Abstract

We use a Mask-RCNN architecture to perform instance segmentation on satellite images of regions in Western Europe, with the goal of determining the footprint of logistics real-estate in said region. In this paper we explain the roadmap to the development of our current model, which is trained first on images with logistics real-estate labeled by the Google Maps API, and then on images with labels added on the basis of high confidence predictions from the first round of training. This model achieves recall of 60% and precision of 76% on our evaluation set, in contrast with Google labels' recall of 30% and precision of 89%, hence providing a superior tool for indicating supply in the region.

Introduction

We propose to develop a deep learning application which detects logistics supply using satellite images to get a better indication of supply in certain areas of Europe. Specifically, we want to get an estimate of the area of logistics real estate given satellite imagery. Hence, the task at hand is the following: building detection using satellite imagery, followed by a building type classification (logistics vs non-logistics). The input to the algorithm is a satellite image. We then use a neural network to output the area of logistics properties.

Related Work

Building detection from satellite imagery has been a challenging deep learning task that has attracted significant research efforts over the last decades[1]. One of the most cited scientific articles related to building detection using satellite imagery is the following: *"Building detection in very high resolution multispectral data with deep learning features"* by Vakalopoulou, Maria, et al. (2015). In this paper, Vakalopoulou et al. propose a novel building detection strategy that builds upon the pre-trained AlexNet network. More specifically, they fed the network with satellite images of "buildings" and "non-buildings". The problem was thus formulated as a binary classification problem. To detect building footprints, features of a specific layer of the AlexNet were subsequently used to train a SVM model. Meanwhile, to increase the accuracy of the building boundary predictions, they embedded spectral information to the training of the model. This strategy was also intended to optimize the detection of smaller buildings.

Previous CS230 projects focusing on building detection using satellite imagery include *Syrov & Ulanov (2019)* and *Bhowmik, Hall, Quinn (2018).* Both projects approached the task of building classification using the Mask-RCNN model. *Syrov & Ulanov* additionally tackled the problem with Conditional GAN, where the discriminator task was to recognize if the input was a real building or artificially generated. The Generator network's task was to fool the discriminator and also come close to the ground truth image of the building footprint.

While these previous approaches serve as a useful starting point, they do not fully comprise the scope of our task. That is, because our task cares about not only detecting building footprint, but also detecting a specific type of building. Hence, we expect there is room to innovate with respect to previous work.

Dataset and Features

The raw dataset consists of coordinates (longitude & latitude) relating to the location of real estate property accompanied with the property type. The dataset is proprietary and is sourced from an institutional real estate investor. In total we have roughly 100k coordinates, of which 10% relates to the class of interest (i.e. logistics). Using the Google Maps API, we obtain satellite images (RGB) corresponding to the coordinates with a size of 640x640. Given the skew to negative cases in data, we had to take care in constructing a balanced dataset to train on.

Table 1: A summary of the building types present in the entire raw dataset

Property Type	Count	% of Total
Office	27,841	27.3%
Residential	35,803	35.2%
Retail	28,233	27.7%
Logistics	9,976	9.8%
Total	101,853	100%

To label the data with the positive masks (i.e. presence of logistics properties), we took the following steps:

1. As the coordinates in the raw dataset may not exactly correlate with position of the building, we performed a Google Maps search within the region of interest using the following 'logistics' keywords: *logistics service* and *wholesaler* (as wholesalers often

[1] Vakalopoulou, Maria, et al. "Building detection in very high resolution multispectral data with deep learning features". *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. IEEE, 2015.

occupy the logistics properties). The Google Maps API returned all centroid coordinates of buildings which appear during the Google search and which we assume to be logistics properties.

2. Using the *Overpass API* we obtained all building edges in the region of interest and correlated these with the centroids obtained from the Google search to retrieve our positive masks.

From the entire raw dataset, we took a subset of the coordinates as converting these to images and retrieving the Google masks was costly beyond a certain point (Google charges the use of its API's after a certain threshold).

Google has not labeled all logistics properties. As such, a caveat to this approach of labeling is that the Google Maps API did not return all logistics properties. This resulted in labels having some noise (false negatives), which had its implications for training and development. Error analysis in an earlier stage of the project showed that this caveat in labeling was significant. In a lot of cases: for an actual logistics property, the algorithm indicated the existence of logistics – but the Google label was absent – lowering the precision where the algorithm actually did a good job. To overcome this bias in the evaluation metric, we labeled images ourselves for validation purposes. As the human labeling process is very time consuming, the ground truth labels for the (initial) training still originate from Google.

Another adjustment we made to the dataset is to add images of urban and rural areas containing no logistics buildings. Initially, we only took images with logistics objects present, based on the assumption that images had enough background (e.g. farmland or other buildings) for the model to distinguish between logistics buildings vs. background and non-logistics buildings. Error analysis, however, showed the model still had some difficulties making this distinction. For example, the model mistakenly classified clumps of trucks as buildings and in some instances mis-interpreted a plain field for a logistics building. Therefore, to increase the performance of our classification model, we added images of urban and rural locations containing no logistics at all.

Table 2 below gives an overview of the final dataset used for training and validation. Note, using the human labels for validation instead of the Google labels makes a significant difference – where Google identifies 197 objects in the validation set, a human would identify 860 objects.

Table 2: Overview of the final dataset used for training and validation

Set	# orginal images	% of total	augm. multiplier	# objects [google]	# objects [human]
Training [logistics]	5349	93%	x2	9861	-
Training [non-logistics]	296	5%	x3	-	-
Validation [logistics]	126	2%	x1	197	860
Total	5771	100%	-	10.058	860

The relatively small size of the validation set is due to the fact the human labelling was very time consuming. To get a representative validation set, we used all human labeled images in the validation set. Future work can focus on creating a sizable test set as well to get a more unbiased estimate of model error. Furthermore, we used data augmentation to increase the size of our training set. The amount of augmentations is indicated by the "augm. multiplier", where *x2 = a horizontal flip*; *x3 = a horizontal & vertical flip*. Pixel values were also rescaled from *0-255* to *0-1* to improve the performance of gradient descent.

Methods

For our baseline model, we implemented **Mask R-CNN**[2] using an open-source tutorial[3]. We fine-tuned the pre-trained Mask R-CNN model and used *torchvision*[4] to train our own dataset. Mask R-CNN, which builds upon Faster R-CNN[5], is a deep neural network that is typically used in computer vision and machine learning to solve image segmentation problems[6]. The model outputs object bounding boxes, classes and masks per image input. As shown in Figure 1, there are two main stages in this neural network:

Stage 1: Generate bounding box proposals for object location within a given image (input).
Stage 2: Predict the object class, refine the bounding box, and generate a mask segmentation mask for each object instance.

[2] https://arxiv.org/pdf/1703.06870.pdf
[3] https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html
[4] https://pytorch.org/docs/stable/torchvision/index.html
[5] https://arxiv.org/pdf/1506.01497.pdf
[6] https://medium.com/@alittlepain833/simple-understanding-of-mask-rcnn-134b5b330e95

The Mask R-CNN architecture we implemented is as follows:

- A backbone CNN for classification (ResNet50) with a Feature Pyramid Network (FPN)[7], *pre-trained on COCO*
- An Instance Segmentation model to identify object masks, *pre-trained on COCO*
- A Region Proposal Network (RPN) anchor generator
- A Region of Interest (ROI) Classifier to identify possible locations of objects

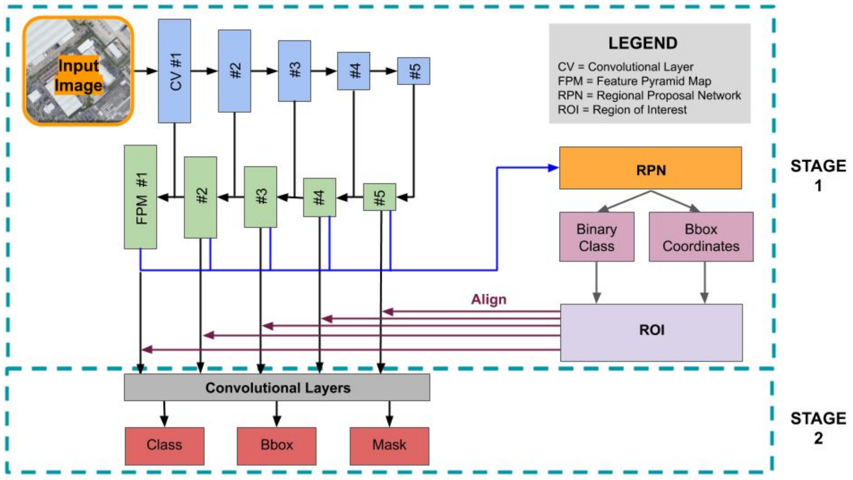

Figure 1: Illustration of the Mask R-CNN architecture in our implementation (adaptation[8])

As shown in Figure 1, the model receives a satellite image as input. Then, a bottom-up computation of convolutional layers is used for feature extraction. The features extracted from the last stage in each layer are then fed into a top-down pathway using lateral connections to generate a pyramid of feature maps, which in turn, extract ROI features. Said features are of different scales and come from different levels of the pyramid, increasing the model's speed and accuracy[9]. In the next stage, the RPN uses the extracted features and anchors to generate proposals for objects of interest. This includes an estimate of the object's bounding box. Then, a ROI Align classifier performs intelligent re-shaping to make the images be of the same size. Finally, the proposed regions are fed into a new set of convolutional layers to generate the object classes, bounding boxes and masks.

In our implementation, we experimented with single-class classification as well as multi-task classification by tweaking the characteristics of the model's classifier (please see the appendix for our data-labeling process flow). For single-task classification our classifier focused on *logistics vs background*, while for multi-task classification, it focused on *logistics vs non-logistics vs background*. The latter did not perform well, which is why we focused on single-task classification.

<u>The Loss Function</u>

The loss function on each sampled region of interest is equal to: $L = L_{cls} + L_{box} + L_{mask}$

Where the classification loss L_{cls} is a logarithmic loss between two binary classes and the bounding box loss L_{box} is a logistic regression over the bounding box coordinates and L_{mask} (normalized between 0 and 1) is the average binary cross entropy loss between the ground truth mask of an object and the predicted segmentation mask. It should be noted that L_{cls} and L_{box} are taken from the Fast RCNN[10] model.

[7] https://arxiv.org/pdf/1612.03144.pdf
[8] https://medium.com/@alittlepain833/simple-understanding-of-mask-rcnn-134b5b330e95
[9] https://arxiv.org/pdf/1703.06870.pdf
[10] https://arxiv.org/pdf/1504.08083.pdf

Experiments/Results/Discussion

Evaluation Metrics:
Considering that we care about getting an estimate of the area of logistics real estate in a certain region: we have specified our evaluation metric as the F1-score regarding the logistics area. More specifically:

- Recall = $\frac{Logistics\ Area\ (Labeled) \cap Logistics\ Area\ (Predicted)}{Logistics\ Area\ (Labeled)}$
- Precision = $\frac{Logistics\ Area\ (Labeled) \cap Logistics\ Area\ (Predicted)}{Logistics\ Area\ (Predicted)}$
- F1 Score = $2 \cdot \frac{Recall \cdot Precision}{Recall + Precision}$

Where the *Logistics Area (Labeled*) relates to the human labeled masks and the *Logistics Area (Predicted)* to the masks predicted by the model. Hereby, we only take into account predictions where the classification probability exceeds the confidence threshold of 0.5. Moreover note that, by focussing on the logistics area, we penalize the model more for missing larger objects and less for missing smaller objects. Lastly, to get a better feel for the validation set – in the 126 images roughly 7 million pixels were labeled as logistics, translating to a prevalence of 14% with respect to all pixels in the set.

Hyperparameters:
- *Optimizer:* we used stochastic gradient descent with momentum and weight decay. Stochastic gradient descent was chosen as our computational resources did not allow to process multiple examples at once.
- *Learning rate:* started at *0.005*, shrinks with factor 10 every 3 epochs. We shrink the learning rate to converge using Stochastic gradient descent.

Results and Discussion: We investigated the performance of *four* models in our implementation. We started with a *baseline model* and each subsequent model aimed at improving issues identified during error analysis of its predecessor. In all our models, we stopped the training when we observed that the validation performance started decreasing. In our error analysis, we used results from the previous checkpoint (epoch). The results of our three models are summarized in Table 3 below:

Table 3: Results from Mask R-CNN model iterations to predict Logistics Buildings vs Background

Model Name	Model Description	Recall	Precision	F1 Score
0	Existing Solution (Google labels)	30%	89%	45%
1	Pretrained Mask RCNN (no fine-tuning)	86%	16%	27%
2	Baseline model (finetuned Mask RCNN)	52%	79%	63%
3	Relabeled input (minimize incorrect labeling)	60%	72%	65%
4	Relabeled input and more non-logistics buildings	60%	76%	67%

As shown in Table 3, the *Existing Solution* (Google Labels) has a low recall percentage (30%), but high precision (89%). The low recall value stems from the limitation that the Google Maps API has not identified all logistics properties, leading to false negatives in the labelling process. We started by training *Model 1* and *Model 2* with this limited set of image labels provided by the Google API to leverage their high precision.

In *Model 1* we ran experiments using the pretrained COCO weights. As expected, the model performed poorly with a F1 score of 27%, as it is trained to detect other objects (e.g. cars, people, animals). The high recall, yet poor precision, indicates the model is detecting a lot of logistics slightly better than random (prevalence of logistics area is 14%, where the model has a precision is 16%). To improve the model's precision, we finetuned the model on logistics images.

Model 2 reflects those changes, with an increased precision of 79% and detecting more than 50% of the labeled logistics area. Despite the limitation of having false negatives in our training set, error analysis and the fact that the recall is better than that of the Google Labels show that *Model 2* can identify more logistics buildings than the Google API (as shown in Figure 2). The model can also accurately distinguish between logistics and non-logistics buildings (e.g. residential). However, the model still has difficulties with contrasting rectangular shapes – for example, in figure 2 it can be seen how *Model 2* mis-interpreted tennis courts, a soccer field and a swimming pool for logistics.

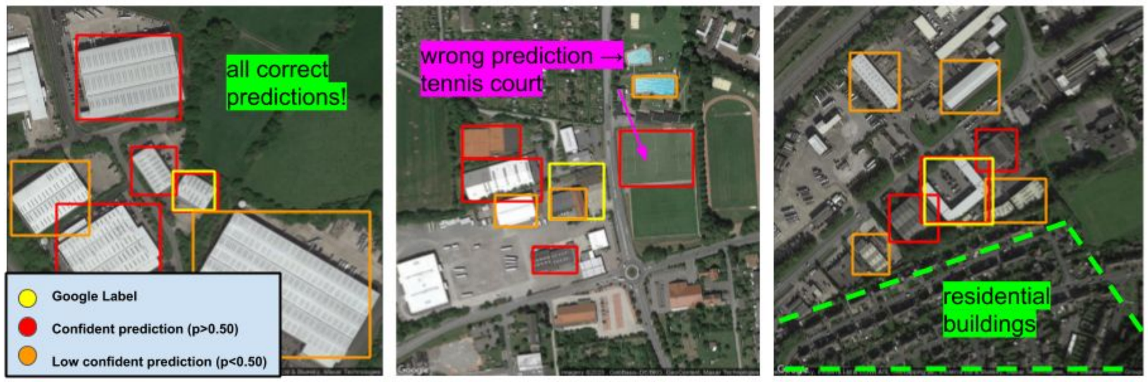
all correct predictions!
Google Label
Confident prediction (p>0.50)
Low confident prediction (p<0.50)
wrong prediction → tennis court
residential buildings

Figure 2: Examples of Model 2 output; identifying more logistics than Google (left); misclassifying large contrasting rectangles (middle); distinguishing between residential and logistics (right). Note that magenta and green annotations were added for illustration purposes.

In *Model 3*, we improved our model's performance even further and reduced the number of false negatives. To achieve this, we fed back into the training set a number of *re-labeled* cases on which *Model 2* gave high probability of being "logistics" – with a confidence threshold of 50%. As in the validation set Google's recall was 30% and the baseline model had a recall of 52%, it is reasonable to assume that relabeling the training set in this way, the number of *false negatives* would be reduced. And as expected, the recall of *Model 3* increased with 8% to 60%. However as the precision of the baseline model was lower compared to the Google Labels, relabeling the training data introduced other incorrect labels – being *false positives*. Hence, we see the precision of *Model 3* decreasing due to false positive labels. Overall, looking at the F1 score, the *Model 3* has a better performance compared to its predecessors

In *Model 4*, our goal was to minimize classification errors (e.g. tennis court example) by training the model on more urban and rural images with no logistics present. The model performance increased as expected – w.r.t. *Model 3*, this model has the same recall yet improved its precision with 4%..

Conclusion/Future Work

While our model's performance is already an improvement over Google's labeling of logistics buildings, the following are steps we would wish to take to further improve it.

- Hyperparameter tuning – in particular, we wish to experiment with the confidence threshold we use to relabel our data to decrease Google's incorrect non-labeling of logistics buildings, without introducing many mistakes by mis-labeling non-logistics buildings
- Continue with several more iterations of the cycle of feeding in confident predictions as labels into our model, until we reach approximate convergence
- Increasing batch size for stochastic gradient descent to ensure that training converges
- Experimenting with UNET[11], an image segmentation architecture which outputs predictions for each pixel in the image
- Training on more data, and experimenting with different data distributions. In particular, does our model systematically make better predictions for certain areas of Western Europe? And, in our eventual real-world use case, what type of test set distribution would we expect to see? Answers to these questions will inform us of biases in our current training set and how to correct for them.

Contributions

Kevin developed the idea and motivation for the problem, obtained and labeled data, and also took the lead on most aspects of our model development given that he has expert knowledge of our real-world task goals. Eleni contributed greatly to articulating all aspects of our work in the milestone write ups and final report, as well as model evaluation. Anastasiya worked on image pre-processing, error analysis and model development. The whole team held frequent group check-ins and troubleshooting sessions, and everyone was generally able to contribute to the span of tasks involved in the project.

References

- Vakalopoulou, Maria, et al. "Building detection in very high resolution multispectral data with deep learning features". *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. IEEE, 2015.
- He, Kaiming, et al. "Mask r-cnn." *Proceedings of the IEEE international conference on computer vision*. 2017.
- Syrov, Ulanov, "Building footprint extraction based on RGBD satellite imagery". *CS230 project,* 2019
- Bhowmik, Hall, Quinn "Building Detection in Satellite Images: Improving Resource Distribution in Rohingya Muslim Refugee Camps". *CS230 project,* 2018
- Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". 2016.
- Girshik R. "Fast-RCNN". 2015.
- Zhang, Xiang. "Simple Understanding of Mask RCNN". 2018. https://medium.com/@alittlepain833/simple-understanding-of-mask-rcnn-134b5b330e95

Appendix

In terms of defining the dataset, we modified our data to fulfill the requirements of the model. To that end, we fed into the model two sets of images:

1. Unlabeled Data: Satellite images containing various mixes of logistics buildings, non-logistics buildings, and background (PNG format);
2. Labeled Data: Satellite images with segmentation masks (logistics buildings vs background). Different colors represent different object instances.

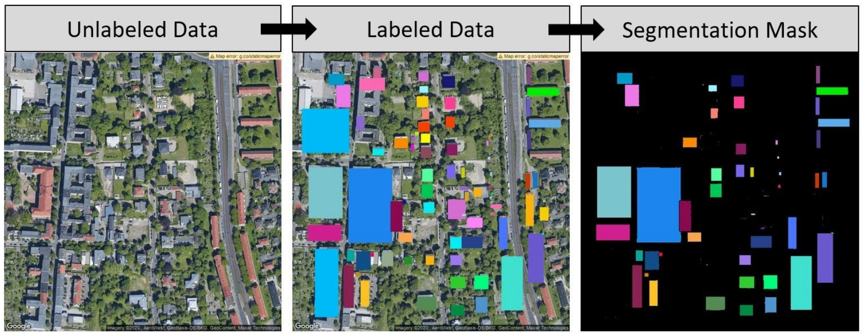

Figure: Labeling data and creating the segmentation masks; This is an example of non-logistics buildings.