

A Framework for Collecting and Classifying Objects in Satellite Imagery^{*}

A. Radhakrishnan¹, J. Cunningham¹, J. Davis¹, and R. Ilin²

¹ Ohio State University, Columbus OH 43210

² AFRL/RYP, Wright-Patterson AFB OH 45433

{radhakrishnan.39, cunningham.844, davis.1719}@osu.edu,
rilin325@gmail.com

Abstract. A major issue with data-hungry deep learning algorithms is the lack of annotated ground truth for specific applications. The high volume of satellite imagery available today, coupled with crowd-sourced map data can enable a new means for training and classifying objects in wide-area imagery. In this work, we present an automated pipeline for collecting and labeling satellite imagery to facilitate building custom deep learning models. We demonstrate this approach by automatically collecting labeled imagery of solar power plants and building a classifier to detect the structures. This framework can be used to collect labeled satellite imagery of any object mapped by spatial databases.

Keywords: Satellite Imagery · Data Generation · Automated Ground Truthing · OpenStreetMap · Deep Learning · Solar Power Plants

1 Introduction

Advancements in deep learning for image classification, segmentation, and object detection are creating profound changes in methods for analysis and inference of imagery. Supervised machine learning tasks are highly dependent on the availability of large, labeled training datasets. Hence, there is always a need to find a dataset which is sufficiently large enough to train a model and broad enough to contain diverse data samples to ensure better generalization performance. Satellite imagery is one such source of expansive and rich data.

Remote sensing is an Earth observation technique by which information about a feature on the Earth's surface is gathered without making physical contact. Earth Observation (EO) satellites are launched into space with their primary mission being remote sensing and providing satellite imagery of the Earth's surface from low Earth orbits. The number of EO satellites has increased exponentially over recent years. The data from these satellites have various applications in fields such as environmental monitoring, disaster management, agricultural engineering, cartography, and military intelligence. EO satellites have

^{*} This work was supported by the U.S. Air Force Research Laboratory under contract #GRT00044839/60056946.

instruments capable of sensing a wide range of bands in the electromagnetic spectrum. Another important property of satellite imagery is the frequency in which new data is captured. A primary advantage of satellite imagery is that archived imagery of multiple spectral bands at high resolution can be collected and processed.

Although there has been a considerable increase in the volume of publicly available satellite imagery, it remains difficult to find a dataset that contains examples of a specific object for a particular task. Similarly, generating new datasets leads to the additional overhead of annotation. Manual annotation is a time-consuming task that becomes difficult with large amounts of imagery. However, public geographical databases and crowd-sourced map information can be used to automatically locate objects and label the ground truth. In this work, we present a generic framework for extracting satellite imagery of specified object categories using public databases. We demonstrate the approach by automatically extracting satellite imagery of solar power plants (though other object classes can be used) from around the world and building a classifier to detect them in novel satellite imagery.

2 Related Work

The availability of satellite imagery has already promoted the use of deep learning algorithms for analyzing remote sensing data. In [1], they focused on predicting land use (cropland, residential, forest, etc.) in satellite imagery of urban areas. However, they were restricted to train with a small, manually-created dataset for 6 European cities. Similarly, in [9] a classifier was trained to predict crop types using satellite imagery having manual ground truth. In [6], solar power plants were detected in geographically restricted satellite imagery and the ground truth classes were manually annotated. Also, there was a large bias between the number of positive and negative examples. Crowd-sourced map information has become widely available today and OpenStreetMap (OSM) [18] is one such popular crowd-sourced spatial database primarily used for the purpose of land use mapping. In [7], they used existing OSM datasets for labeling water, farm, grass, etc. on Landsat imagery. In [15], OSM data were used for filling gaps in existing land cover maps. In [2], OSM raster images were used to superimpose land use labels on satellite imagery. Similarly, in [8], OSM raster data were used to superimpose labels of roads and buildings on existing satellite imagery to perform semantic image segmentation. In [20], OSM data were integrated with high spatial resolution imagery to classify types of buildings.

Overall, most of these approaches suffered from the lack of large problem-specific labeled training examples, which has been a bottleneck for applying deep learning techniques. Also, those employing OSM data used existing filtered OSM datasets. Our proposed framework can dynamically query the OSM database for the required data thus bypassing having to download and filter the entire OSM database. Additionally, our proposed model extracts OSM data in vector format allowing us to work with the object boundaries (points, lines, or polygons). We

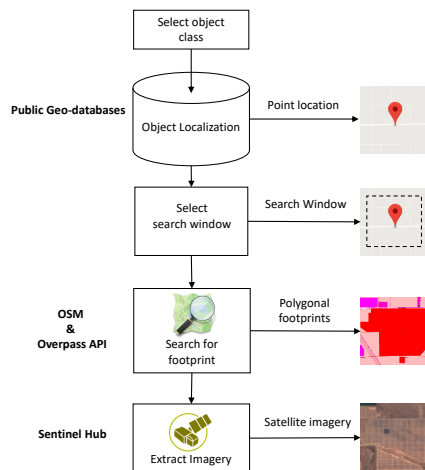


Fig. 1: Data generation pipeline architecture.

can also exploit the broader categories of features mapped by OSM. Furthermore, we exploit the use of additional geo-location data sources to help localize the objects.

3 Data Generation Pipeline

We developed a modular framework for collecting satellite imagery of a specified object class using public geographical databases and crowd-sourced labels. The different modules of our framework are shown in Fig. 1 and explained below.

3.1 Satellite Imagery

In this work, we employ the European Space Agency’s Sentinel-2 (A and B) [5] constellation of twin satellites. It is a wide-swath, high-resolution, multi-spectral imaging mission supporting Copernicus Land Monitoring studies (monitoring vegetation, soil, and water cover, as well as observation of inland waterways and coastal areas). The Sentinel-2 Multispectral Instrument samples thirteen spectral bands in the visible, near-infrared, and short-wave infrared part of the spectrum with a spatial resolution of 10 meters (m) per pixel for the four optical and near-infrared (NIR) bands, 20 m for the six red edge and shortwave infrared (SWIR) bands, and 60 m for the three atmospheric correction bands.

Sentinel Hub’s [17] Open Geospatial Consortium (OGC) API can be used to provide services for data access, display, and processing within hours of image acquisition. The `sentinelhub` Python package facilitates OGC web requests to download and process satellite imagery in batch scripts thus circumventing the need to manually download Sentinel-2 data from the web. Sentinel Hub’s Web Map Service (WMS) is used to request downloads of certain Sentinel bands

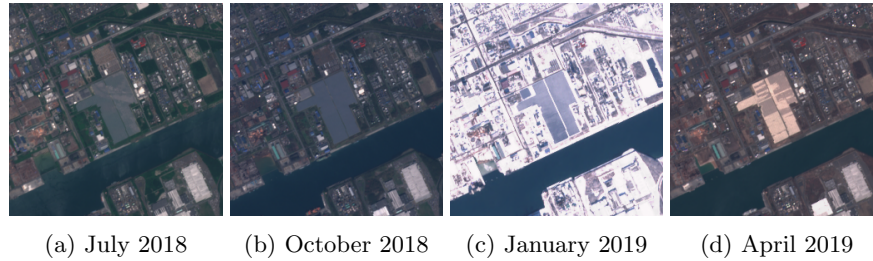


Fig. 2: Sentinel-2 imagery of the same area acquired at four different times.

using different settings such as maximum cloud coverage, time range of imagery acquisition, image format, size, etc. The region of interest to be downloaded is represented as a bounding box in geo-coordinates. This enables the extraction of satellite imagery for any particular region of interest at any required time period, thus avoiding time mismatches between imagery and any ground truth data. Figure 2 shows some examples of Sentinel-2 RGB imagery acquired at four different times highlighting the variations in the image across different climatic conditions (notice the snow cover in Fig. 2(c)).

3.2 Object Localization

An object in satellite imagery can be a building, forest, industrial area, etc. selected to train a classifier. The geo-coordinates of many types of objects can be found using publicly available resources, such as a catalog of street addresses or a database containing geo-coordinates. Given a list of addresses, open-source Geocoders such as Nominatim [13] can be used to determine their geo-coordinates. Python’s `geopy` package supports several popular geocoding web services. Also, some public geo-databases exist that already provide a geo-coordinate for a category of interest (e.g., power plant database [3]). However, such data sources may have only an approximate (inaccurate) location for the object. We therefore will need to search for the object near the given geo-coordinate. For this, we use a global spatial database which contains object footprints to identify bounding regions.

3.3 OpenStreetMap

OpenStreetMap (OSM) [11] is a collaborative project used to create a crowd-sourced spatial database of the world. It was started in 2004 with the simple idea of multiple contributors with local knowledge collaborating to create a detailed labeled map. We employ OSM as our auxiliary data source to extract polygonal footprints for the given geo-location of objects obtained from public databases.

The OSM data model consists of three basic data structures: *Nodes*, *Ways*, and *Relations*. A *Node* represents a geographic point expressed in latitude and

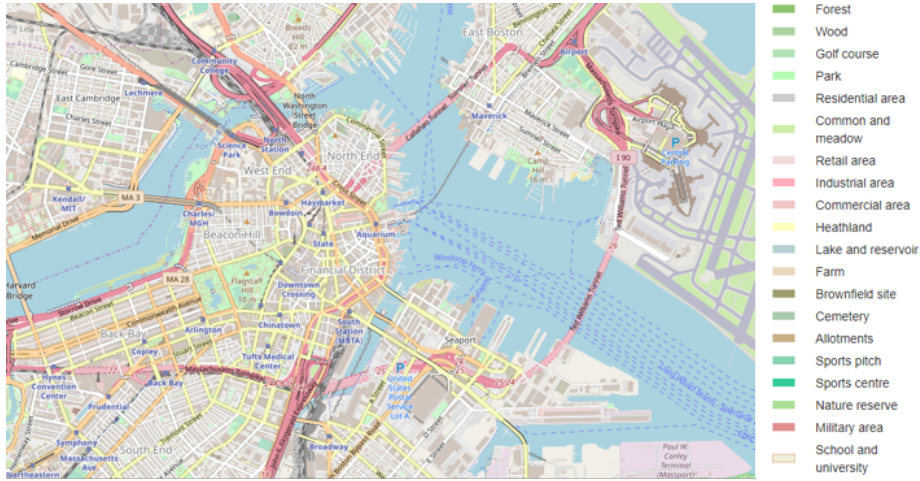


Fig. 3: OSM region with land use labels.

longitude. A *Way* constitutes at least two *Nodes* (polyline or polygon). A *Relation* is a logical collection of *Ways* and *Nodes* (multipolygon). The physical features on the Earth’s surface are described by OSM using tags with key-value pairs attached to its basic data structures (*Nodes*, *Ways*, and *Relations*). The key is used to describe a topic, category, or feature type (e.g., building), and the value details the specific form of the key (e.g., residential). The OSM Map Features pages on the OSM wiki [12] lists the tags agreed upon by the OSM community. Figure 3 shows an example OSM urban area with land use labels.

There are multiple methods for downloading data from OSM. It is possible to get the data in the form of XML formatted .osm files. OSM also provides its entire database as a Planet.osm file which is updated weekly to reflect new changes to its database. But often we do not need to work with the entire database when there is a specific region of interest. In such cases, the Overpass API [14] can be used, which is a read-only, web-based service that accepts queries to download custom filtered datasets. Python’s *overpy* package provides Python bindings to the Overpass API.

We use the object’s initial geo-coordinate to restrict our search space in OSM to avoid the overhead of having to search throughout the entire planet to localize our object footprints. We next define a search space around the object’s geo-coordinate. We build a query in the Overpass Query Language which contains the search criteria and a bounding box defining the search window. We use a pre-determined size for the search window based on the average size of our objects. We then search within the window using the Overpass API for the polygonal footprint of the object class.

We search for both *Relations* and *Ways*, as some objects might be represented as *Relations* (multipolygon) which are made up of multiple *Ways* (polygons). In such cases, we extract all *Ways* which are a member of a *Relation*. The *Ways*

are made up of an ordered list of *Nodes* which form the vertices of the object’s polygonal footprint. A bounding box is fit to the entire footprint using the min and max latitude and longitude extents. Sentinel Hub is then used to download the satellite imagery corresponding to the footprint bounding box generated for the object.

Using the various packages related to Sentinel and OSM, a batch script can be used to extract imagery of a selected object type from the associated list of geo-coordinate locations. We can use this same approach to generate negative (non-object) examples by querying OSM using the Overpass API to ensure the absence of an object.

4 Example and Experiments

We demonstrate the proposed framework by procedurally generating a training and testing dataset with positive and negative examples of solar power plants. To show the applicability of the extracted dataset, we train an image classifier using a Convolutional Neural Network (CNN) to detect the selected object class. The main focus of this work is the automatic generation of annotated imagery (other object classes and classification models could be used).

4.1 Solar Power Plants

We leverage the World Resources Institutes Global Power Plant Database (GPPD) [3], which is a comprehensive, open-source database of power plants around the world. Each power plant listed has information on its geo-location, capacity, generation, ownership, and fuel type. It is continuously updated as new information becomes available. The database version used in this work includes over 28K power plants. However as previously mentioned, some of the geo-locations are only approximate and may not even exist on the footprint of the power plant.

We center our search region in OSM on the provided geo-coordinate of a solar power plant from the database. We define our search window by ± 0.2 decimal degrees in latitude and longitude around the geo-coordinate (approximately a 16×16 km area centered around that geo-coordinate). We then search through OSM using the Overpass API in the search window for *Relations* and *Ways* with the tags ‘**generator:source = solar**’ or ‘**plant:source = solar**’ (standard tags for solar power plants in OSM, note that other key-value tags for different objects can be used). We extract all *Ways* which are a member of *Relations*. The *Ways* give us the desired polygonal footprints of the solar power plants. Figure 4(a) shows multiple polygons that make up an OSM relation representing a solar power plant.

4.2 Image Collection

The next step is to extract the satellite imagery for the located object footprints. We selected to evaluate the RGB and NIR bands as they have the highest spatial resolution (10 m) among the bands in the Sentinel-2 imagery. Also, the solar

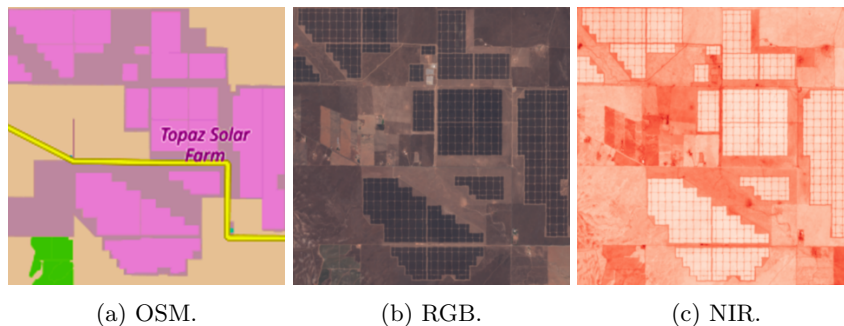


Fig. 4: OSM polygonal footprint and its corresponding Sentinel-2 RGB and NIR band imagery.

power plants were visually more distinct in the RGB and NIR bands to the human eye (NIR is closer to the visible range than thermal). Other bands may be more applicable to different object classes. Figure 4(b)&(c) show the corresponding Sentinel-2 RGB and near-infrared band imagery for the OSM polygonal footprint of a solar power plant (see Fig. 4(a)). Table 1 lists the central wavelength, bandwidth, and spatial resolution for each of the Sentinel-2 bands used.

Table 1: Spectral band properties for Sentinel-2 (A and B).

Sentinel-2 Bands	Sentinel-2A		Sentinel-2B		Spatial resolution (m)
	Central wavelength (nm)	Bandwidth (nm)	Central wavelength (nm)	Bandwidth (nm)	
B02: Blue	492.4	66	492.1	66	10
B03: Green	559.8	36	559.0	36	10
B04: Red	664.6	31	664.9	31	10
B08: NIR	832.8	106	832.9	106	10

Positive Examples. To extract multiple positive examples of each solar power plant, we randomly sampled 5 points within each polygonal footprint and extracted a 256x256 sized image chip centered around each of these points (see Fig. 5). To download the Sentinel-2 imagery, we used Sentinel Hub’s WMS request. We downloaded imagery for July 2018, October 2018, January 2019, and April 2019 to account for variations in the appearance across different climatic conditions (as seen in Fig. 2). We additionally set the maximum cloud cover percentage to 30% to filter out imagery where the solar power plant may be occluded by clouds. Since the cloud coverage is estimated on larger Sentinel-2 tiles (100 km x

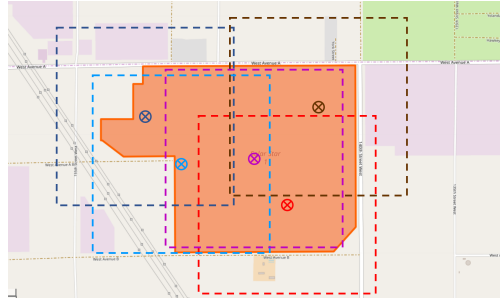


Fig. 5: Randomly sampled points and their cropping windows.

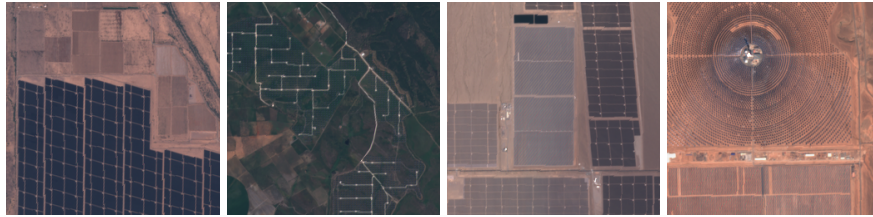


Fig. 6: Positive examples of solar power plants.

100 km areas), and not just for the region defined by a footprint bounding box, heavily clouded imagery might still be present in the smaller image chips. Also, a part of the image might contain white (empty) regions if that particular data acquisition only partially intersected the specified footprint’s bounding box. We manually removed such corrupt images from our collection.

We generated approximately 20K positive samples from 400 solar power plants as our training data and another 500 positive samples from 100 different solar power plants for testing purposes. We created a validation set by randomly sampling 2K images from the training set. Figure 6 shows a few positive samples of solar power plants generated by this method.

Negative Examples. The negative samples consist of land regions with no solar power plants, but contain a diverse collection of urban as well as randomly sampled land areas. We sampled from two separate world cities databases of urban areas across the world. We used the freely available World Cities Database [16] (provides an accurate and up-to-date database of the world’s cities and towns) for training data and Nordpil’s World Database of Large Urban Areas [10] which contains a different set of geo-coordinates of urban areas for generating test data. Both databases provide a single geo-coordinate for geo-referencing cities and towns. The remaining negative samples are located by randomly generating a geo-coordinate on the Earth’s surface and verifying with OSM that the randomly generated point is over land (not over water).

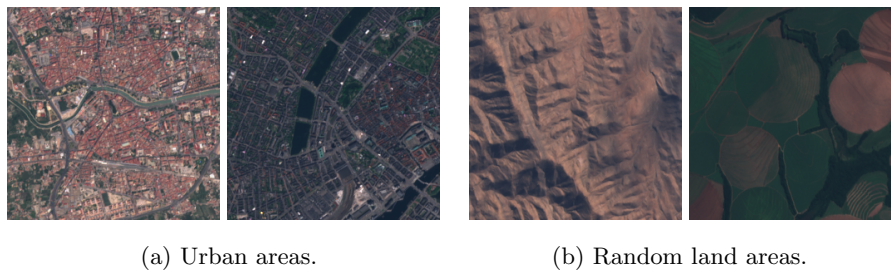


Fig. 7: Negative examples.

We extracted 256x256 sized image chips centered on the geo-coordinates of the selected urban and random land areas. We again use the Overpass API to query the OSM database to ensure that the bounding boxes contain no solar power plants. Several heavily clouded image chips remained in our negative samples. We collected approximately 20K negative samples for training and another 500 negative samples (250 urban areas, 250 randomly sampled land areas) for testing. As before, we randomly sampled 2K images from the negative training set with an equal proportion of images from urban areas and randomly sampled land areas for the validation set. Some negative examples from urban and random land areas are shown in Fig. 7.

4.3 Image Classification Model Architecture

We employed a standard CNN architecture to train our image classification model. The network employs three downsampling blocks, each composed of a convolutional layer (Conv) with 32 filters of size 3x3 and ReLU activation, followed by a 2x2 max pooling layer (MaxPool). The output from the final downsampling block is flattened and fed to 2 fully-connected layers (FC) having 512 nodes each with ReLU activation. The output layer consists of a single node with a sigmoid activation function. Separate models were trained for NIR (1 channel), RGB (3 channels), and RGBNIR (4 channels). The input to the network is the image chip of size 256x256 and the number of input channels is equal to the number of bands being used to train the model. The model architecture is shown in Fig. 8 and takes an input of size 256x256 with either 1, 3, or 4 channels.

Each image was preprocessed by normalizing the pixel values to the range [-1,1]. The model was trained using stochastic gradient descent with a batch size of 32 images and momentum of 0.9. We initialized the weights in each layer using Xavier initialization and the biases were initialized with values drawn from a normal distribution with zero mean and unit standard deviation. The learning rate was initialized to 0.001 and adjusted using “Poly” learning rate adaptation [4,19]. We trained each model for 30 epochs and selected the model from the epoch which had the highest validation accuracy. In addition, we also created a

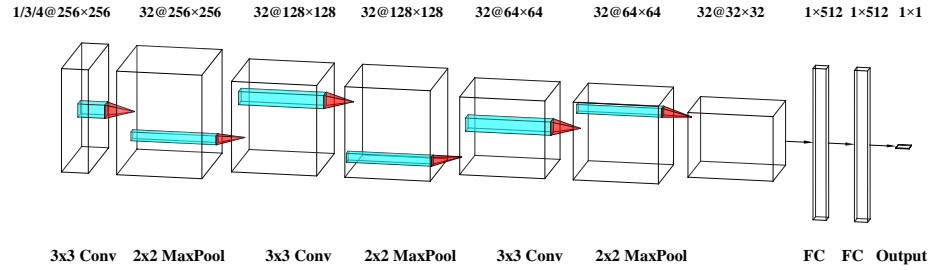


Fig. 8: CNN Model architecture.

Table 2: Comparison of results from models trained on different bands.

Bands	Accuracy	Precision	Recall	F1 score	Parameters
NIR	0.92	0.98	0.86	0.91	17,059,905
RGB	0.96	0.98	0.93	0.95	17,060,481
RGB+NIR	0.96	0.98	0.94	0.96	34,120,386
RGBNIR	0.97	0.99	0.94	0.97	17,060,769

model that gives predictions obtained by averaging the sigmoid outputs of the individual RGB and NIR models (RGB+NIR).

4.4 Results

The results of the various models on the solar power plants are summarized in Table 2. The best performance was given by the model trained on all the four bands together (RGBNIR), with an accuracy of 0.97 and an F1 score of 0.97. As expected, the joint RGBNIR model with early fusion (at the input) incorporating inter-dependency was better than the individual models. It was also slightly better than the independent late fusion approach (RGB+NIR). The RGBNIR model had an increase of 288 parameters over the RGB model and 864 parameters over the NIR model, but it had nearly half of the parameters of the dual late-fusion approach (RGB+NIR). This shows that the model's performance improves as we increase the number of bands used for training, advocating the use of potentially even more bands provided by Sentinel.

4.5 Alternate Object Classes

The above framework can be used to collect satellite imagery of other object classes by swapping the public data source used to extract the point location of the required object class and then using their corresponding OSM tags to search for their polygonal footprints. Figure 9 shows imagery generated using the framework for airports, reservoirs, and ports.

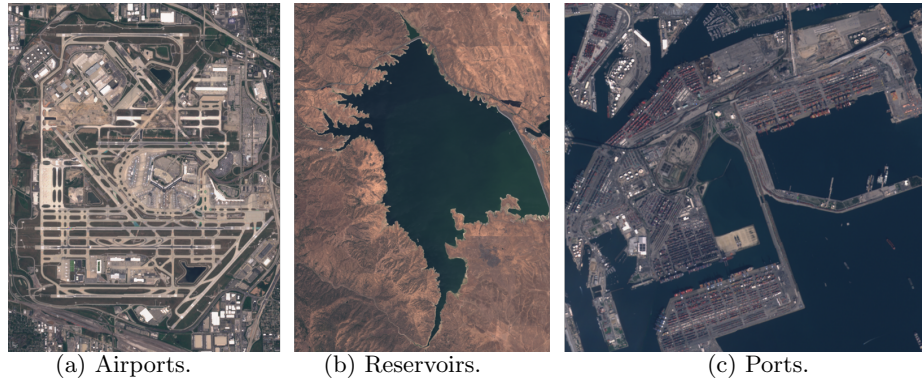


Fig.9: Example imagery collected for alternate object classes.

5 Summary

Manually generating and annotating imagery is a time-consuming task and a commonly faced problem. This task becomes more complicated with the use of supervised deep learning models which require large volumes of labeled training data. In this work, we presented a general framework employing existing geographical databases and satellite imagery to automatically generate labeled imagery of a selected object class for a classification task. We demonstrated this method by selecting solar power plants as our object class, though other classes could also be used with the framework. We achieved compelling results with multiple image bands which validate the proposed pipeline. We next plan on leveraging the framework to examine the use of the polygonal footprints to label each pixel in the image to facilitate semantic image segmentation. We expect the use of deep learning on satellite imagery to increase as more avenues are opening for automated annotation.

References

1. Albert, A., Kaur, J., Gonzalez, M.C.: Using convolutional networks and satellite imagery to identify patterns in urban environments at a large scale. In: Proceedings ACM SIGKDD (2017)
2. Audebert, N., Le Saux, B., Lefèvre, S.: Joint Learning from Earth Observation and OpenStreetMap Data to Get Faster Better Semantic Maps. In: Proceedings CVPR Workshop: Large Scale Computer Vision for Remote Sensing Imagery (2017)
3. Byers, L., Friedrich, J., Hennig, R., Kressig, A., Li, X., McCormick, C., Valeri, L.M.: A global database of power plants. World Resources Institute **18** (2018)
4. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE Trans. on Pattern Analysis and Machine Intelligence **40**(4) (2017)

5. European Space Agency: Copernicus Sentinel-2 data [2019], <https://sentinel.esa.int/web/sentinel/missions/sentinel-2>
6. Ishii, T., Simo-Serra, E., Iizuka, S., Mochizuki, Y., Sugimoto, A., Ishikawa, H., Nakamura, R.: Detection by classification of buildings in multispectral satellite imagery. In: ICPR (2016)
7. Johnson, B.A., Iizuka, K.: Integrating OpenStreetMap crowdsourced data and Landsat time-series imagery for rapid land use/land cover (LULC) mapping: Case study of the Laguna de Bay area of the Philippines. *Applied Geography* **67** (2016)
8. Kaiser, P., Wegner, J.D., Lucchi, A., Jaggi, M., Hofmann, T., Schindler, K.: Learning aerial image segmentation from online maps. *IEEE Trans. on Geoscience and Remote Sensing* **55**(11) (2017)
9. Kussul, N., Lavreniuk, M., Skakun, S., Shelestov, A.: Deep learning classification of land cover and crop types using remote sensing data. *IEEE Geoscience and Remote Sensing Letters* **14**(5) (2017)
10. Nordpil: World database of large urban areas, 1950-2050 (2019), <https://nordpil.com/resources/world-database-of-large-cities/>
11. OpenStreetMap contributors: Planet dump retrieved from <https://planet.osm.org> (2017), <https://www.openstreetmap.org>
12. OpenStreetMap Wiki: Map Features (2019), https://wiki.openstreetmap.org/w/index.php?title=Map_Features&oldid=1819914
13. OpenStreetMap Wiki: Nominatim (2019), <https://wiki.openstreetmap.org/w/index.php?title=Nominatim&oldid=1848597>
14. OpenStreetMap Wiki: Overpass API (2019), https://wiki.openstreetmap.org/w/index.php?title=Overpass_API&oldid=1872170
15. Schultz, M., Voss, J., Auer, M., Carter, S., Zipf, A.: Open land cover from OpenStreetMap and remote sensing. *International Journal of Applied Earth Observation and Geoinformation* **63** (2017)
16. SimpleMaps- Pareto Software, LLC: World Cities Database (2019), <https://simplemaps.com/data/world-cities>
17. Sinergise Ltd.: Modified Copernicus Sentinel data [2019]/Sentinel Hub, <https://sentinel-hub.com/>
18. Sui, D., Goodchild, M., Elwood, S.: Volunteered geographic information, the ex-afford, and the growing digital divide. In: *Crowdsourcing Geographic Knowledge*. Springer (2013)
19. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: *Proceedings of CVPR* (2017)
20. Zhao, W., Bo, Y., Chen, J., Tiede, D., Thomas, B., Emery, W.J.: Exploring semantic elements for urban scene recognition: Deep integration of high-resolution imagery and OpenStreetMap (OSM). *ISPRS Journal of Photogrammetry and Remote Sensing* **151** (2019)