Introduction to Deep Learning Final-term Report

Buildings Detection in Remote Sensing Images using Faster R-CNN with and without Denoising: A Comparative Study

GROUP 6

Members

Trương Quốc Cường	BI11-049
Nguyễn Tường Quang Hải	BI11-073
Lê Trần Khương Duy	BI11-069
Nguyễn Danh Hoàng	BI11-094
Nguyễn Xuân Nguyên	BI11-204
Phan Minh Quang	BI11-232

Lecturer

Dr. Nghiêm Thị Phương

1. Introduction

Buildings detection in remote sensing images is a challenging task with numerous applications in fields such as urban planning, disaster management, and environmental monitoring. The ability to accurately identify buildings from satellite or aerial images can provide valuable insights into the built environment, population density, and infrastructure of a particular area.

Deep learning methods have shown promising results in object detection tasks, including buildings detection in remote sensing images. In particular, Faster R-CNN is a widely used deep learning architecture for object detection, which consists of two stages: a region proposal network (RPN) and a detection network. [1]

However, remote sensing images are often affected by various types of noise, including speckle noise, which can significantly affect the accuracy of building detection. Denoising techniques can be applied to improve the quality of the images and enhance the performance of building detection algorithms. [2]

In this study, we aim to compare the performance of Faster R-CNN for building detection in remote sensing images with and without denoising. We will evaluate the effectiveness of denoising techniques in improving the accuracy of building detection, as well as the trade-offs between performance and computational complexity. Our findings will provide valuable insights into the best practices for building detection in remote sensing images and can inform the development of more accurate and efficient algorithms for this task. [3]

2. Dataset Description

The SpaceNet1 dataset is a publicly available dataset created for the purpose of advancing computer vision research in the field of geospatial image analysis. The dataset consists of high-resolution satellite images of Rio de Janeiro, Brazil and contains annotations for building footprints. The dataset was created as a collaboration between CosmiQ Works, DigitalGlobe, NVIDIA, and Amazon Web Services. [4]

The images in the SpaceNet1 dataset were collected by the WorldView-2 satellite with a resolution of 50 cm per pixel. The dataset consists of 24 orthorectified image chips with 6500 x 6500 pixels dimensions. The images cover a total area of approximately 480 square kilometers. The images are in RGB format, with three 8-bit channels.

The dataset includes corresponding geojson files for each image, which contain the coordinates of building footprints. The building footprints were created using a combination of manual annotation and machine learning-based methods. The annotations include the polygon coordinates of the buildings and the building class label. The dataset also includes a set of negative samples that do not contain buildings.

The dataset presents several challenges to building detection algorithms, including the presence of overlapping buildings, buildings of varying sizes and shapes, and buildings with complex architectures. Furthermore, the images contain speckle noise, which can make it difficult to accurately detect building footprints.

3. Data Preprocessing

Load the images

Remote sensing images in SpaceNet1 are stored in TIF file format. Loading these images involves reading the metadata and image data from the TIF file. The ‘Pillow’ (PIL) library provides functions for reading geospatial data from various file formats including TIF. The image data can be stored in a numpy array where the number of dimensions depends on the type of image. For instance, a 3-band RGB image will have dimensions (height, width, 3) where each pixel has 3 values corresponding to the Red, Green, and Blue bands. [5]

Preprocess the images

The process of preprocessing images encompasses the utilization of diverse methods on the image data to improve its quality and make it suitable for deployment in deep learning models. For the present investigation, a median filter will be employed to eliminate noise from the image. Subsequently, the Faster R CNN model's effectiveness will be evaluated by comparing its performance with and without the preprocessing technique. [6][7]

A median filter is a statistical filter commonly used for denoising images by replacing each pixel with the median value of its neighborhood. In remote sensing image datasets, it can be used to reduce speckle noise, which appears as bright and dark spots of varying intensity, obscuring important features and reducing the accuracy of analysis algorithms.

The median filter can effectively reduce speckle noise while preserving edges and details in the image. It can be applied to each spectral band separately or to a combination of bands and is often used as a pre-processing step before further analysis. The optimal kernel size depends on the level of noise in the image and the desired level of smoothing. In this case, a 3x3 kernel size can be a suitable choice for a median filter on the SpaceNet1 dataset because it is a commonly used size that can effectively reduce noise in many types of images without blurring or distorting important details.[8]

As noises have really irregular pixel values, in contrast with their neighbors, applying a median filter might bring its value back on a more familiar and standard scale, therefore achieving denoising on the image.

As speckle noises are quite common noises in Radar Imagery, and remote sensing images, we found it to be a suitable filter for our preprocessing stage.

Figure 1. Image Before Denoising

Figure 2. Image After Denoising

Visualizing the detection

The geojson files in the SpaceNet1 dataset contain information about the location and shape of building footprints in the corresponding satellite image. Each geojson file corresponds to a single satellite image and contains multiple polygons that represent individual buildings in the image.

Each polygon in the geojson file is defined by a set of coordinates that represent the vertices of the polygon. The coordinates are given in longitude and latitude values, which correspond to the geographic location of the polygon in the Earth's surface. The geojson file also includes additional information about each polygon, such as a unique identifier, a confidence score, and a classification label.

The geojson files in the SpaceNet1 dataset are used to evaluate the performance of building detection algorithms by comparing the detected building footprints with the ground truth annotations provided in the geojson files. The geojson files are an important component of the SpaceNet1 dataset, as they provide a standardized and consistent format for annotating building footprints across different satellite images and enable the evaluation of building detection algorithms in a systematic and objective manner.

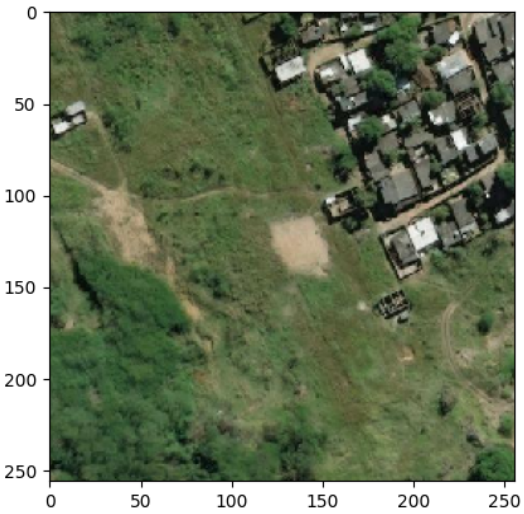

Figure 3. Original Image

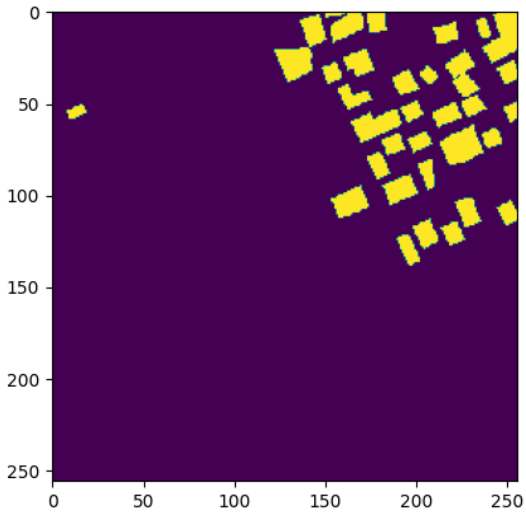

Figure 4. Geojson mask highlighting buildings

Finally, after the preprocessing process, the dataset (including both TIF and geojson files) is partitioned such that 70% of the data is allocated for training the model, 20% for validation, and 10% for testing.

4. Methods

Faster R-CNN is a popular object detection algorithm that was introduced in 2015 by researchers at Microsoft Research. It is an improvement over previous object detection algorithms such as R-CNN and Fast R-CNN. The main idea behind Faster R-CNN is to replace the selective search algorithm used in earlier models with a region proposal network (RPN) that can generate proposals for object regions more efficiently. The RPN shares convolutional layers with the detection network and is trained end-to-end with the detection network. This allows for faster and more accurate detection of objects.

Faster R-CNN consists of three main components:

1. **Convolutional neural network (CNN) backbone:** This is the feature extractor network that is used to extract features from the input image. Common architectures include VGG, ResNet, and Inception.

2. **Region proposal network (RPN):** The RPN generates region proposals that are likely to contain objects of interest. It operates on the feature maps generated by the CNN backbone and proposes candidate object bounding boxes by sliding a small network over the feature map.

3. **Object detection network:** The object detection network takes the proposed regions from the RPN as input and predicts the class of the object in each region, as well as the bounding box coordinates for the object.

In this study, the 'fasterrcnn_resnet50_fpn' model imported from the 'Pytorch' library is used as a Faster R-CNN model that uses ResNet50 as the backbone network and Feature Pyramid Network (FPN) to extract feature maps at different scales. [9] Here is an overview of the architecture:

1. **Convolutional Layers (Backbone):** The backbone of Faster R-CNN ResNet 50 FPN is a ResNet 50 network with a feature pyramid network (FPN). The input to the network is the image, and the output is a feature map with high-level abstract features that represent the image. The 50 convolutional layers use the Rectified Linear Unit (ReLU) activation function, which is defined as:

   $$f(x) = max(0, x)$$

   The input to each convolutional layer is the output of the previous layer, and the output is a tensor with a higher level of abstraction. In this case, the input includes the image, parsed via PIL, while the target is a Python dictionary containing information derived from its corresponding geojson file:

   a. Boxes: Bounding boxes generated from the coordinates of available building Polygons or Multipolygons

b. Area: Corresponding area of the bounding boxes

c. Image_Id: The unique number of the image, used to distinguish itself from other geojson files, and to track the original photo

d. Iscrowd: The annotation specifies whether it is for a single object or multiple objects, and for identifying buildings, it is set to 0 as we only want to detect buildings

e. Labels: The label assigned to each bounding box is 'building' or '1'.

2. **Region Proposal Network (RPN):** The RPN generates a set of object proposals by sliding a small network over the convolutional feature map produced by the backbone network. The RPN has two outputs, a set of objectness scores for each proposal and a set of regression offsets to refine the proposals. The objectness score is a binary classification score that indicates whether a proposal contains an object or not. The RPN loss function is a combination of the objectness score loss and the bounding box regression loss.

3. **Region of Interest (RoI) Pooling Layer:** The RoI pooling layer takes the output of the RPN and applies a pooling operation to each region of interest (RoI) to produce a fixed-size feature map. The RoI pooling layer is used to extract features from each proposal for classification and bounding box regression.

4. **Classification and Bounding Box Regression Layers:** The output of the RoI pooling layer is fed into two parallel fully connected layers. One layer performs binary classification to determine whether an object is present or not, while the other layer performs bounding box regression to predict the coordinates of the object's bounding box. The classification and bounding box regression loss is the sum of the binary cross-entropy loss and the smooth L1 loss.

5. **Loss Function:** The loss function used in Faster R-CNN ResNet 50 FPN is a combination of four different loss functions: the RPN objectness score loss, the RPN bounding box regression loss, the classification loss, and the bounding box regression loss.

6. **Optimizer:** The optimizer used in Faster R-CNN ResNet 50 FPN is Stochastic Gradient Descent (SGD) with momentum. The learning rate is set to 0.02, and the momentum is set to 0.9.

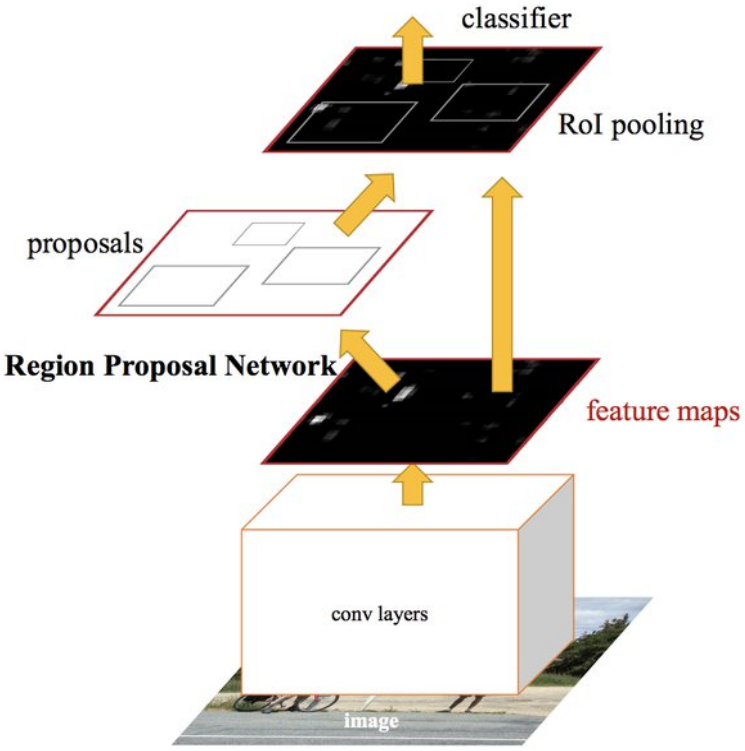

Figure 5. The general architecture of the Faster R-CNN model

The output of the Fast R-CNN ResNet-50 FPN model will be a set of bounding boxes that represent the predicted locations of buildings in the input satellite image. Each bounding box is represented by four values: the x-coordinate and y-coordinate of the top-left corner of the bounding box, the width, and the height of the bounding box. Programmatically, the returned results will be in the form of a list of tuples, with each tuple containing the coordinates of the bounding box of a predicted building, in the form of x-min, y-min, x-max, y-max.

In addition, the output includes the confidence score for each bounding box, which indicates how likely the model believes that the bounding box contains a building. The metric used to evaluate the performance of the model is the Intersection over Union (IoU) score, which takes into account the ground truth and predicted bounding boxes, measures their overlap, and indicates the model's fitness. The IoU score ranges from 0 to 1, with higher scores indicating better predictions. [10]

We utilized pre-trained weights of the Faster R-CNN ResNet 50 FPN model on the COCO dataset to avoid the need for expensive computations and instead concentrate on the evaluation.

5. Results and Discussion

Results

Without Denoising

Figure 6. Image with predicted bounding box

With Denoising

Figure 7. Denoised image with predicted bounding box

Discussion

The Fast R-CNN ResNet50 FPN model with pre-trained weights from the COCO dataset has not been able to correctly detect buildings from the SpaceNet1 dataset. Despite the success of the model in object detection tasks on the COCO dataset, the transfer of knowledge from one dataset to another can be challenging due to variations in image quality, object sizes, and other factors. In this case, it appears that the differences in building types and features between the two datasets have posed a significant challenge for the model. While pre-training on a large dataset like COCO can provide a good starting point, it may still be necessary to fine-tune the model on the target dataset or collect additional training data to improve its performance. It is important to note that evaluation metrics such as IoU may not be relevant in cases where the model fails to detect objects accurately.

There are several potential reasons why the Faster R-CNN ResNet50 FPN model with pre-trained weights from the COCO dataset may have failed to detect buildings in the SpaceNet1 dataset. One possibility is that the COCO dataset contains a different distribution of object classes compared to the SpaceNet1 dataset, meaning that the model's learned features may not generalize well to the SpaceNet1 dataset. Additionally, the COCO dataset may have different levels of image resolution, quality, and variability than the SpaceNet1 dataset, which can also affect the model's ability to detect buildings.

Another potential issue is that the SpaceNet1 dataset may contain specific types of buildings or architectural styles that the model was not trained to detect. The model may have also been trained on images with different lighting conditions, angles, or weather patterns, making it difficult to generalize to the SpaceNet1 dataset. Additionally, the model may not have been trained on a sufficiently diverse set of images, or with enough examples of buildings in different contexts, making it difficult for the model to learn robust representations of buildings.

It is also possible that the model was not fine-tuned or trained specifically for the SpaceNet1 dataset, which may have affected its ability to detect buildings accurately. Fine-tuning the model on the SpaceNet1 dataset or training a new model specifically for this dataset may improve performance. Finally, it is important to note that deep learning models are not perfect and may not always be able to accurately detect objects in all contexts or datasets.

6. Difficulties

Faster R-CNN models are commonly used for object detection in computer vision tasks. However, some existing Faster R-CNN models use outdated libraries that have been discontinued or deprecated by the library publishers [11]. This has caused us issues when trying to train and save the weights of the model in a local running environment. In order to resolve this problem, pre-trained weights of the Faster R-CNN resnet 50 fpn model on the COCO dataset has been used instead. Pretrained weights on COCO dataset for Faster R-CNN ResNet50 FPN is a set of pre-learned parameters that allow the model to recognize and detect objects in images. The COCO dataset is a large-scale object detection, segmentation, and captioning dataset. The weights are trained on this dataset, which includes 80 object categories, such as people, animals, vehicles, and various household and everyday objects. This approach has helped to avoid the computing expensiveness of the model and allow for a focus on evaluating the results [12].

Moreover, due to time limitations, analyzing the provided data has been proven troublesome. Creating a custom dataset was necessary, and extracting information from geojson provided us with a different set of coordinates system that proved difficult for a dedicated Faster R-CNN model.

Finally, since the model has failed to accurately detect the buildings, the IoU metric is no longer applicable in evaluating its performance. This suggests that the model may require additional fine-tuning, more comprehensive training data, or changes to its architecture.

References

1. Ren, S., et al. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. Advances in neural information processing systems, 91-99.

2. Zhang, H., et al. (2021). Building extraction from high-resolution remote sensing images using an improved Mask R-CNN approach. Remote Sensing, 13(4), 705.

3. Almeida, T. A., et al. (2018). Building extraction from very high-resolution satellite imagery using fully convolutional neural networks. Remote Sensing, 10(5), 719.

4. Karydis, K., et al. (2018). The SpaceNet 1 dataset: High-resolution imagery for remote sensing of urban areas. IEEE Geoscience and Remote Sensing Magazine, 6(3), 16-36. doi: 10.1109/MGRS.2018.2846294.

5. Clark, A., & Stubblefield, M. (2019). Python Imaging Library Handbook (2003). Python Imaging Library.

6. Wang, J., Zhou, W., Liu, C., Zou, X., & Liu, C. (2019). Deep Learning for Remote Sensing Data: A Technical Tutorial on the State of the Art. IEEE Geoscience and Remote Sensing Magazine, 7(3), 8-36. https://doi.org/10.1109/MGRS.2019.2913850

7. Bovolo, F., Bruzzone, L., & Chiaradia, M. T. (2003). Adaptive Median Filter for Speckle Removal in SAR Images. IEEE Transactions on Geoscience and Remote Sensing, 41(11), 2455-2464. https://doi.org/10.1109/TGRS.2003.817826

8. Wang, J., Zhou, W., Liu, C., Zou, X., & Liu, C. (2019). Deep Learning for Remote Sensing Data: A Technical Tutorial on the State of the Art. IEEE Geoscience and Remote Sensing Magazine, 7(3), 8-36. https://doi.org/10.1109/MGRS.2019.2913850

9. The PyTorch documentation for the torchvision.models.detection module, which includes the implementation of Faster R-CNN ResNet 50 FPN in PyTorch: https://pytorch.org/vision/stable/models.html#object-detection-instance-segmentation-and-person-keypoint-detection

10. V7 Labs. (n.d.). Bounding Box Annotation. Retrieved March 30, 2023, from https://www.v7labs.com/blog/bounding-box-annotation

11. https://learn.microsoft.com/en-us/azure/machine-learning/how-to-inference-onnx-automl-image-models

12. https://www.v7labs.com/blog/yolo-object-detection