Object Identification

Jawwad Shamsi

March 22, 2024

1 Some Concepts

1. **Classification** Given an image identify the class it belongs to. e.g. dog or dog and cat
2. **Image Localization** location of cat or dog with in the image. location of a single object within the image
3. **Object Detection** To detect multiple objects with location, i.e. class plus location
4. **Image Segmentation** Divide an image into multiple parts and extract useful information. There will be some parts which do not contain any information. grouping pixels which contain similar information.
5. **Object detection vs Image Segmentation** The former has bounding box across each object. The box will be rectangular or square. The latter will have a pixel-wise mask for each object

2 Classification with Localization

Object Classification refers to identifying existence of an object (or various kinds of objects) within an image. Localization implies that where in an image the object is located.

2.1 Classification

We have seen Conv. nets are being used for object classification (dog vs cat).

Suppose we are developing object identification for the self driving car based system. Suppose that following objects can occur within an image :

1. Pedestrian (P)
2. Car (C)
3. Motorcycle (M)
4. Background (B)(if none of 1,2,and 3 is in the image, then we will assume it is a blank image)

2.2 Localization

. The location of the object within an image is identified through a **bounding box**. A bounding box has four parameters:

1. b_x: The X coordinate of the center of the box
2. b_y: The Y coordinate of the center of the box
3. b_w: The width of the box
4. b_h: The height of the box.

a bounding box specifies coordinates of an object with in an image. where 0,0 is the upper left corner and 1,1 is the lower right corner.

NN will output the above four items as well as the probabilities.

$$y = \begin{pmatrix} \mathrm{P}_c \\ \mathrm{b}_x \\ \mathrm{b}_y \\ \mathrm{b}_w \\ \mathrm{b}_h \\ \mathrm{C}_1 \\ \mathrm{C}_2 \\ \mathrm{C}_3 \end{pmatrix} \tag{1}$$

where P_c is the probability of the class. It is 1 for the object and 0 for background. and C_1, C_2, and C_3 are the probabilities of object, pedestrian, car, and motorcycle. Only one of them is present.

For instance, equation 2 shows occurrence of car

$$y = \begin{pmatrix} 1 \\ \mathrm{b}_x \\ \mathrm{b}_y \\ \mathrm{b}_w \\ \mathrm{b}_h \\ 0 \\ 1 \\ 0 \end{pmatrix} \tag{2}$$

whereas equation 3 shows don't care condition if the image only has background.

$$y = \begin{pmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{pmatrix} \tag{3}$$

2.3 Loss Function

$$l(\hat{y}, y) = (\hat{y} - y_1)^2 + (\hat{y} - y_2)^2 + ... + (\hat{y} - y_8)^2 (4)$$

where yˆis the estimated value of y.

y has 8 elements. The above equation holds if y_1 is 1, i.e. when there is an object

loss is sum of square over all the elements.

If y_1 is 0, then

$$l(\hat{y}, y) = (\hat{y} - y_1)^2 (5)$$

we can also use log likelihood loss for C_1, C_2 and C_3.

3 Localization and landmarks

A NN can also just output x and y coordinates of important points in an image.

sometimes called landmarks.

Example: For a face recognition system, we want to find where is the left corner of eyes in a face. For this purpose, we would like to output two points l_1x, l_1y

for two eyes and two points per eye, we can use four points l_1x, l_1y, l_2x, l_2y, l_3x, l_3y, and l_4x, l_4y

What if we would like to extract more landmarks, face, jaw position.

If we generate different landmarks to detect if a person is smiling or different other actions.

so the input for such a system is an image and it will process through a Convnet and output 129 variables.

face or not? and then l_1x, l_1y, ... $l_{64}x$, $l_{64}y$, It has applications for different systems such as snapchat.

For training we use a labeled dataset. We can also use action detection.

labels should be consistent for training.

4 Sliding Window Technique

The sliding window technique utilizes the concept of sliding a window throughout the image to detect an object. The underlying principle utilizes training a classifier to identify occurrence of an object within a cropped image. During the testing phase, a window is moved throughout an image to detect the object. Since the size and location of the object within an image is unknown, the window traverses throughout an image (with a stride) for detection. The technique involves several iterations of sliding. In each iteration, the size of the window and stride is kept fixed. Several iterations with increasing size of the window are made to detect the object. The technique has low performance when it is integrated with a NN based classifier. This is due to high computational cost. However, it can be used with simple machine learning based classifiers. Figure 1 illustrates the concept.

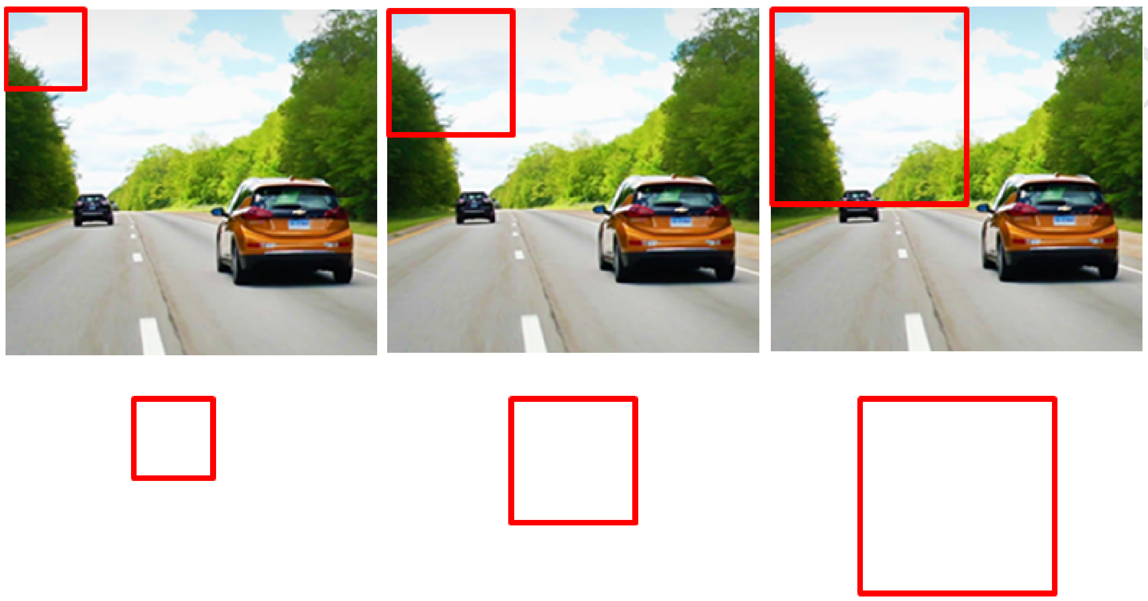

Figure 1: Object Detection through Sliding window

5 Incorporating Sliding window in Convolutional Neural Networks

The sliding window algorithm has been too slow. Let's implement this algorithm in conv.net.

5.1 Converting a fully connected layer to a conv.layer

Let us assume an input image of 14x14x3 with 16 filters of filter of 5x5. The resultant will be 16 feature maps 10x10. after applying max pooling of 2x2, the resultant image will be 5x5 (16 count). A fully connected layer will have 5x5x16=400 neurons, we can add another FC layer of 400 and then apply the softmax function to output y. For four classes (P,C,M,B) , we can have four neurons at the output. Figure **??**illustrates a fully connected network. Let's see how these layers can be drawn as conv. layers. figure **??** shows a corresponding convolutional neural network.

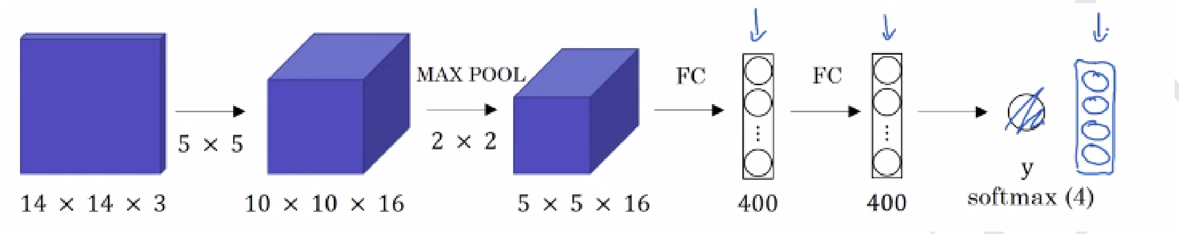

Figure 2: Fully Connected Network

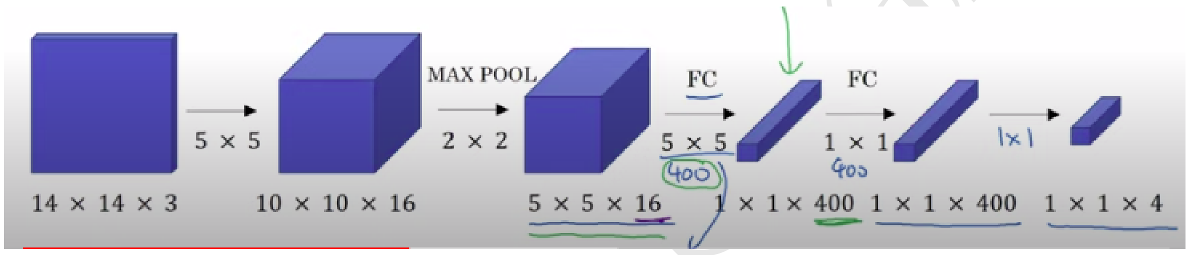

Figure 3: Fully Connected converted to to CNN

Recall: Sliding window has a massive overhead in identifying objects. (Why?). This overhead can be reduced with a slight modification in CNN.

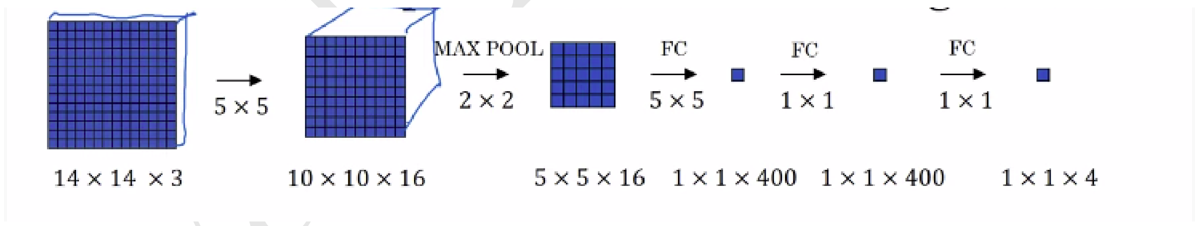

Figure 4: A Fully Connected Network

Figure 2 shows a fully connected network, whereas figure 3 shows a corresponding Conv. network. The figure is an extended image from the trained network in figure 2. We can see from the above figures, we turned the Fully Cconnected (FC) layer into a Conv layer using a convolution with the width and height of the filter (which is the same as the width and height of the input).

To understand the process, let us first understand how an FC system has been transformed. Figure 4 shows the transformation.

Now let us understand the sliding window with fully connected.

Assuming that we have a 16 x 16 x 3. We wold like to apply sliding window to this image. Through the convolution-based implementation, we would run this Conv. net four times each rectangle size will be 16 x 16.

Instead of applying a conventional sliding window technique, convolutional mechanism is applied to output a region. In that, each squared box in the output corresponds to a distinct o/p through sliding window. Figure 5 illustrates the process.

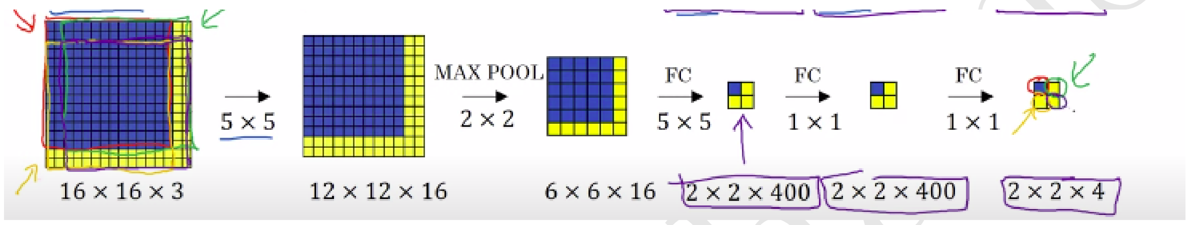

Figure 5: Sliding Window to a fully connected CNN