Object Detection: Lecture 2

Ujjwal
ujjwal.ujjwal@inria.fr

From the previous class

- Basics of Detection.
- Faster-RCNN.

Today’s Class

- SSD
  - Theory
  - Implementation
- Feature Pyramid Networks.

SSD: **S**ingle **S**hot MultiBox **D**etector

Positive traits:

1. Very Simple
2. Very Fast

Negative Traits:

1. Very Experimental
2. Not easy to extend to new architectures

Get the Implementation

Download the implementation from here.

The implementation is in TensorFlow 1.x

It is one of the reliable non-official implementations.

In the rest of this presentation we will refer to files in this implementation.

The Basic Idea of SSD

- Take a CNN architecture.
- Add some extra layers to it.
- Take outputs from multiple CNN layers.
- Attach each output to detection specific components.
- Collect all the detections and perform post-processing.

The CNN architecture

- The original implementation was given for VGG16.
  - We will go by the original implementation.
- Refer to the function “ssd_net” (Line 377) in the file nets/ssd_vgg_300.py
- This function constructs the VGG16 network (Lines 392-412).
- It then goes on to add extra layers to the network (Lines 414-442).
  - These extra layers are experimental.
  - For other choices of base network, these extra layers could be different.

Detection Specific Components

- Lines 444-457 add detection specific components to specific layers.
- The names of these layers is given in line 78.
- We will now get to understand these detection specific components.

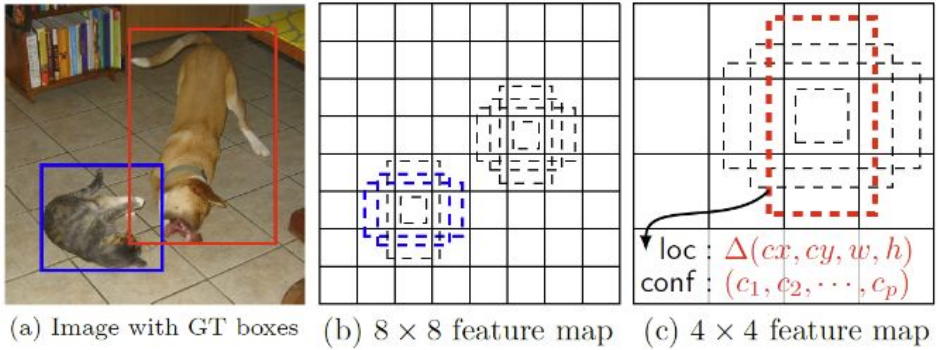
Detection Specific Components

There are 2 detection specific components:

1. Classification Layer.
2. Bounding Box Regression Layer.

Detection Specific Components: Implementation

- You can see in line 450 that a specific function is called to add detection specific components.
- This function is defined in line 350.
- Before we describe this function we should understand anchors or priors in SSD.



SSD: Multiple Bounding Boxes for Localization (loc) and Confidence (conf)

- After going through a certain of convolutions for feature extraction, we obtain **a feature layer of size *m*×*n* (number of locations) with *p* channels**, such as 8×8 or 4×4 above. And a 3×3 conv is applied on this *m*×*n*×*p* feature layer.
- **For each location, we got *k* bounding boxes.** These k bounding boxes have different sizes and aspect ratios. The concept is, maybe a vertical rectangle is more fit for human, and a horizontal rectangle is more fit for car.
- **For each of the bounding box, we will compute *c* class scores and 4 offsets relative to the original default bounding box shape.**
- Thus, we got **(*c*+4)*kmn* outputs.**

Detection Specific Components: Implementation

- In Line 350 the same concept has been implemented.
- And this is the whole model.

What happens during Testing ?

- During testing:
    -

Training

For training we need a couple of things:

1. Groundtruth bounding boxes with class labels.
    a. Comes from the training dataset.
2. Predicted bounding boxes.

Training: Groundtruth bounding boxes and labels

- Groundtruth bounding boxes and labels are mapped to individual feature maps.
  - It is the same thing as rescaling all the bounding boxes and getting their coordinates in individual feature maps.
  - Lines 158-195 in nets/ssd_common.py
- Once you are able to match the Groundtruth boxes to individual feature maps, it is easy to compute losses.

Loss Function

In SSD, the loss function has the form:

$$L(x,c,l,g) = \underbrace{\frac{1}{N}}_{\text{Number of matched boxes}}(\underbrace{L_{conf}(x,c)}_{\text{Classification Loss}} + \alpha \underbrace{L_{loc}(x,l,g)}_{\text{Regression Loss}})$$

Loss Function

- If N =0, then loss is set to zero.
- Classification loss is exactly the same as in Faster-RCNN.
- Regression loss is also exactly the same as in Faster-RCNN.

Loss Function: Implementation

- Lines 511-596 in nets/ssd_vgg_300.py

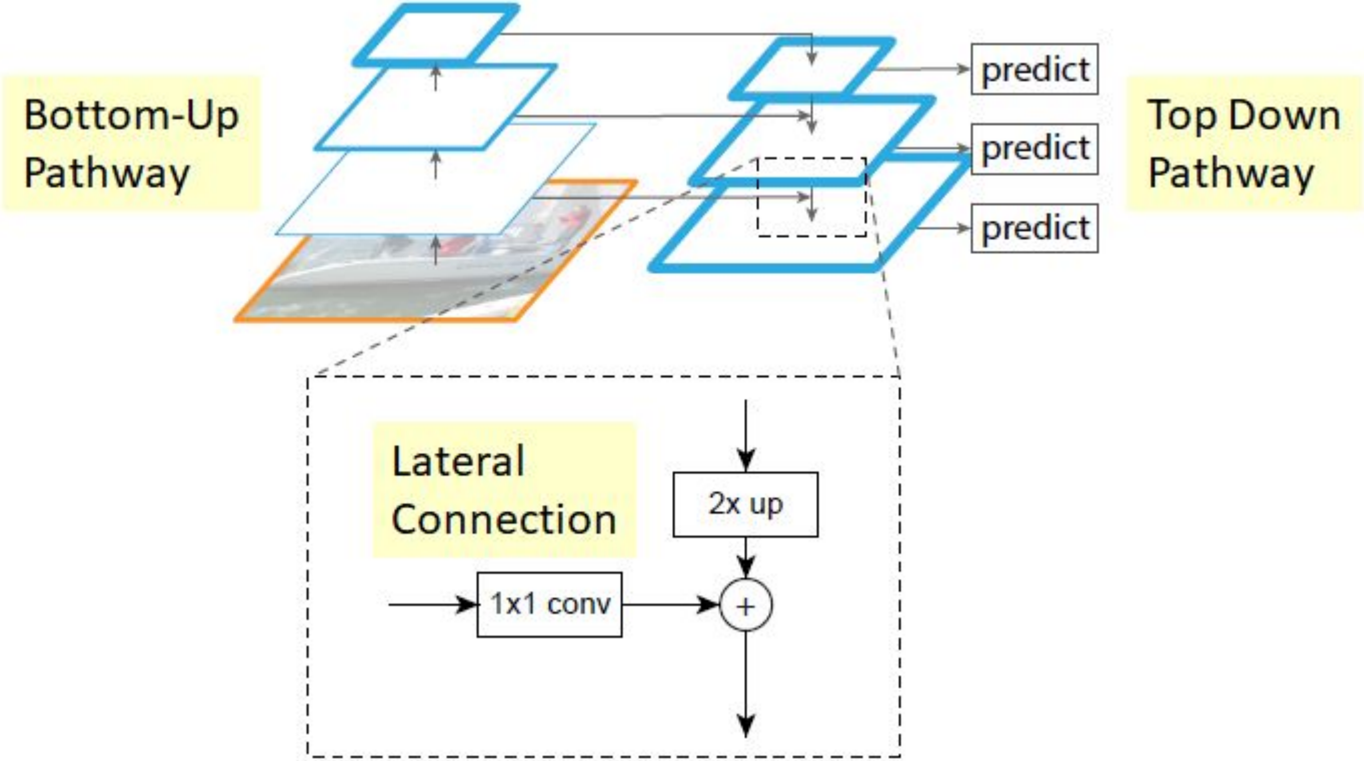
Constructing Anchors

- SSD anchors are basically the same as Faster-RCNN anchors.
- Their scales and aspect ratios are more carefully chosen.

Experiments

- Replace VGG16 with ResNet-101 in the provided implementation.
- Run the code using instructions provided in the README.md for Pascal VOC dataset.

Feature Pyramid Networks (FPN)



What is the basic idea here ?

- Detect objects from multiple layers
  - Same as SSD but more disciplined.
  - No more experimental addition of extra layers as in SSD.
- Utilize the power of higher and lower layers simultaneously.

Bottom-Up Pathway

- Refers to the feed-forward computation taking place as in a normal CNN architecture.

Top Down Pathway

Top down pathway has two operations:

1. Upsampling of higher layer feature maps.
2. Merging with corresponding feature maps in the bottom-up pathway.
   a. This involves a 1x1 convolution to make the number of filters equal to that in upsampled feature maps.
   b. The resulting feature map is added to the upsampled feature maps.
3. A 3x3 convolution is applied on each merged feature map.
   a. This is to compensate for the aliasing effect caused by upsampling.
   b. Let us call the resulting feature maps as detection maps.

Detection

- Detection is applied over each detection map.
  - One can either use Faster-RCNN or SSD or any other detection approach for the final detection.