Πολυτεχνείο Κρήτης

Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Comparison of Artificial Intelligence systems for the detection of objects on UAV-based images.

ΤΡΙΜΑΣ ΧΡΗΣΤΟΣ

ΧΑΝΙΑ, ΙΟΥΝΙΟΣ 2021

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ

Καθηγητής Μιχαήλ Ζερβάκης (Επιβλέπων)

Αναπλ. Καθηγητής Μιχαήλ Λαγουδάκης

Καθηγητής Ευριπίδης Πετράκης

Ευχαριστίες

Περίληψη

Abstract

Contents:

1.Introduction

2.Object Detection

2.1 Definition

Object detection, is a computer technology related to computer vision and image processing, that deals with detecting instances of semantic objects of a certain class in digital images and videos. In other words, given an image or a video stream, an object detection model can identify which of a known set of objects might be present and provide information about their positions within an image.

2.2 Uses

With the evolution of cameras and the oversimplification of data gathering and processing, object detection can be used in the following areas/industries:

a) Video Surveillance.
b) Search and Rescue missions.
c) Anomaly detection.
d) Self-driving vehicles.

2.3 Concept

Every object class has its own special features that helps in classifying the object to its own class. Object detection models take as input images and try to understand those features, so that it can classify objects in the same class.

2.4 Methods

Methods for object detection fall into machine learning-based approaches. It is vital for that kind of approaches, to first define features using on of the following algorithms:

- Viola-Jones object detection framework based on Haar features.
- Sliding Window + Image pyramid.
- Scale-invariant feature transform or SIFT.
- Histogram of oriented gradients or HOG, features.

After the definition of the features, classification methods such as Support Vector Machines, can be used to do the classification.

While the sliding-window approach was the leading paradigm in classic computer vision, with the resurgence of deep learning, single-stage and two-stage detectors came to dominate object detection. Deep learning techniques are able to do end-to-end object detection without specifically defining features, and are typically based on Convolutional Neural Networks or CNN.

As previously stated, there are two kinds of deep learning detectors. Single-stage and two-stage detectors.

Two-stage detectors use a Region Proposal Network to generate regions of interest in the first stage, and send the region proposals down the pipeline for object classification. Usually those models have higher accuracy, but the inference time is quite big. Most known algorithms are:

- R-CNN
- Fast/er R-CNN

On the other hand, single-stage detectors take an input image and learn the class probabilities and bounding box coordinates. Such models reach lower accuracy, but they are much faster than two-stage detectors. Most known algorithms are:

- RetinaNet
- You Only Look Once or YOLO
- Single Shot MultiBox Detector or SSD

3.Models

Since the purpose of this diploma thesis is to compare models, the following algorithms were chosen.

3.1 RetinaNet

RetinaNet is a single, unified network composed of a backbone network and two task-specific subnetworks. The backbone, is responsible for computing a convolutional feature map over an entire input image and is an off-the-self convolutional network. More specifically, as the backbone of the RetinaNet the Feature Pyramid Network has been adopted. FPN is a standard convolutional network with a top-down pathway and lateral connections so the network efficiently constructs rich, multi scale feature pyramid from a single resolution input image. Each level of pyramid can be used for detecting objects at a different scale. Besides the backbone network, there are also two more subnetworks. The first one is responsible for object classification on the backbone's output and the second subnet performs convolutional bounding box regression.

3.1.1 Feature Pyramid Network

Traditionally, in computer vision, featurized image pyramids have been used to detect objects with varying scales in an image. Featurized image pyramids are feature pyramids built upon image pyramids. This measns one would take an image and subsample it into lower resolution and smaller size images, thus forming a pyramid. Hand-engineered feaures are then extracted from each layer in the pyramid to detect objects. This makes the pyramid scale-invariant and the process is quite intensive in terms of computation and memory.

With the development of deep learning, these hand-engineered features were replaced by CNNs. Later, the pyramid itself was derived from the pyramidical hierarchical structure that CNNs have (for example SSD). In CNNs the output size of feature maps decreases after each successive block of convolutional operation, and forms a pyramidal structure.

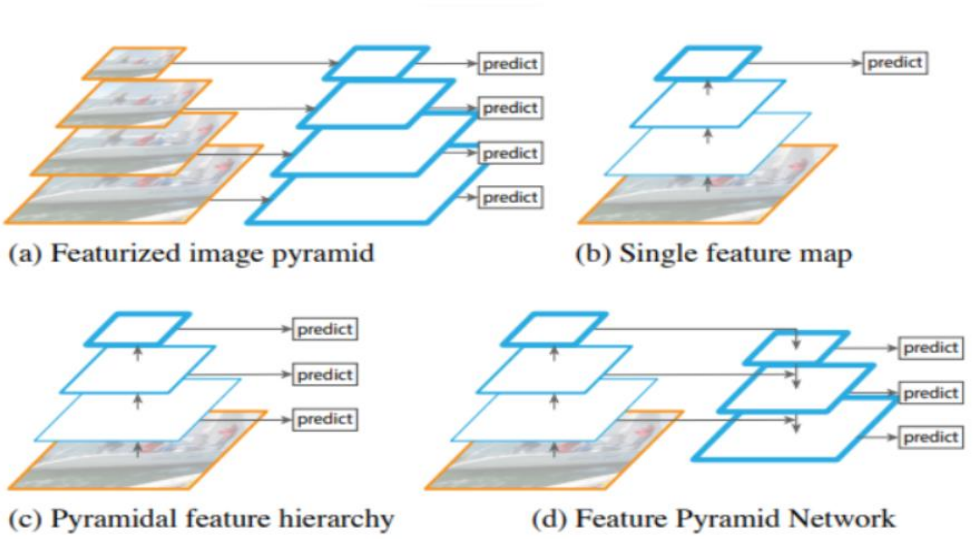

Image ?. Different types of pyramid architectures

Various architectures (Image ?) have been proposed that utilize the pyramid structure. In example (a), the Feature image pyramid that was discussed before is exhibited. As mentioned before, it is compute intensive. In example (b), the Single (scale) feature map is shown. Although it detects faster than (a), pyramids are still needed to get the most accurate results. Pyramidal feature hierarchy (c), utilize only semantically rich information, but it does not reuse the multi-scale feature maps from different layers and therefore small objects cannot be detected. Feature Pyramid Network or FPN (d), covers the flows of all the other architectures. FPN creates an architecture with rich semantics at all levels as it combines low-resolution semantically strong features, with high resolution semantically weak features. To achieve this, a top-down pathway with lateral connections to bottom-up convolutional layers has been created.

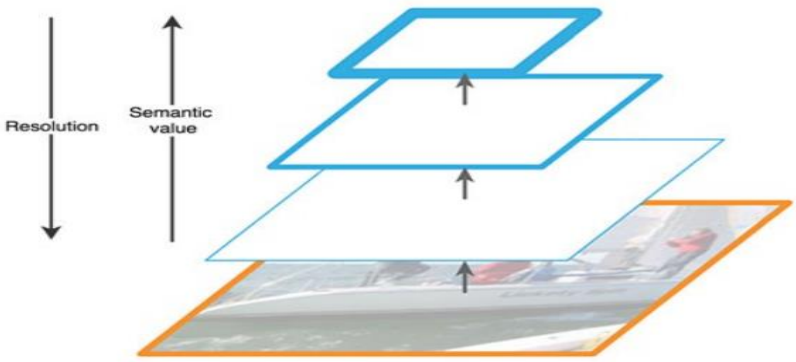

Image ?. Feature extraction in FPN

3.1.2 Anchors

When a Neural Network predicts multiple objects in a picture, the network is actually making thousand predictions and only showing the ones with the highest confidence of containing an object. In other words, the detector creates thousand of anchor boxes for each predictor that represent the ideal location, shape and size of the object it specializes in predicting. For each anchor box, it calculates which object's bounding box has the highest overlap divided by non-overlap. This is called Intersection over Union or IOU. If the highest IOU surpasses a certain threshold, then it signals the anchor box that it should detect the object that gave the highest IOU.

3.1.3 Classification/Regression subnetworks

The classification subnet, predicts the probability of object presence at each spatial position for each anchor and for each object class. This subnet is a small Fully Connected Network, attached to each FPN level, and the parameters of this subnet are shared across all pyramid levels. The design is simple. Taking an input feature map with X channels from a given pyramid level, the subnet applies four 3x3 convolutional layers, each with filters equal to the number of the feature map channels, followed by ReLU activations, followed by a 3x3 convolutional layer, with the number of classes times the number of anchors, filters. Finally, sigmoid activations are attached to output the binary predictions per spatial location.

In parallel, a small FCN is attached to each pyramid level for the purpose of regressing the offset from each anchor box, to a nearby ground-truth object, if one exists. The design is identical to the classification subnet, with the exception that it terminates in (4xNumberOfAnchors) linear outputs per spatial location. For each anchor per spatial location, these 4 outputs predict the relative offset between the anchor and the ground-truth box.

3.1.4 RetinaNet Architecture

The one stage RetinaNetwork architecture as previously stated, uses a Feature Pyramid Network on top of a feedforward ResNet architecture as backbone, in order to generate a rich, multi-scale convolutional feature pyramid.

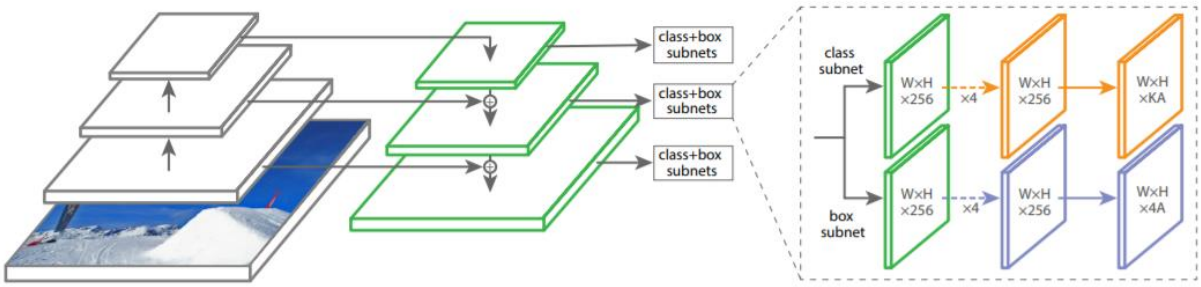

Image ?. RetinaNet architecture.

ResNet is used to construct the bottom-up pathway. It composed of many convolution modules, each with many convolution layers. Moving up, the spatial dimension is reduced by 0.5 or the stride is doubled. The output of each convolution module is labeled as C_i and later used in the top-down pathway (FPN).

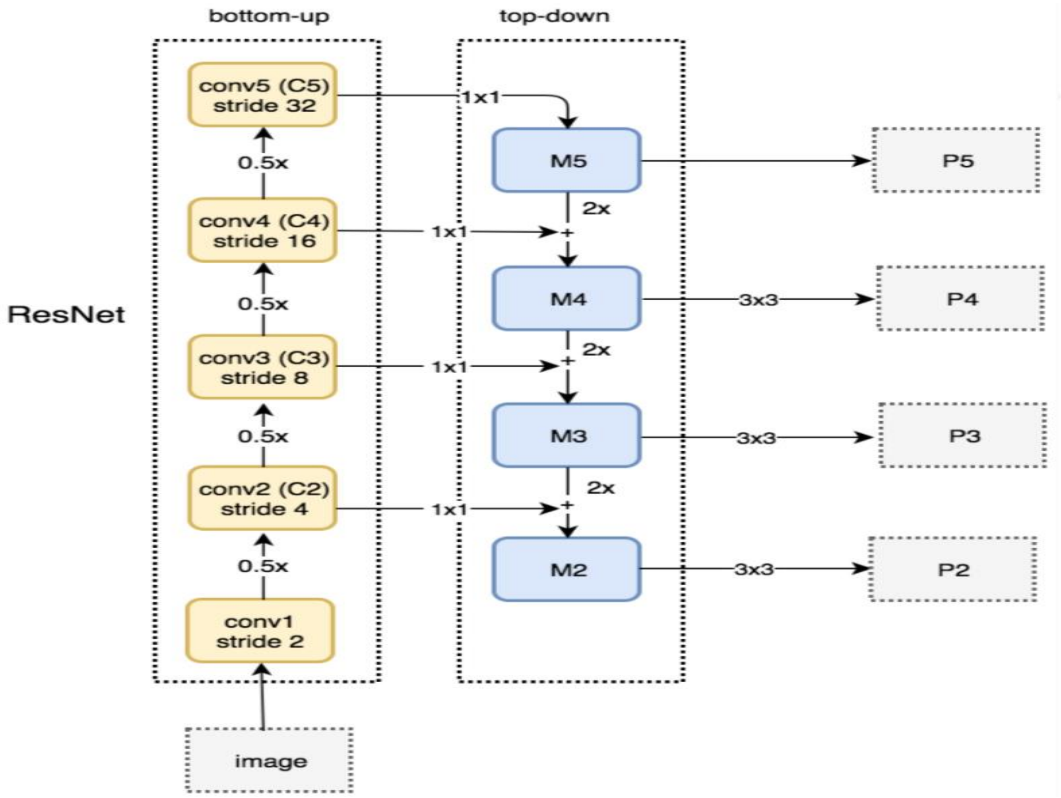

Image ?. Backbone architecture.

For the top-down pathway, a 1x1 convolution filter is applied to reduce the convolutional modules depth and create the feature map layers. As we go down the top=down path, the previous layers are getting upsample by 2, using nearest neighbors upsampling. Again, a 1x1 convolution is applied to the corresponding feature maps in the bottom-up pathway. Then they are getting added element-wise. Finally, a 3x3 convolution merges all layers. This filter reduces the aliasing effect when merginf with the upsampled layer. This process, stops for Conv1 module, because the spatial dimensions are too large and will slow down the process too much. Furthermore, since the classification and box regressor subnets, share the same inputs from the feature maps outputs, all pyramid feature maps have the same size output channels.

To this backbone, the RetinaNet attaches tow subnetworks that were described in section 3.1.3, one for classifying anchor boxes, and one for regressing from anchor boxes to ground-truth object boxes.

The design of the Network is simple but effective, because allows to focus more on optimization, using Focal Loss, an enhancement over Cross-Entropy Loss that eliminates the accuracy gap between RetinaNet, and state-of-the-art two-stage detectors like Faster R-CNN with FPN, while maintain a low inference time.

3.1.5 Loss Function

Single stage models suffer from an extreme foreground-background class imbalance problem due to dense sampling of anchor boxes. In RetinaNet, at each pyramid layer there can be thousands of anchor boxes, but only a few will be assigned to a ground-truth object, while the vast majority will be background class. These easy examples or detections with high probabilities, although resulting in small loss values can collectively overwhelm the model. Focal loss reduces the loss contribution from easy examples and increases the importance of correcting misclassified examples.

Starting from the cross entropy (CE) loss for binary classification:

$$CE(p, y) = \begin{cases} -log(p), & if\ y = 1 \\ -log(1-p), & otherwise \end{cases}$$

In the above $y \in \{\pm 1\}$ specifies the ground-truth class and $p \in [0,1]$ is the model's estimated probability for the class with label $y = 1$. Changing the notation for convenience and defining p_t:

$$p_t = \begin{cases} p, & if\ y = 1 \\ 1 - p, & otherwise \end{cases}$$

Now the CE can be rewritten as:

$$CE(p, y) = CE(p_t) = -\log(p_t)$$

A common method to address the class imbalance is to introduce a weighting factor $a \in [0,1]$ for class 1 and 1-α for class -1. The weighting factor can be treated as a hyperparameter to set by cross validation. Again for notational convenience the *α-balanced* CE loss function is defined as:

$$CE(p_t) = a_t \log(p_t)$$

As [writters] suggest, the large class imbalance encountered during training of dense detectors, overwhelms the cross-entropy loss. Easily or well classified negatives, comprise the majority of the loss and dominate the gradient. While α balances the importance of positive/negative examples, it does not differentiate between easy/hard examples. Therefore, a reshaped form of CE is proposed to down-weight easy examples and thus focus training on hard negatives.

By adding a modulating factor to the cross-entropy loss, the focal loss is defined:

$$FL(p_t) = -(1 - p_t)^{\gamma} \log(p_t)$$

The focal loss is visualized in the next figure for values of gamma between zero and five. It is important to state that if gamma equals to zero, then the focal loss function behaves like cross-entropy. Besides that, when an example is misclassified and p_t is small, the modulating factor is near 1 and the loss is unaffected. As p_t tends to become 1, the factor goes to 0 and the loss for well-classified examples is down-weighted.

In practice, an *α-balanced* form of the focal loss is preferred.

$$FL(p_t) = -a_t(1 - p_t)^{\gamma} \log(p_t)$$

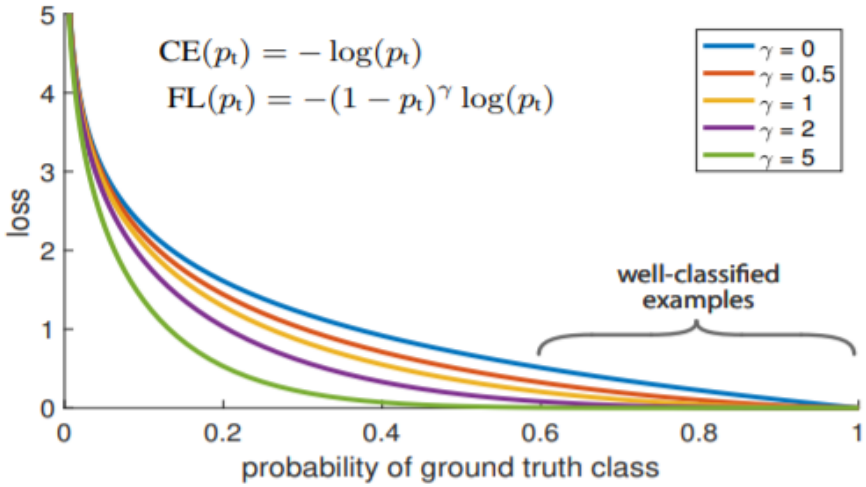

Image ?. Focal Loss.