORK

This paper presents an in-production data management system that enables building detection from satellite imagery at global scale. We plan to add support for more types of map features to the system and enable the detection of different feature types in parallel.

REFERENCES

[1] Anil Batra, Suriya Singh, Guan Pang, Saikat Basu, C.V. Jawahar, and Manohar Paluri. 2019. Improved Road Connectivity by Joint Learning of Orientation and Segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[2] Buru Chang, Yonggyu Park, Donghyeon Park, Seongsoon Kim, and Jaewoo Kang. 2018. Content-Aware Hierarchical Point-of-Interest Embedding Model for Successive POI Recommendation.. In *IJCAI*. 3301–3307.

[3] Yao-Yi Chiang, Weiwei Duan, Stefan Leyk, Johannes H Uhl, and Craig A Knoblock. 2020. Training Deep Learning Models for Geographic Feature Recognition from Historical Maps. In *Using Historical Maps in Scientific Studies*. Springer, 65–98.

[4] Weiwei Duan, Y Chiang, Crang A Knoblock, Stefan Leyk, and J Uhl. 2018. Automatic generation of precisely delineated geographic features from georeferenced historical maps using deep learning. In *Proceedings of the AutoCarto*.

[5] Facebook. [n.d.]. Map With AI. https://mapwith.ai.

[6] Edwin H Jacox and Hanan Samet. 2007. Spatial join techniques. *ACM Transactions on Database Systems (TODS)* 32, 1 (2007), 7–es.

[7] Ke Li, Gang Wan, Gong Cheng, Liqiu Meng, and Junwei Han. 2020. Object detection in optical remote sensing images: A survey and a new benchmark. *ISPRS Journal of Photogrammetry and Remote Sensing* 159 (2020), 296–307.

[8] Sophia B Liu. 2014. Crisis crowdsourcing framework: Designing strategic configurations of crowdsourcing for the emergency management domain. *Computer Supported Cooperative Work (CSCW)* 23, 4-6 (2014), 389–443.

[9] Terhi J Lohela, Oona MR Campbell, and Sabine Gabrysch. 2012. Distance to care, facility delivery and early neonatal mortality in Malawi and Zambia. *PloS one* 7, 12 (2012), e52110.

[10] Subramanian Muralidhar, Wyatt Lloyd, Sabyasachi Roy, Cory Hill, Ernest Lin, Weiwen Liu, Satadru Pan, Shiva Shankar, Viswanath Sivakumar, Linpeng Tang, and Sanjeev Kumar. 2014. f4: Facebook's Warm BLOB Storage System. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*. USENIX Association, Broomfield, CO, 383–398. https://www.usenix.org/conference/osdi14/technical-sessions/presentation/muralidhar

[11] Alejandro Newell, Kaiyu Yang, and Jia Deng. 2016. Stacked Hourglass Networks for Human Pose Estimation. *CoRR* abs/1603.06937 (2016). arXiv:1603.06937 http://arxiv.org/abs/1603.06937

[12] OpenStreetMap. [n.d.]. Open Database License. https://wiki.openstreetmap.org/wiki/Open_Database_License.

[13] Hansi Senaratne, Amin Mobasheri, Ahmed Loai Ali, Cristina Capineri, and Mordechai (Muki) Haklay. 2017. A review of volunteered geographic information quality assessment methods. *International Journal of Geographical Information Science* 31, 1 (2017), 139–167. https://doi.org/10.1080/13658816.2016.1189556 arXiv:https://doi.org/10.1080/13658816.2016.1189556

[14] Tobias G. Tiecke, Xianming Liu, Amy Zhang, Andreas Gros, Nan Li, Gregory Yetman, Talip Kilic, Siobhan Murray, Brian Blankespoor, Espen B. Prydz, and Hai-Anh H. Dang. 2017. Mapping the world population one building at a time. *CoRR* abs/1712.05839 (2017). arXiv:1712.05839 http://arxiv.org/abs/1712.05839

[15] Lichen Zhou, Chuang Zhang, and Ming Wu. 2018. D-LinkNet: LinkNet with Pretrained Encoder and Dilated Convolution for High Resolution Satellite Imagery Road Extraction. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2018), 192–1924.