CMSC 491/691: Computer Vision **Spring 2024**

Lecture 10: 2024-03-06, Evaluating Object Detectors & Image Translation

Lecturer: Tejas Gokhale *Scribe: Andrew Smith*

10.1 Recap of Last Lecture

In the last lecture, we focused on the various ideas and developments behind the field of object detection with an emphasis on a few specific implementations including regression of bounding boxes, sliding windows, histograms of oriented gradients, and selective search. These methods had different strengths and weaknesses between each other but the common issues surrounding the field as a whole were how expensive an algorithm was and how well did it perform.

In lecture 10, the focus is on that second task of evaluating how well the networks performed (ability to detect objects "correctly"). The first step to understanding how well an object detector is performing is determining the measurable values that the detector will compute. After detecting an object, the algorithm will produce five values: an x-coordinate, y-coordinate, height, and width corresponding to the bounding box of the object, and a label to describe what that object is.

10.2 Bounding Box Evaluation

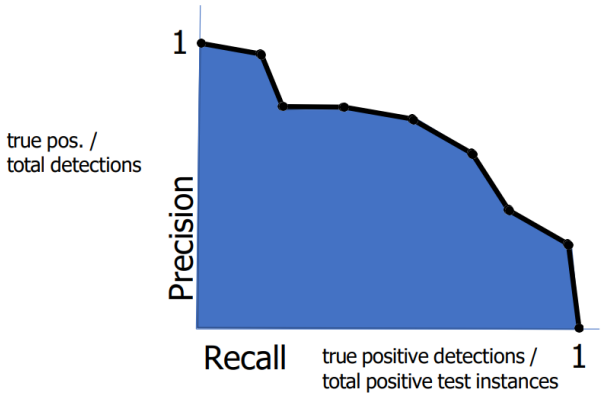
In this supervised learning scenario, evaluating the bounding box for an object is actually relatively simple. Given a ground-truth bounding box A and a predicted bounding box B for some object, it is determined to be a true or false positive by the method known as **Intersection over Union (IoU).**

$$\textbf{IoU}(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{10.1}$$

Otherwise known as Jaccard Similarity, **IoU** aims to maximize intersection between A and B while simultaneously minimizing union between A and B. In a perfect Scenario where A and B are the same, the intersection would be 100% between both bounding boxes and the union would be the same as just one of them.

10.3 Label Evaluation

In addition to the bounding boxes, the model must be trained and evaluated on its ability to predict the correct classes. This can be achieved by maximizing the average precision of the model. To compute the average precision, plot precision ($TP/TP + FP$) against recall ($TP/TP + FN$) and find the area under the curve created by those:



10.4 Non-Maximum Suppression

Developed to partially tackle the first issue mentioned in 10.1, 'expense', non-maximum suppression was developed as a greedy algorithm to discard irrelevant detections made by a model. Along with those five values from earlier, each detection made by the model can come with a confidence level as to how confident the algorithm is that it has detected an object. Non-maximum suppression is simply the act of iterating through all of the detections and discarding possible objects with low confidence levels.

Note 10.1 *Non-Maximum Suppression often uses a simple, greedy algorithm to determine which objects pass through and which are discarded.*

10.5 Image Transformation

Images transformation is the act of taking an image and in some way modifying it. There are different kinds of transformations like point operations, neighborhood operations, and warping. Each have different functions but generally point operations and neighborhood operations, also known as filtering, focus on changing pixel values, while warp operations focus on the orientation of the pixels. This can also be described as filters changing the range of an image and warping changing the domain.

10.6 Warping

One major challenges in the computer vision field of feature detection is the ability to detect the same features in different images, taken from different angles. This is better known as transformation invariance, the ability to detect the same features regardless of a transformation.

Warping is the broad category for all types of these 2D transformations that move, re-scale, and otherwise change pixel locations in an image. In the context of feature detection, determining how an image has been warped is the key to creating these 'transformation invariant' systems. To understand how to determine these warping functions let us first understand how they function:

Take for example the scale transformation, where the new (x', y') set is just a function of those variables multiplied with some scalars:

$$x' = ax \quad y' = by \tag{10.2}$$

This simple equation can be written in matrix form as follows:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{10.3}$$

In general, this form works very well for scale based transformations, but issues will arise using these "heterogeneous coordinates" if the transformations' begin to move unscaled.

10.7 Projective Geometry

The elegant solution to this issue, is projective geometry. Projective geometry adds an extra dimension to all of the calculations to allow for these different types of transformations. 2D matrices are now represented with a third dimension to represent "homogeneous coordinates".

$$\begin{bmatrix} x \\ y \end{bmatrix} => \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{10.4}$$

And now the scale transformation representation will look like like:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{10.5}$$

This results in the same information as before, just with a different representation. Now, if the transformation also needed to included a two pixel shift to the right we can finally represent it with our new dimension:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & 0 & \mathbf{2} \\ 0 & b & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{10.6}$$

Note 10.2 *As shown above, matrices can be composed with each other to form more complex transformations. The center matrix above is just the composition of the scale transformation* $\begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{bmatrix}$ *with the translation* $\begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

10.8 Classes of Transformations

Warp transformations can be further broken down and described by the number of "degrees of freedom" that they provide. Degrees of freedom are just the number of different parameters being changed within a transformation. As an example, the simple scale transformation we have been dealing with has two degrees of freedom (dof) since A and B effect the output (x', y'). Here is a short list of the broad classes for transformations:

Group	Matrix	Distortion	Invariant properties
Projective 8 dof	$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$		Concurrency, collinearity, **order of contact**: intersection (1 pt contact); tangency (2 pt contact); inflections (3 pt contact with line); tangent discontinuities and cusps. cross ratio (ratio of ratio of lengths).
Affine 6 dof	$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Parallelism, ratio of areas, ratio of lengths on collinear or parallel lines (e.g. midpoints), linear combinations of vectors (e.g. centroids). The line at infinity, $\mathbf{l}_\infty$.
Similarity 4 dof	$\begin{bmatrix} sr_{11} & sr_{12} & t_x \\ sr_{21} & sr_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Ratio of lengths, angle. The circular points, $\mathbf{I}, \mathbf{J}$ (see section 2.7.3).
Euclidean 3 dof	$\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Length, area

Take special note of Affine and Projective transformations. Affine transformations map lines to lines and maintain parallel properties between lines, effectively encapsulating all 2D transformations in one category. Projective on the other hand opens up those last 2 dof for a total of 8 dof. Transformations in this catagory may warp the images in a third dimension, akin to moving a camera to a diagonal angle for a new picture. This comes all the way back to our original issue of trying to determine the transformations between different images of the same setting taken from different angles.