Augmenting One World Terrain (OWT) Data with Civilian Infrastructure by Using Machine Learning Techniques

Matthew Rolfe, Dartangan Jackson

Dignitas Technologies

Orlando, FL

mrolfe@dignitastechnologies.com,
djackson@dignitastechnologies.com

Marjaneh Safaei, Freddie Santiago

Dignitas Technologies

Orlando, FL

msafaei@dignitastechnologies.com
fsantiago@dignitastechnologies.com

ABSTRACT

Modern simulation environments generally do not contain high fidelity Civilian Infrastructure (CI) features, or they lack the necessary content (e.g., data model and attribution) to simulate the second and third order effects caused by a critical component outage. This problem can often be traced back to the Geographic Information Systems (GIS) data used to create the simulation environments. Although GIS data repositories, like OpenStreetMap or National Geospatial-Intelligence Agency's (NGA) Geospatial Repository and Data (GRiD), contain some level of CI feature content, this content is often incomplete and does not contain enough metadata to properly model the relationships and dependencies between CI features. Therefore, a new approach is needed for the simulation and training community that leverages and enhances the existing CI content.

In this paper we describe an automated method for collecting CI features from various image sources, including satellite, street view, and drones that leveraging machine learning (ML) modules to detect and georegister multiple CI features simultaneously. We define a framework for orchestrating the execution of ML modules within Docker containers, which establishes a set of services for CI feature collection. This framework will be flexible to allow integration of new feature detection modules, adjustable to execute ML services in a variety of permutations, and scalable to handle very large areas of coverage by utilizing a distributed processing environment for feature detection. Automating the process of CI feature collection will result in significant savings in time and money when compared to manual collection efforts. By enhancing the CI features available in geospatial databases, training scenarios in urban environments can consider the second and third order effects caused by outages.

ABOUT THE AUTHORS

Matt Rolfe is a Senior Software Engineer at Dignitas Technologies. He has over 16 years of experience with a wide variety of software applications, including simulation Interoperability on SE Core A&I and distributed network exercises for OneSAF, CCTT, and AVCATT, using DIS and HLA, all involving M&S under Army Contracts.

Dartangan Jackson has been working as a software engineer for 6 years focusing mainly on M&S. He has worked on a wide range of projects such as ARL's Adaptive Learning and Intelligent Tutoring for the Marksmanship Training using GIFT, and the Augmented Reality Sand Table ARES. He is currently working as a part of the Megacities and Underground Simulation Environments (MUSE) research effort.

Marjaneh Safaei has a strong research background with a PhD in computer science. She is highly skilled in machine learning/deep learning, computer vision and image processing methodologies as well as related tools and technologies. She has published over 10 scientific conference papers in the top tier artificial intelligence and machine learning conferences. She is currently working as a machine learning engineer/researcher at Dignitas Technologies.

Freddie Santiago has over 15 years of applied experience related to M&S applications with a specific focus on terrain databases, synthetic environment services, dynamic terrain, and database validation and correlation testing mechanisms for programs, such as SE Core A&I/CVE, OneSAF, CDT, CCTT, AVCATT, and OWT. He is presently the lead of the MUSE research effort to represent critical infrastructure and underground environments within Megacities for military simulations.

Augmenting OneWorld Terrain (OWT) Data with Civilian Infrastructure by Using Machine Learning Techniques

Matt Rolfe, Dartangan Jackson

Dignitas Technologies

Orlando, FL

mrolfe@dignitastechnologies.com,
djackson@dignitastechnologies.com

Marjaneh Safaei, Freddie Santiago

Dignitas Technologies

Orlando, FL

msafaei@dignitastechnologies.com,
fsantiago@dignitastechnologies.com

INTRODUCTION

The Army's Synthetic Training Environment (STE) and One World Terrain (OWT) aim to provide a next-generation, whole-world environment for simulation and training. Simulations will no longer be limited to relatively small, disjoint areas of interest. Rather, the Army's goal is to provide up-to-date geographical information data to trainers and soldiers at the point of need (PoN). Additionally, the Army seeks to provide enhanced training for potential military engagements in environments that are difficult to train live; for example, dense urban environments (i.e., megacities). The 3-dimensional (3D) complexity of operational environments in megacities, including high-rises, underground environments, and limited maneuverability space, presents challenges for Army training (see Figure 1). Operations in megacity environments must also consider the side effects of damaging or destroying Civilian Infrastructure (CI), whether intentional or not. CI includes infrastructure components for energy, water, communications, transportation, critical manufacturing, and financial services. Disabling CI components can have significant humanitarian or operational repercussions. Therefore, it is important that simulation and training scenarios support environments with enough fidelity to properly model CI along with 2nd and 3rd order effects caused by the failure of a CI component. Consider, for example, the implications to the execution of a military operation if an electric power plant goes offline in a training simulation; whether through deliberate or accidental means (e.g., collateral damage).

Figure 1: Example of a Megacity and its complex environment

Modern simulation environments generally do not contain high fidelity CI features. This problem can often be traced back to the Geographic Information Systems (GIS) data used to create the simulation environments. Although GIS data repositories, like OpenStreetMap (OSM) or National Geospatial-Intelligence Agency's (NGA's) Geospatial Repository and Data (GRiD), contain some level of CI feature content, this content is often incomplete and does not contain enough metadata to properly model the relationships and dependencies between CI features. Historically, simulation systems have also lacked sufficient hardware to support complex simulations. Limitations are often related to disk storage capacity, system memory, processing power, and 3D rendering capabilities.

The evolution of GIS data and collection methodologies has, up until recently, outpaced the ability for simulation systems to keep up with the GIS content being produced. Although simulation systems attempt to make use of the GIS data, the content is often used at a low-level of detail to meet the needs and limitations of specific systems. However, the recent, rapid evolution of the gaming industry has caused ripple effects through many industries, including modeling and simulation (M&S). Off-the-shelf gaming hardware can now outperform many of the simulation systems in use by the Department of Defense (DoD). Therefore, hardware is no longer the bottleneck when it comes to realistic training simulations. The M&S industry now finds itself in a situation where the hardware has leapfrogged the fidelity of existing GIS data and can efficiently simulate very complex environments, but the sufficiently detailed GIS data is not available. Therefore, a new approach is needed for the M&S community that leverages and enhances the existing GIS content, including CI data. In this paper we propose Machine Learning (ML) methodologies to identify and map CI features to help create and complete the CI data models to achieve higher levels of fidelity. Using available satellite, street-view, and drone imagery, we leverage ML models to identify objects of interest and georegister these objects so they can easily be included in simulations.

CURRENT STATE OF CIVILIAN INFRASTRUCTURE DATA

The shortfalls in modeling CI and the effects of CI failures in simulation can often be attributed to a lack of data, simulation model fidelity, and capable hardware. For example, the hardware for many legacy simulation systems cannot handle all the extra data necessary to properly simulate CI. To begin our research, we analyzed a variety of data repositories to determine the level of support for CI features. In this section, we provide information about some of the commonly used repositories for Army simulation training and their level of support for CI.

OSM is a collaborative project to create a free editable map of the world[1]. This collaborative/crowdsourced environment allows anyone to update the repository with new content. Although OSM does have some high-level CI data, such as power plants, substations, water treatment, and telecom information, it is generally tailored to the needs of a single user or project focusing on that area and data point. Additionally, because the content is crowdsourced, it is not necessarily up to date, potentially contains inaccurate information, and may not have every item that critical infrastructure models need.

According to NGA's website, GRiD was developed "to serve as the National System for Geospatial-Intelligence's (NSG's) enterprise level database for geospatial data. GRiD is designed to store, process, visualize, and disseminate a variety of geospatial datasets" (2021). The data content within GRiD is better curated than that of OSM since it is meant to support intelligence agencies. However, this means that content within GRiD evolves much slower and some of the data may contain restrictive classifications. Although GRiD does contain high quality imagery and other GIS content, it does not generally define CI features.

"The NSG Open Mapping Enclave (NOME) is an open-source collaborative mapping environment designed to allow NGA and its partners to work together simultaneously to generate dynamic content in support of missions throughout the world" (Statement for the Record before the Senate Select Committee on Intelligence 2016). Similar to the other geospatial data sets we analyzed, NOME has the same issue with sourcing data. Data is sourced via users and can take quite a bit of time.

A common theme across the analyzed data repositories is that the availability of CI data varies from city to city and country to country with no overarching standard. In some areas, data is provided as open source, while in others it is nonexistent. Even when CI content exists in these repositories, it does not contain enough metadata to properly model the relationships and dependencies between CI features.

GENERATING CIVILIAN INFRASTRUCTURE DATA

Our research into existing CI data content found that although crowdsourced data does contain some level of CI features, the content is not complete enough to support the generation of synthetic environments for simulations. Therefore, a new process must be developed to augment existing data and create new content as needed. One solution to this problem is to establish guidelines for crowdsourced collections of CI features so that all content meets a minimum set of requirements. This, however, is difficult to enforce within data repositories which are not consistently curated, such as OSM. Even in cases where repositories are better curated (e.g., NOME) the generation of CI data

[1] For more information on Open Street Map visit https://www.openstreetmap.org/copyright

content is too slow to meet the ever-increasing demand of the DoD M&S systems. Thus, we propose a system which leverages machine and deep learning methodologies to detect and annotate CI features using commonly available data as input. Once properly trained, such a process will be capable of exhaustively analyzing regions of the world to create accurate CI data. This process will also be capable of evolving to handle feature types beyond CI.

Our initial experiments for automated CI feature generation focus on detecting features related to the power grid (e.g., power stations, substations, pylons, and powerlines). Our approach sought to create an ML module capable of analyzing satellite imagery to detect desired features. This required providing the underling machine learning neural network with enough annotated training data for it to understand what it is expected to detect. In preparation for training, our team collected roughly 30 satellite images and hand-annotated them to create a bounding box around the relevant features. We then executed a supervised neural network training process to iterate on the images in order to allow the neural network to adjust its internal weights (i.e., guesses) until it reached an acceptable detection rate. This resulted in a proof of concept for our automated detection process, which provides a viable solution to automated CI feature detection (specific details on this process will be discussed in other sections within this paper). Although the initial proof of concept proved viable, the creation of a robust process capable of accurately detecting CI features in a variety of world regions requires a significantly larger training dataset, often needing thousands of images. This is time consuming and potentially expensive to create manually. Thus, we experimented with an automated process for creating training data by leveraging existing, albeit limited, CI data from OSM as a guide in locating CI content within satellite images, with the end goal of training the ML modules enough to go above and beyond any content found within OSM (see Figure 2).

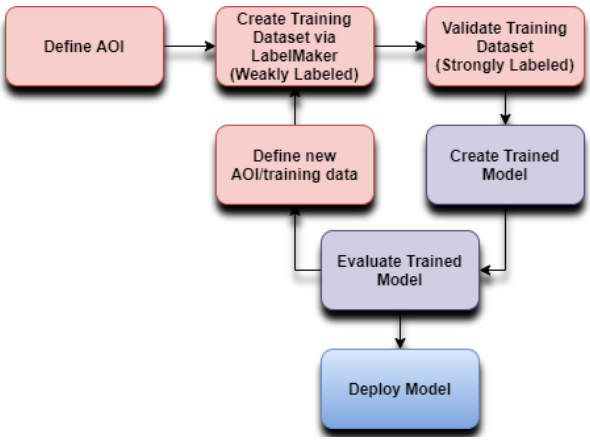

Figure 2: High level Model approach

We use Label-Maker, an opensource project, as an automated application to generate the datasets for the training and validation of models. In our process, Label-Maker allows the user to specify an area of interest along with objects of interest that the model is trained to detect. It uses OSM data to determine specific locations containing relevant CI data in order to pull satellite imagery containing those objects of interest. The satellite imagery can be obtained from any Application Programming Interface (API) that supports XYZ Tiles, also known as Slippy Map Tilenames[2]. Mapbox, a mapping and location cloud platform for developers, provides satellite images, but other providers can be used as well (e.g., Google Maps, Bing Maps). Label-Maker then creates label files, containing metadata about the desired objects from OSM, as well as annotated images containing the bounding boxes of the desired objects. The generated imagery follows the Slippy Map Tilenames naming convention by default, which contains information about the image's tile location and zoom level and allows for the calculation of the real-world GPS coordinates of the image, as well as the pixel precision. Training data is considered *weakly* labeled until it is validated by human since it could potentially contain errors. A human will validate the data by verifying the accuracy of the annotations (e.g., correct bounding box sizes), determining the usability of the image (e.g., too blurry or not enough detail), and confirming the validity of the labels (e.g., incorrectly labeled a parking lot as a substation). This validation process is less time consuming than a human creating the training data from scratch since it minimizes the need to manually search for relevant images and annotate them.

Validation is performed using a modified version of a common open-source tool named Labelme. Labelme is a popular general use annotation tool within the ML community because of its ease of use and vast number of built-in features. Labelme can create annotations for segmentation and bounding boxes for object detection. We use this tool to verify the validity of the images pulled from Label-Maker. Users can modify annotations, add new annotations, or remove images entirely from the dataset if they are invalid. The resulting training data is considered *strongly* labeled since it

[2] For more information on Slippy Map tilenames visit https://wiki.openstreetmap.org/wiki/Slippy_map_tilenames

has been verified by a human. Validated data is then used to finalize the dataset and export, in this case, TensorFlow tf-records that are used for the training and testing of models with TensorFlow.

The TensorFlow 2 object detection framework is used to train the models, using MobileNet and EfficientNet as its underlying neural network backbone. Although TensorFlow can be used to create a model from scratch, it is often more efficient to start with an existing pre-trained model and provide additional training to suit the specific need. A variety of pre-trained models are available to act as a backbone to the new models, where each pre-trained model has different strengths and weaknesses, such as speed vs. accuracy or the type of objects they are better at detecting. The use of pretrained models as a foundation is also beneficial since it requires significantly less training data to get started due to the fact that they are already trained to detect the primitive shapes (e.g., lines, edges, circles), which are then used to derive and detect more complex features. In our experiments, MobileNet and EfficientNet models were pretrained using the Coco17 dataset[3], then additional training was performed to fine-tune them until they produced satisfactory results for our use case. The trained models are then tested using new imagery that it has not seen before. If the model performs as expected, it is then deployed in the CI feature detection system.

Finally, deployed models in a CI feature detection system analyze new satellite images to detect relevant features. The model reviews each image it receives and draws a bounding box around each detected object/feature. If the object is detected and the model has a certain level of confidence in its detection (e.g., > 60% confidence), the imagery is automatically passed to a georeferencing algorithm to determine the real-world location of the object. The georeferencing algorithm uses the images with the Slippy Tilenames format, as previously mentioned, to calculate the real-world location of the objects by first deriving the upper left bound of the image (i.e., image origin) using information provided by the image's file name. This, combined with the zoom level and uniform image size, allows the system to calculate a value for the distance per pixel. The real-world latitude and longitude of the object is then calculated by finding the pixel offset between the image's origin and the object's bounding box center, then the distance per pixel is used to derive the final coordinate values.

Evolution of the CI Feature Detection and Data Generation Process

The process of extracting CI data evolved as the team gathered lessons learned and the desired detectable objects varied. Over the course of our research, we experimented with detection and extraction of features like sewers and manholes, fire hydrants, and power grid networks. All of these features posed unique challenges requiring different approaches to overcome. The information contained in this section describes potential pitfalls and lessons learned as our experimentation evolved and eventually culminated in the process described in the previous section.

Our research into ML techniques for CI feature extraction stems from the need to create a process for the procedural generation of underground synthetic environments. One goal for procedurally generating underground environments is to create relatively accurate representations of sewer networks. Existing GIS data (e.g., vector data) can be used when available, but not much usable data was available. Therefore, we began experimenting with image processing and ML techniques to detect some of the components that make up a sewer network. In this case our targets were locations of manholes since those are critical points from which a large portion of the network can be derived.

The first attempt was analyzing satellite imagery using ML to detect the location of manholes. Unfortunately, the available satellite imagery did not contain enough resolution for manhole identification, even for a human. So, we shifted our focus to street level imagery for a closer point of view to the desired features. We considered two threads with this approach. With the first, we considered full motion video captured from the dashcam of a Tesla vehicle. The second approach considered street level imagery from Google and Bing maps. Using the Tesla video, we created a proof of concept to detect manholes in images using video frames to train an ML model with Google's DarkFlow system, and then process other frames to test the manhole features detected. Although the detection accuracy was not great, given the minimal training data set, it did provide a level of confidence that the process was viable and could potentially be applied to other full motion video processing, such as drone videos. This research thread, however, was deactivated since it was not easily scalable to other world regions. We shifted to using street level images to reproduce the manhole detection process executed on the Tesla video. This time we collected a handful of images (~50) from Google Street View (SV) and annotated them to train a model using DarkFlow. Results were once again not very

[3] The coco dataset is freely available at https://cocodataset.org/

accurate considering the small set of training images, but they were still promising. This experiment also brought to light the problem that proper training of an ML feature detection module requires several training images for it to be accurate, which requires significant time or manpower.

Our research to this point demonstrated that manholes can be detected from street level imagery, so we expanded our work to test the flexibility of our approach by considering new features. We began by testing how quickly we could train a new model to detect fire hydrants in a city environment, specifically Boston, MA. We also began experimenting with a process to geolocate the features once they are detected. Google SV was once again used to gather images for fire hydrants, and then they were annotated for training. In order to expedite the collection of street level images, we created a script that used a shapefile of fire hydrant locations, provided by the city of Boston, to automatically download Google SV images at those location and at different orientations. The resulting 320 images were then sifted by hand to collect the best ones for annotation and training. The trained model achieves ~80% positive detection rate using the testing image set. The final trained model was then tested in a new area of the city by executing a script to create sample points along the roads using GIS vectors as guides, download images from Google SV at the sample locations, pass the images to the ML model for detection, and finally geolocate the objects. Any objects detected and geolocated were automatically validated against the Boston fire hydrant dataset to verify proper placement. The results are shown in Table 1. The table shows that the model was able to detect and properly geolocate ~31% of the hydrants within 0.75 meters of the real-world locations, which is a reasonable number considering that the team collected images, annotated them, and trained the model in less than a week. However, the large number of false positives (i.e., detected hydrants where there are none) was concerning since it required a human to filter the invalid hydrant detections before the accuracy of the geolocation process could be gauged. In Table 1, we can see 133 images that were marked as containing hydrants. Out of those, 103 were false positives. Additionally, out of the 27 valid hydrant identifications, 10 of them were duplicates; meaning that the same hydrant was reported from multiple images. This resulted in 17 correctly identified and geolocated hydrants.

Total Fire Hydrants	55
Identified Hydrants	133
Valid Identifications	27
Duplicates	10
Unique Valid Locations	17
False Positives	103
Unidentified Hydrants	38
Ratio Valid To Total	30.9%

Table 1: Fire Hydrant Results

Our low-level success in geolocation of detected fire hydrants is attributed to a few issues. First, the set of training images for the model is created using regular Google Street View images. However, the images used for testing are panorama images from Google, which are often warped based on camera lens (see Figure 4). This confused the model and resulted in a large number of false positive results or missed hydrants. Second, panoramas were used because they are typically paired with a depth field which can be used to project the pixels in an image to a real-world location. However, in testing we found that the origin of the panoramas was often offset by a few meters from the location reported by Google and the original location used for the image request. This caused the geolocation process to report incorrect locations. There were also problems with multiple objects being reported for a single hydrant if multiple images contained the object, as well as some objects not being reported at all because they were always blocked from view. Lastly, we found that geolocating objects using a minimal set of street level images was inherently inaccurate unless some information is available about the dimensions of the object being detected. An image's pixels are projected out using the associated depth map, but those pixels can only go as far as the front face of the detected object (see Figure 3). This means if the object is very deep, the error in geolocation can become significant. Although fire hydrants are small enough to minimize the error, it was still significant enough to throw off our automated validation process for geolocated objects.

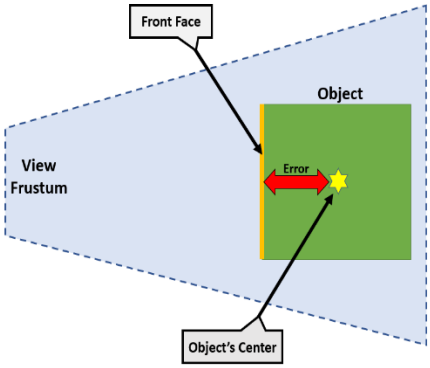

Figure 3: Object Geolocation Error

Figure 4: Google Panorama

From here, our experiments shifted to target near-term use cases for CI in One World Terrain (OWT) datasets, with power grids being the lowest common denominator across most CI use cases. Thus, we reverted to using satellite imagery for detecting power grid components since they are typically large enough to be easily discernible in satellite images. Using lessons learned from previous experiments, we devised a process to automate the collection and annotation of training images. Specific information on the process of detecting power grid components from satellite images are covered in the next section.

Power Grid Component Detection

We continued the use of semantic segmentation for initial uses with satellite imagery-based detection. A model was trained using strongly labeled satellite imagery of power plants as the dataset. It contained Coal, Nuclear, Hydro, and Wind powered power plants[4]. Challenges arose with the use of semantic segmentation during the geolocation process because the segments were often oddly shaped, not always one solid object, or had pixel artifacts that were unrelated to the actual object (see Figure 5). It proved difficult to determine which portion of the segmented object to geolocate. When geolocating all segments, a vast amount of irrelevant data is generated along with a vast number of false positives. The process also requires human judgement to filter out artifacts and handle the complex cases of partial segmentation. Because of these complications, we decided to sideline the segmentation approach and use bounding boxes for feature detection in our experiment.

Figure 5: (Left) Semantic segmentation problem. (Right) Bounding box detection example

Bounding boxes allow easier geolocation with satellite imagery, as the center of the bounding box is easy to calculate. There is no noise in the image that requires a human to make judgement calls for false positives. Bounding box detection also provides a confidence level to the prediction which supports fewer potential false positives (see Figure 5). However, semantic segmentation is ultimately a better approach for feature extraction since it provides a higher level of accuracy when outlining complex features[5].

Our process uses openly available information for real-world object locations, such as power plants and substations, and gathers satellite imagery directly from these locations. This eliminates most of the randomness experienced with the previous Street View method since it is no longer capturing images hoping one of the images contains the object. Label-Maker is an open-source product that provides the functionality to both gather satellite images of specific requested objects as well as use Open Street Map information to automatically annotate those objects. These annotations still need to be manually validated by a human but since most of the images contain relevant information, the process is significantly faster.

Data preparation consists of two main phases, data validation and conversion to the proper training format. Any data obtained requires human verification for accuracy and to eliminate any data that could prove detrimental to the training set. The human in the loop uses a modified version of the Labelme tool to accomplish this task. It is flexible enough to allow the user to manually sift through and annotate the vast amount of data received from the automated image collection process, as well as modify the pre-existing annotated data generated by Label-Maker. The user is required

[4] Dataset created by Duke University freely available at https://doi.org/10.6084/m9.figshare.5307364.v1
[5] V. Lempitsky, P. Kohli, C. Rother and T. Sharp, discuses the benefits of using boing boxes prior to segmentation

to modify any annotations that are not accurate, remove any images that do not contain the objects, and add missing annotations.

Once data has been validated by the user, it must be converted to the proper format for the model training framework. For this test case the data is converted to tf-records to be compatible with TensorFlow 2. This process is straightforward and standard with most TensorFlow models. A ground truth class file containing the different object classes (e.g., power substations) to be detected, along with the imagery itself, is passed into a conversion script that automatically generates the records. The dataset is then broken up into a training set and an evaluation set during this process. This is standard practice for creating ML models as it sets aside a new set of images the model has never encountered to evaluate the accuracy of the model. Once the process is completed, the object detection model is ready to begin training.

The first step in the training process is to adjust model parameters such as learning rate and batch size. A pre-trained model is used as a base for the new model to significantly decrease the required training time. These pre-trained models are openly provided by TensorFlow along with statistics on their performance. SSD_EfficientNet and SSD_Mobilenet, both pre-trained off the COCO-17 dataset, are used as the backbone for our models. The model begins training and is checked periodically to monitor its status. The model's loss rate and other statistical information is collected automatically during training and used during the evaluation process. Once training is complete the model is then evaluated to determine if it meets the desired performance requirements.

Experiments	Network Architecture	Dataset	Pre-trained Weights	Batch Size	Total Steps	Base Learning Rate	Warmup Learning Rate	Warmup Step	Precision (AP) (all area) IOU = 0.5-0.95	Precision (AP) (all area) IOU = 0.75	Precision (AP) (large area) IOU = 0.5-0.95	Recall (large area) IOU = 0.5-0.95
Baseline	SSD_EfficientNet	Substation (WL)	Coco-17	8	60,000	N/A	N/A	N/A	47%	52%	52%	67%
Model_1	SSD_EfficientNet	Substation (WL)	Coco-17	20	7,500	0.0133	0.001	500	53%	65%	60%	71%
Model_2	SSD_Mobilenet	Substation (WL)	Coco-17	16	6,000	0.025	0.005	100	56%	64%	64%	75%
Model_3	SSD_EfficientNet	Substation (SSL)	Coco-17	20	7,500	0.0133	0.001	500	66%	75%	68%	80%
Model_4	SSD_Mobilenet	Substation (SSL)	Coco-17	16	6,000	0.025	0.005	100	76%	90%	77%	85%

Table 2: Machine Learning Model Training and Evaluation Metrics

During the evaluation process, the evaluation dataset (generated during the conversion steps) is passed to the model to test how the model performs on new data it has never encountered before. If the model does not perform up to standards, the information gathered during the training process is reviewed and adjustments are made where necessary. This might involve gathering more training data, further validation of the dataset, or adjusting parameters within the configuration file. If the model performs well on this data, it is accepted and deployed to be one of the models within the object detection modules. Table 2 shows some of the results from multiple configurations to improve the accuracy of the model. In Experiment_7 the model was able to achieve 85% valid detection over the evaluation dataset.

Deployed models support geo-referencing the objects that they detect. The models are fed new data with each image containing metadata about where the image was taken and the zoom level of the image. The results are passed to the geo-referencing algorithm which begins calculating the real-world locations of the detected objects.

EXPERIMENTS

Experiments were run using four of the models presented on Table 2. These models were trained using substation images from the central Florida area. The images were gathered by passing the extents of the desired area or interest, along with a list of desired features, to LabelMaker in order to identify locations with relevant substation data at zoom level 18. LabelMaker then downloaded and automatically annotated 742 training images (i.e., WL - Weakly Labeled), which were eventually trimmed down to 496 images after human validation (i.e., SSL - Semi-Strongly Labeled). A testing set of images was created using the same process for the Augusta, Georgia area at the same zoom level. This resulted in a testing set with 53 WL images containing 55 substations, and 34 SSL images remained after human validation. The specific models used from Table 2 were Experiment_1, Experiment _4, Experiment _5, and Experiment _7. These models represent different combinations of the EfficientNet and MobileNet neural networks with WL and SSL training data.

Each trained model was used to process the Augusta test dataset. Any images which resulted in a substation detection confidence level of 30% or above were passed to geolocation creation process. Geolocations for this experiment were single points derived from the center of the substation's bounding box. The resulting bounding boxes and geolocations

from each model were then validated. Detected bounding boxes were compared with the Augusta WL bounding boxes to determine the level of agreement, specifically the Intersection Over Union (IOU) and mean IOU(mIOU). The geolocations were validated visually to determine whether the points were within the bounds of the feature in the real world.

RESULTS

Total Substations	34

EN = EfficientNet MN = MobileNet	Substation Predictions	Valid Identifications	Unique Valid Locations	False Positives	Unidentified Subsations	Ratio Valid to Total	SL Confidence to mIOU R-squared Correlation
Model 1 (EN + WL)	27	21	19	6	15	55.88%	24.30%
Model 2 (EN + SSL)	28	24	23	4	11	67.65%	30.50%
Model 3 (MN + WL)	40	27	25	13	9	73.53%	39.20%
Model 4 (MN + SSL)	38	27	25	11	9	73.53%	60%

Table 3: Substation detection results from Augusta dataset

Model Result Notes

- Substation Predictions are the total number of identified Substation with an IOU greater than 0.5.
- Valid Identification is a prediction that correctly identifies the substation object.
- Unique Valid Locations represents the number of valid locations accounting for the duplicates.
- False Positives are the total number of identified substations with an IOU less than 0.5.
- Unidentified Substations represents the number of stations that were not identified. In other words, the Total Substations minus the Unique Valid Locations.
- Ratio Valid to Total equals the Unique Valid Locations divided by the Total Substations.
- The linear regression R-squared value shows us any correlation between the model's confidence and IOU of its identification. While this value does not indicate accuracy or precision of the model it does help us quickly identify areas to investigate where a high confidence may be associated with a low IOU result. This could mean, for example, that the model had a high level of confidence that it detected a substation, but the resulting bounding box did not match very well with the bounding boxes of expected substations.

Results Conclusion

Both models were able to correctly identify and locate over 50% of the actual CI features. There is evidence that training the models on semi-strongly labeled datasets decreased the number of false positives while maintaining or even improving valid identifications. MobileNet performed extremely well with the valid identifications where as EfficientNet showed its strength with a low number of false positives and duplicates. Based on the results from the experiment it has been determined that it is possible to accurately detect and geolocate CI features from satellite imagery using machine learning methods.

LESSONS LEARNED

Our initial goal was using automation to save time in enhancing geospatial datasets with high fidelity CI data. We discovered that creating and refining data collection and ML detection processes still require time investment during the initial phases of the process. The automated gathering of training data saves time but only creates a weakly labeled dataset which still needs to be validated by a human. Once a dataset is human validated the training set is considered semi-strongly labeled and used to create a trained model for testing. We considered the training data to be semi-strongly labeled instead of strongly labeled because we also identified subcategories of substations that could be split to create a more accurate model. For this test run we considered all substations to be a single category. Each step of the process requires specialized tools that help create a final training set that produces viable results with a high-level of confidence.

Satellite imagery tiling and zoom levels created a few issues that require some pre- and post-processing. Using a standard Slippy tile format of 256 x 256 images, each object requires a specific zoom level for optimal viewing. Power poles are detected and identified best at zoom level 19, 30 cm/pixel, while larger power stations are best viewed at zoom level 15, 4.75 m/pixel. Even with identified ideal zoom levels for each type of object, there is no guarantee that an object exists in a single tile. Bounding boxes identifying an object can overlap tiles and leave corners and small sections that indicate the area is a substation but do not contain enough area to provide relevant imagery. Leaving these objects in the training material causes the model to perform poorly since feeding it nonrelevant information returns poor results. Once a model is deployed in a testing environment this produces a miss since we do not expect a model to identify small sections of an object independently of the whole object. Identifying these areas and either re-tiling the object by combining and cutting existing tiles or implementing a Level of Detail approach to use a higher zoom will help rectify issues and help provide cleaner training material and output data.

When training ML models we used a single object of Substations for simplicity. While this makes our experiment more focused, it isn't necessarily the ideal for training object detection models. Images with multiple objects annotated for training can help create a more accurate model. It also makes application easier as a single model can identify more than one object.

The lead time for creating an acceptable trained model is greater than anticipated. Our experience with the process of changing Hyper Parameters, cleaning up training data and interpreting results was a much larger task than expected.

ROADMAP FORWARD

Based on the knowledge gained through our research, we can apply the processes described within this paper in a variety of ways to enhance CI content within OWT datasets as well as provide benefit to the M&S community. In this section we will briefly describe future threads that will be investigated to carry the research forward.

Detect and publish new CI content

Considering that many existing data repositories do not contain enough GIS data related to CI, any new content created through our proposed semi-automated process could be published to open repositories. The auto-generated data would need to be curated to ensure accuracy. However, once the process matures and achieves a high level of confidence it may only require spot checks, rather than exhaustive validation, before being published.

Multi-repository Sampling and Merging

Early in our research we found that existing data repositories would sometimes contain erroneous data since much of it was crowdsourced and not very tightly curated. This, however, doesn't necessarily mean that the data is erroneous to the point that it is completely unusable. Therefore, we propose a process that scours a variety of data repositories searching for CI content and determines a level of confidence for the content. Such a level of confidence can be derived by how much agreement there is in the placement of features across repositories. Our proposed ML process could also be used as an additional boost to the level of confidence. For example, all repositories agree that there is a power plant at a specific location, but their outlines differ slightly. The ML process could be triggered at the given location to see which outline fits best. Any data with a high level of confidence could be merged and published to a repository (e.g., NOME).

Docker Services for CI Feature Detection and Extraction

Crowdsourced data collection, although problematic at times, is a great way to generate content quickly. If CI content detection and extraction tools were available to the community as services, the amount of available CI content could increase significantly. These tools could be exposed to the community through existing frameworks, such as the STE World Server (STEWS), or other host systems. The CI services could also be frequently updated to use the "best of breed" machine learning models for feature detection and extraction. Additionally, services for specific models could be combined in a variety of ways to detect complex CI features. For example, a complex model could be composed to not only detect the existence of an airport, but also make use of other models to detect critical components within the airport (e.g., control tower, terminal, security gates).

3D Content Analysis and Augmentation

As complex CI content in military simulations becomes more prevalent, there is bound to be a lag between the desire to support specific CI features in a training scenario and the available content within the simulation terrains. Because of this, we propose the creation of a tool capable of analyzing 3D terrains to detect CI features. If CI features are detected, the tool will analyze the corresponding data structures to verify that the necessary attribution exists to support CI simulation. This could include proper feature labeling, verification of critical attribution content and ranges, and validation of dependencies (e.g., a building must be connected to a power source). Additionally, the tool could be used to process 3D terrains and embed CI attribution if it does not exist.

CONCLUSIONS

Our results show the feasibility of semi-automated processes to geolocate CI objects using freely available satellite imagery and other GIS data. Currently, this process requires human participation to validate and fill in the gaps. For example, identifying power towers and substations in a city area cannot be 100% accurate but a human will easily see missed areas and fill in the gaps with a quick check of the imagery.

Beyond geolocation, objects such as power plants can be further divided into subcategories, including solar, hydroelectric, or nuclear and more accurate models can be created and trained using these categories.

One area that needs further research is completing CI feature data and metadata. Can we train a ML model to identify power production capability of a plant from imagery? Can we estimate a substations throughput ability or a high yield tower via markers in the images? Estimated coverage areas for commercial, industrial, and residential areas could be inferred, and those types of attributes could be used in simulation if it contained CI content and a data model for it.

REFERENCES

Bradbury, Kyle; Brigman, Benjamin; Chandrasekar, Gouttham; Collins, Leslie; Hossain, Shamikh; Jeuland, Marc; et al. (2017): Power Plant Satellite Imagery Dataset. figshare. Dataset. https://doi.org/10.6084/m9.figshare.5307364.v1

Cardillo, Robert. "Statement for the Record before the U.S. Senate Select Committee on Intelligence." *Statement for the Record before the U.S. Senate Select Committee on Intelligence | National Geospatial-Intelligence Agency*, National Geospatial-Intelligence Agency, 28 Sept. 2016, www.nga.mil/news/Statement_for_the_Record_before_the_U.S._Senate_Se.html#:~:text=The%20NSG%20Open%20Mapping%20Enclave,of%20missions%20throughout%20the%20world.

Home. GRiD. (n.d.). https://grid.nga.mil/grid/.

Lempitsky, V., Kohli, P., Rother, C., & Sharp, T. (2009). Image segmentation with a bounding box prior. *2009 IEEE 12th International Conference on Computer Vision*. https://doi.org/10.1109/iccv.2009.5459262

Statement for the Record before the Senate Select Committee on Intelligence. (2016). https://www.nga.mil/assets/files/20160927_DNGA_SSCI_SFR_(1).pdf