CS230

Multispectral Satellite Imagery Feature Detection

Category: Computer Vision

Elvis Dowson
Artificial Intelligence Graduate Certificate
SCPD, Stanford University
SUNet ID: edowson

Abstract

This document is the final project report for the detection and classification of features in satellite imagery acquired using multispectral sensors.

1 Introduction

Multispectral sensors generally capture image information across several wide-bands (typically 3 to 10) of the electromagnetic spectrum. Hyperspectral images on the other hand, can consist of hundreds to thousands of much narrower-bands. Surfaces of different objects reflect or absorb light and radiation in different ways. The ratio of reflected light to incident light is known as reflectance and is expressed as a percentage.

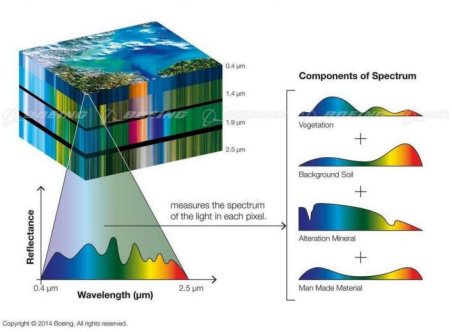


Figure 1: Multi-spectral / Hyperspectral Imaging Technology

Having multi-spectral or hyper-spectral data provides additional spectral channels to detect and classify objects in a scene based on its reflectance properties across several bands. This technique can be used to detect and classify objects, additionally based on its reflectance, such as: buildings and man-made structures, roads, vegetation, water bodies, and vehicles.

Multi-spectral and hyper-spectral imaging technology [1] have many real world applications and are used in the fields of agriculture, ecology, oil and gas, oceanography and atmospheric studies.

[1] Hyperspectral Imaging Technology, Copyright © 2014 Boeing.

2 Related Work

Existing approaches have used the U-Net architecture to perform the binary segmentation of roads using the Massachusetts Roads and Buildings Dataset [8]. Others have used an ensemble of U-Net neural networks to classify and segment 10 classes of objects using multi-spectral data using the Kaggle DSTL competition dataset [9][10].

Another approach uses a variation of U-Net called the Tiramisu Architecture to classify and segment objects belonging to a single class (either buildings or roads) using the SpaceNet multi-spectral dataset [11]. Finally, others have used the Single Shot MultiBox Detector (SSD) method to segment and extract building footprints [12].

3 Dataset and Features

3.1 Multi-Spectral Satellite Images

The dataset used for this project is based on the one provided by Kaggle for the DSTL Satellite Imagery Feature Detection (SIFD) competition [1]. It consists of a total of 25 labelled images captured across 20 different bands with varying number of channels and resolutions. The RGB-Band dataset is synthetically generated from the P-Band and M-Band using PanSharpening.

The ground truth masks were generated using labelled polygonal shape data for 10 classes for each of the 25 images in the dataset, which includes buildings, structures, roads, tracks, trees, crops, water-ways, standing waterbodies, trucks and cars.

3.1.1 Issues with the Dataset

The total number of labelled training samples is quite small and there is a heavy imbalance in the class distribution in the training dataset. Additionally, obtaining a larger labelled dataset was infeasible for the scope of this project because of the large effort required to manually label the remaining images in the dataset for all the 10 classes using multi-polygons.

There are 32 private test images which are not supplied with the downloadable dataset but you can submit your predictions to the Kaggle to obtain the performance scores against their test set. This causes another problem because the validation and test datasets do not come from the same distribution.

The number of channels and the resolution of the images for different bands vary significantly, e.g. 3-ch RGB-band (3348 x 3403), 1-ch P-Band (3348 x 3403), 8-ch M-Band (837 x 851) and 8-ch A-Band (134 x 137).

The images captured by the satellite for the different bands are also not captured at the same time, it is shot in such a way that in the M-Band, channels 2, 3, 5 and 7 are shot a few seconds after channels 1, 4, 6 and 8. This leads to ghost artifacts in the moving objects, as well as a shift in the images due to the motion of the satellite over the earth.

Each sample in the dev and test set was manually reviewed to ensure that there was a good representation of classes and a good distribution of features. Towards the end, a further 9 samples were removed from the train set that consisted mostly of crops and vegetation, in order to balance out the classes in the dataset, more or less equally, for the 10 classes.

3.1.2 Data Augmentation

Because of the small dataset, the data samples were augmented by randomly cropping and flipping the images before sending it to the model via a dataset data-loader. This dataset is split using 8:1:1 ratio to for the train, validation and test samples. The number of samples seen by the neural network model depends on the batch size and number of epochs during train, validation and test.

In the end, the model was trained on 11 train, 2 dev and its performance evaluated with 3 test original high-resolution images that were fed to the network using data augmentation.

3.1.3 Image Pre-Processing

A series of image pre-processing steps are order to overcome the problem of varying image resolutions and image misalignment.

The RGB images were normalized in the range [0,1] and processed to improve visual perception by performing scale percentile processing, which involved filtering out the lower 1 and higher 99 percentile pixel values.

The low resolution images are up-scaled to the same resolution as the high-resolution RGB images. The upscaling process for the low-resolution A-band images yielded visually good results when using OpenCV functions. A further improvement could be made using Super-Resolution with a CNN, but this hasn't been done yet, at the time of writing this report.

To correct the issue of image alignment and registration, the high-resolution RGB image is used as a reference to compute the warp matrix transforms to apply euclidean transforms to the misaligned image using OpenCV function. This too yielded acceptable results and registration for the up-scaled low-resolution A-Band images. The borders of the images were padded with border reflect to account for the shift in the images as a result of the alignment step. The area around the image borders is then cropped to remove the borders to discard the artificially generated border padding to prevent errors when comparing the input image with the generated labelled ground truth mask image during training.

4 Approach

The approach used a U-Net [3] to solve an initial binary segmentation problem for the detection of buildings in Satellite Images [2], using the 3-ch RGB images.

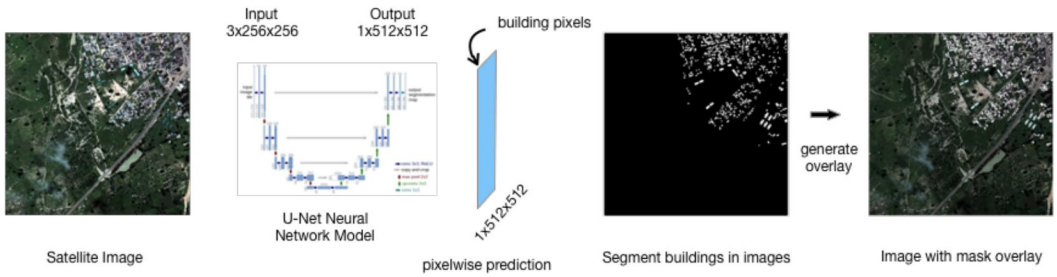


Figure 2: Building Feature Detection from Satellite Images using U-Net Neural Network Model

4.1 U-Net Neural Network

The convolution neural network used was based on the U-Net architecture developed by Ronneberger, et al [4], which is characterized by a U-shaped sequence of traditional CNN contracting layers followed by an equal number of expanding layers with skip connections. It has been shown to be effective in biomedical semantic segmentation tasks, with small datasets. It has proved to be equally effective at binary classification of objects from satellite images using a small dataset.

4.2 Loss Function

We define the loss function as:

$$\mathcal{L}(y_{\text{true}}, y_{\text{pred}}) = \text{binary cross entropy loss} + (1 - \text{dice co-efficient}) \tag{1}$$

where:

[2]Satellite Imagery © DigitalGlobe, Inc.

$$\text{binary cross entropy loss} = -\sum(y_{\text{true}} log(y_{\text{pred}}) + (1 - y_{\text{true}}) log(1 - y_{\text{pred}}))$$

$$\text{dice co-efficient} = \frac{1}{N} \sum_{i=1}^{N} \frac{2 \sum(y_{\text{true}} \times y_{\text{pred}}) + 1}{\sum y_{\text{true}} + \sum y_{\text{pred}} + 1}$$

4.3 Evaluation Criteria

The performance of the neural network model was evaluated by measuring the dice coefficient during train, validation and test.

5 Experiments

In terms of experiments, I used an Adam optimizer with a learning rate of 0.001 for gradient descent with an LR scheduler to train the neural network. I used an input size of 3x512x512 with a batch size of 3. These experiments were run on an i7 4.2GHz single CPU system with 48GB of DDR4 RAM and 3 x NVIDIA Titan V 12GB HBM2 GPUs using PyTorch 0.4.0.

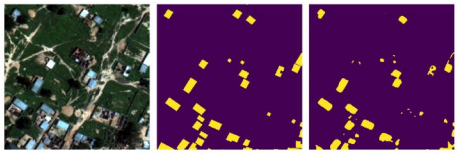
A batch size of 3 distributed over 3 GPUs meant that each GPU was effectively processing a single image in parallel and it maxed out the GPUs memory to about 90%. This resulted in very noisy oscillation for the loss during training, and it could have been improved by increasing the batch size.

Further investigation revealed that the best accuracy was obtained at around the 69th epoch for a 300 epoch run. This leads to the conclusion that the training performance could be improved further by increasing the batch size and size of the dataset, because the model might have over-fit on the training data. The test results were good however which indicates that the train set was harder than the test set.

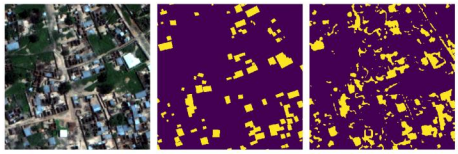
5.1 Results

The following images show the performance of the U-Net neural network model during training, validation and testing. The model shows an acceptable level of performance during train, a relatively lower performance during validation and a similar performance on par with the validation set by looking at the IOU (Intersection Over Union).

We can conclude from this observation that there is a variance problem, the model has over-fit the data and that the model needs to be regularized. This can be solved by getting more data, increasing the batch size and reducing the learning rate. Batch normalization has already been added to the layers. Adding a small dropout factor of 0.1 can help prevent over-fitting and aid in generalization of the neural network.

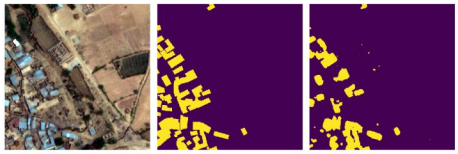


(a) Train Image 299, Mask & Prediction



(b) Validation Image 299, Mask & Prediction

(c) Test Image 1, Mask & Prediction



(d) Test Image 2, Mask & Prediction

Figure 3: Model prediction for building classes during training, validation and testing

5.2 Metrics

The metrics for train, dev and test for the initial run with an input size of 512x512, batch size 3 for epoch 289, using a graph smoothing factor of 0.5 have been summarized in Table 1.

Metric	Train	Dev	Test
Accuracy	0.3692	0.2859	0.5410
Loss	0.6920	1.219	0.9266

Table 1: Evaluation metrics with 512x512 input, batch size 3 for 300 epochs

The differences in metrics between the train and dev set indicate that I potentially have a variance and data mismatch problem. The difference between dev and test set, with the test set having a higher accuracy score than the dev set, indicate that the test set was probably easier than the dev set.

5.3 Discussions

This was a challenging project to work on, due to small size of the dataset and the amount of preprocessing required for the input images. This required data augmentation of the high-resolution images, in terms of random cropping of images and flipping.

Midway during the project, I discovered that the Python Imaging Library (PIL), used internally by PyTorch TorchVision, skimage and the Augmenter library, could not handle multi-channel images and kept truncating multi-channel images and masks to 3-channels. I had to temporarily restrict the inputs to 3-ch RGB images for the initial scope of this project.

It took a lot of effort to get a working U-Net model with PyTorch, largely due to errors on my part, in calculating loss and accuracy metrics, due to differences in channel ordering, when dealing with Torch Tensors converted to Numpy arrays. The U-Net model performed well after training for 300 epochs with a batch size of 3, using 3-ch RGB images with a 512x512 input resolution, but this probably was a combination of balancing the distribution of classes in the train and dev set and over-fitting due to the small size of the dataset.

6 Conclusion

The U-Net architecture appears to be suitable for the task of binary segmentation of building features from satellite images. However, it remains to be seen if it can scale well to the task of multi-class segmentation using a single network. Additionally newer architectures such as SegCaps [5] might offer better performance and accuracy for the same task.

6.1 Future Work

In terms of future work, I plan on completing the following tasks:

1. Update PyTorch's TorchVision library to use an OpenCV 3 imaging backend to handle multi-channel GeoTIFF images.
2. Repeat the U-Net binary classification experiment for buildings with a larger dataset.
3. Extend the U-Net network and evaluate its suitability for the task of multi-class image segmentation, for the 10 labelled classes in the DSTL SIFD dataset.
4. Evaluate the use of a capsule network based SegCaps [5] model for binary segmentation and multi-class image segmentation; compare its performance with the U-Net architecture.
5. Incorporate additional channels (e.g. Short Wave InfraRed A-Band 7, and additional reflectance indices for water and vegetation) to incorporate additional spectral information and evaluate its impact on neural network model performance.

References

[1] Dstl Satellite Imagery Feature Detection.
https://www.kaggle.com/c/dstl-satellite-imagery-feature-detection/data

[2] Wen, Jeff. "Road Extraction using PyTorch".
https://jeffwen.com/2018/02/23/road_extraction/

[3] Iglovikov, Vladimir, Sergey Mushinskiy, and Vladimir Osin. "Satellite Imagery Feature Detection using Deep Convolutional Neural Network: A Kaggle Competition." arXiv preprint arXiv:1706.06169 (2017).

[4] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." International Conference on Medical image computing and computer-assisted intervention. Springer, Cham, 2015.

[5] LaLonde, Rodney, and Ulas Bagci. "Capsules for Object Segmentation." arXiv preprint arXiv:1804.04241 (2018).

[6] Hinton, Geoffrey, Nicholas Frosst, and Sara Sabour. "Matrix capsules with EM routing." (2018).

[7] Sabour, Sara, Nicholas Frosst, and Geoffrey E. Hinton. "Dynamic routing between capsules." Advances in Neural Information Processing Systems. 2017.

[8] Mnih, Volodymyr. "Machine Learning for Aerial Image Labeling". Massachusetts Road and Building Detection Datasets.
https://www.cs.toronto.edu/ vmnih/data/

[9] Dstl Satellite Imagery Competition, 1st Place Winner's Interview: Kyle Lee.
http://blog.kaggle.com/2017/04/26/dstl-satellite-imagery-competition-1st-place-winners-interview-kyle-lee

[10] Dstl Satellite Imagery Competition, 3rd Place Winner's Interview: Vladimir & Sergey.
http://blog.kaggle.com/2017/05/09/dstl-satellite-imagery-competition-3rd-place-winners-interview-vladimir-sergey

[11] Solving SpaceNet Road Detection Challenge With Deep Learning.
https://devblogs.nvidia.com/solving-spacenet-road-detection-challenge-deep-learning

[12] Chartock, Elliott, Whitney LaRow, and Vijay Singh. "Extraction of Building Footprints from Satellite Imagery."
http://cs231n.stanford.edu/reports/2017/pdfs/550.pdf

[13] DigitalGlobe Platform - Earth Imaging Basics.
https://platform.digitalglobe.com/category/earth-imaging-basics