Introduction

Object detection is a recurring problem in computer vision that involves finding and classifying objects in an image. For example, a self-driving car would want to find and classify objects such as pedestrians, other cars, traffic lights, and the road. The detection problem is distinct from pure classification and localization in that many different kinds of objects may be in the image and a variable number of objects may also be present (or not at all). A common paradigm in object detection (and the broader problem of classification) is to extract features from images then train a classifier on said features. Both steps have several available options. Feature extraction can be carried out using histogram of oriented gradients (HOG), local binary patterns (LBP), and filter banks, just to name a few. Similarly, classification of these extracted features can be carried out using any number of machine learning algorithms, such has k nearest neighbors (kNN), support vector machines (SVM), and random forests (RF). Here, I present two solutions to a toy object detection problem – segmenting rivers from background in a series of aerial images of forests. First, an RF classifier is trained and validated on features extracted by HOG and LBP. Second, segmentation is applied using a sliding window across validation and external testing set images. Finally, dice coefficients are computed to compare the predicted segmentation labels to a ground truth.

Methods

Dataset

There were three aerial images provided originally – two large images (16384x16384) and one smaller image (4000x6000). For the purposes of cross-validation (and brevity), one 1280x1280 image was extracted from each large image. Additionally, a manually-created segmentation mask was generated for these extracted images as well as the smaller image.

For

Problem Definition: detection (classification) and segmentation of rivers in aerial photos

Methodology: classification will be carried out using HOG and LBP descriptors and RF; segmentation will be carried out using a sliding window; training and validation on two large images and testing on third (unseen) image

Patch size 64

-fits inside the object we want

-number of features is small enough not to get into trouble with dimensionality

Train on 1

Training acc 0.999999999999999

Confusion matrix

10000 0

0 10000

Validation acc 0.9999046154642316

Confusion matrix

4037 11

0 4048

Val dice

.2894

.8053

Train on 2

Train 1.0

Confusion matrix

Validiation 0.9987514749999999

Confusion matrix

9933 67

438 9562

Val dice

.8481

.9501