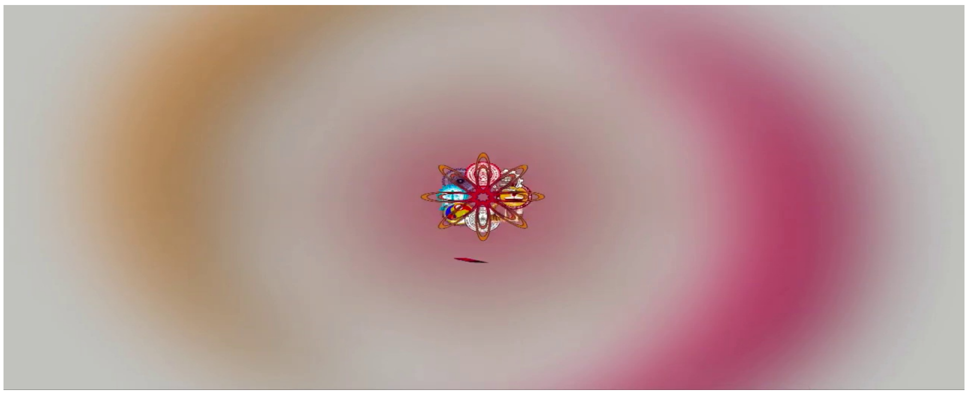
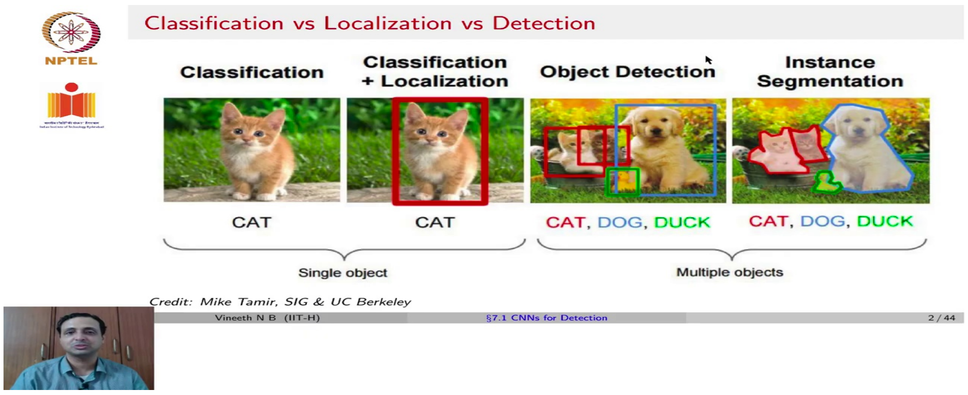


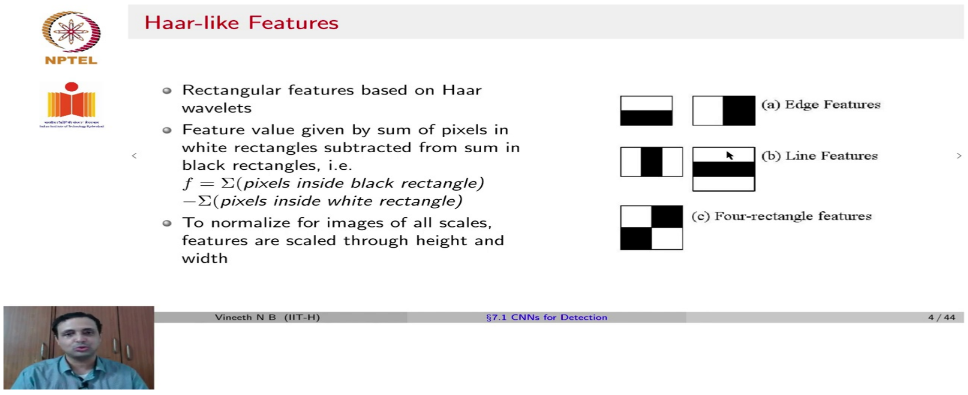
In object detection, you go one step further and you detect all possible occurrences of the objects in your set of classes in a given image as well as localize them.In the localization task, there is only one object which you localize whereas in object detection there could be multiple objects, multiple instances of the same object, you could have a cat and a dog or two dogs and one cat all of these possibilities.



Classification vs Localization vs Detection

Credit: Mike Tamir, SIG & UC Berkeley

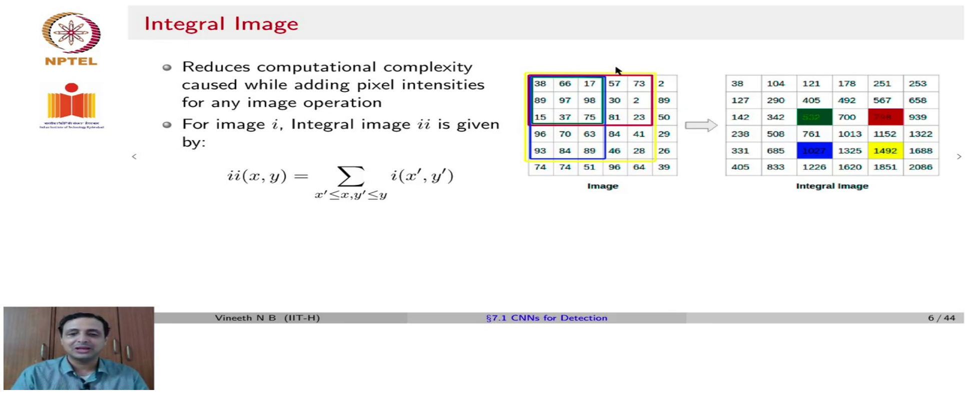
Finally, the task of segmentation where the job is to label each pixel as belonging to one of say c classes depending on the number of classes we have.So, you can see here that there are generally a region of black and a region of white but there could be different combinations of those you could have just two regions white and black horizontally aligned vertically arranged or you could have a black sandwiched between a white vertically or horizontally or you could have checkerboard patterns and you can keep going forward you will get various kinds of patterns as you use the same idea.So, the idea behind the haar filter is, the feature value is given by the sum of the pixels in the white rectangles subtracted from sum of the pixels in black rectangles.Remember this is convolution, so things would get flipped which is why you have the output of convolving with a Haar filter would be the sum of the pixels inside the black rectangle minus the sum of the pixels inside the white rectangle, remember black is 0 white is 1 but because of the flipping this is what the output would turn out to be.



Haar-like Features

- Rectangular features based on Haar wavelets
- Feature value given by sum of pixels in white rectangles subtracted from sum in black rectangles, i.e.
  $f = \Sigma(\textit{pixels inside black rectangle})$
  $-\Sigma(\textit{pixels inside white rectangle})$
- To normalize for images of all scales, features are scaled through height and width

So, let us see what an integral image is? So, the idea is to reduce computational complexity when you are adding pixel intensities for whatever option and the integral image is defined as, for any pixel in the integral image you sum up all the pixel values up till that point starting from the origin on the top left and the summation of all the pixel intensities is what is filled at a particular location.So mathematically, you could write it as, integral image at x comma y location would be the summation over x prime less than or equal to x where x is the current location y prime less than or equal to y, y being the current y location i of x prime y prime.



Integral Image

- Reduces computational complexity caused while adding pixel intensities for any image operation
- For image i, Integral image ii is given by:

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y')$$

Image

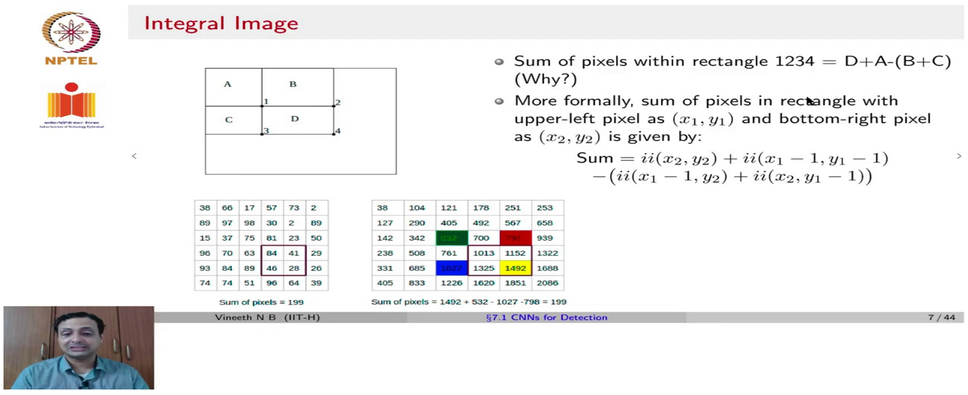
38	66	17	57	73	2
89	97	98	30	2	89
15	37	75	81	23	50
96	70	63	84	41	29
93	84	89	46	28	26
74	74	51	96	64	39

Integral Image

38	104	121	178	251	253
127	290	405	492	567	658
142	342	532	700	798	939
238	508	761	1013	1152	1322
331	685	1027	1325	1492	1688
405	833	1226	1620	1851	2086

Vineeth N B (IIT-H) §7.1 CNNs for Detection 6 / 44

And s of x, y is the cumulative row sum and to make this recurrent definition work you have to define s of x at minus 1 x comma minus 1 is equal to 0 and the integral image at minus 1, y is equal to 0.Let us assume now that we want to find the sum of the pixel intensities inside this rectangle defined by 1, 2, 3, 4 A is this top left rectangle B is this entire rectangle including a and including a part of it is A, C is this entire rectangle and D is this full rectangle.You can work this out and see that it would be quite true.



Integral Image

- Sum of pixels within rectangle 1234 = D+A-(B+C) (Why?)
- More formally, sum of pixels in rectangle with upper-left pixel as (x_1, y_1) and bottom-right pixel as (x_2, y_2) is given by:

$$\text{Sum} = ii(x_2, y_2) + ii(x_1 - 1, y_1 - 1) \\ - (ii(x_1 - 1, y_2) + ii(x_2, y_1 - 1))$$

38	66	17	57	73	2
89	97	98	30	2	89
15	37	75	81	23	50
96	70	63	84	41	29
93	84	89	46	28	26
74	74	51	96	64	39

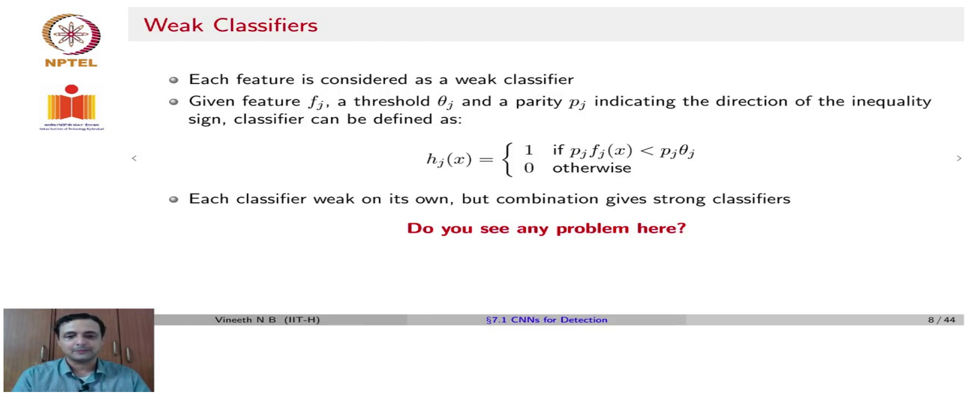
Sum of pixels = 199

38	104	121	178	251	253
127	290	405	492	567	658
142	342	532	700	798	939
238	508	761	1013	1152	1322
331	685	1027	1325	1492	1688
405	833	1226	1620	1851	2086

Sum of pixels = 1492 + 532 - 1027 -798 = 199

Vineeth N B (IIT-H) §7.1 CNNs for Detection 7 / 44

So, more formally you can say that sum of pixels inside a rectangle with an upper left pixel as x1, y1 and bottom right pixel as x2, y2 could be given by the integral image at x2, y2 plus the integral image at x 1 minus 1 y 1 minus 1 which is remember x 1, y 1 is the top left you will have to go 1 pixel before that minus integral image of x 1 minus 1 y 2 plus integral image of x 2, y 1 minus 1, so you can work this out to see that this is actually true.So, you would notice now that in this particular case, let us say you wanted to find out the sum of pixel intensities in this particular square block, so then what you could do is you could you already have the integral image computed, so you only have to look up a certain value certain set of values here and say 1492 plus 532 minus 798 minus 1827.Why is this useful? When you want to use a convolution such as a Haar filter for these kinds of images, if you had a 2 rectangle feature, what we mean by a 2 rectangle feature is let us assume that you have a Haar filter which is given by just a black region followed by a white region.Remember we said that this is the sum of pixel intensities in the black region minus the sum of pixel intensities in the white region, such a convolution for filter would only need 6 lookups in your integral image, why so? You would need one lookup for all of these corners, so there are six corners here including the middle ones you will need those 6 lookups and you will be able to compute what would be the output of the haar filter without actually doing any kind of a convolution, you can just look up the integral image and be able to get these values.



Weak Classifiers

- Each feature is considered as a weak classifier
- Given feature f_j, a threshold θ_j and a parity p_j indicating the direction of the inequality sign, classifier can be defined as:

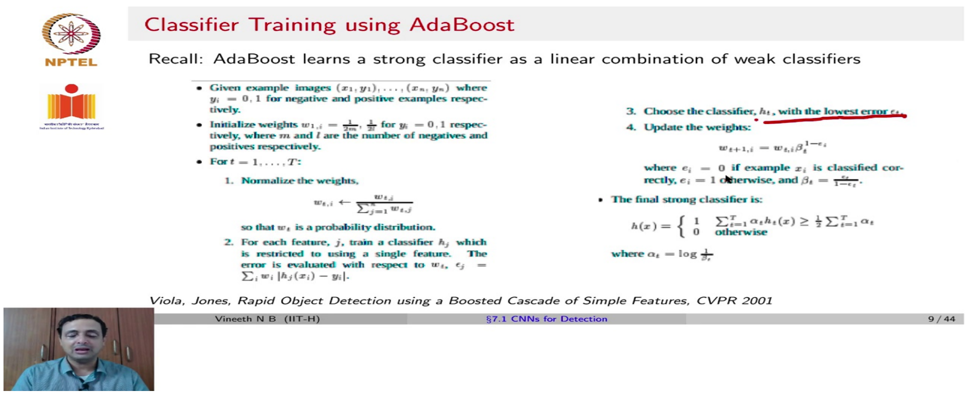
$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases}$$

- Each classifier weak on its own, but combination gives strong classifiers

Do you see any problem here?

Vineeth N B (IIT-H) §7.1 CNNs for Detection 8 / 44

Now, for each feature j which is one of your Haar features among the all possible combinations that you could have, you train a classifier j which is a hj which is a weak classifier which is restricted to using a single feature and the error with respect to this weak classifier is given by epsilon j which is given by wi into hj of xi minus yi, rather hj of xi is the output of this weak classifier, yi was the expected output, so how close this weak classifier was to the expected output multiplied by the weight of that sample that we are evaluating, that is the standard Adaboost approach to weight more erroneous samples a little higher and less erroneous samples a little lower.But now, the Viola Jones approach to using Adaboost says, among all those possible features which would have given you several different classifiers, choose the classifier ht which with the lowest error epsilon t is among all your possible classifiers choose the one that gives you the least error.



Classifier Training using AdaBoost

Recall: AdaBoost learns a strong classifier as a linear combination of weak classifiers

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:
  1. Normalize the weights,
  $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
  so that w_t is a probability distribution.
  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i \, |h_j(x_i) - y_i|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_t.
  4. Update the weights:
  $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$
  where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- The final strong classifier is:

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

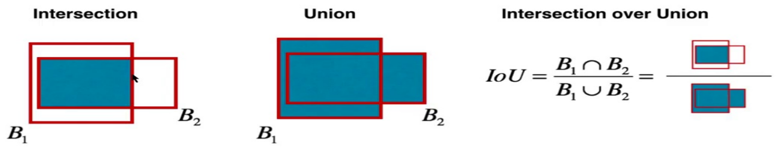
where $\alpha_t = \log \frac{1}{\beta_t}$

Viola, Jones, Rapid Object Detection using a Boosted Cascade of Simple Features, CVPR 2001

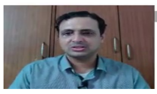
Vineeth N B (IIT-H) §7.1 CNNs for Detection 9 / 44

Based on the error of this classifier you re-weight all your samples as you see here, this is again the standard reweighting step in AdaBoost and you repeat this over and over again and your final strong classifier after a certain number of iterations, remember now you would not be using all features you are only using the T number of features where capital T is a parameter that you can give.If you went a little further and included 200 features, remember the above example we had only two kinds of Haar filters, if you included 200 kinds of Haar filters, when we say 200 kinds you arrange those black rectangles and white rectangles in several ways, you could have them as a checkerboard 2 cross 2 checker board a 5 cross 5 checker board you could have 3 rectangles you could have 5 rectangles 7 rectangles all those possibilities are open.So remember, that when you have a cascade the final detection rate and the false positive rate are found by multiplying the false positive rate and reduction rate of each of these stages.You could then achieve a false positive rate of 10 power minus 6 across a 10 stage cascade if each stage had a deduction rate of 0.99 and a false positive rate of 0.3.Because if you had a false positive rate of 0.3 in each stage of the cascade, you would then have 0.3 power 10 because you are talking about 10 stage cascade so 0.3 into 0.3 into 0.3 and so on, we will have 0.3 power 10 which is roughly about the order of 10 power minus 6.So the idea of classifier cascade allows us for each stage in the cascade to have a reasonable detection rate of 0.99 and a false positive rate of 0.3 which as we saw is possible with just a few features and we just keep refining these over stages of the cascade and obtain the false positive rate we ideally wanted.

Intersection over Union (IoU)

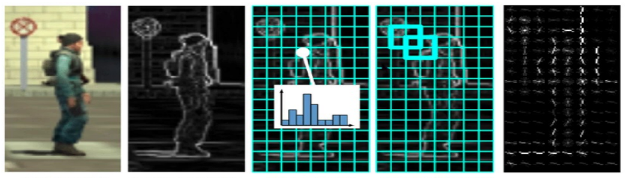


Credit: *Hands-On Convolutional Neural Networks with TensorFlow, Iffat Zafar*



When you do this for all the cells in your image, you would get a final descriptor which looks like this where you have a set of histograms of oriented gradients across different locations in the image.

Histograms of Oriented Gradients[2]



detection window slides over an image

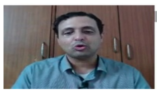
at each location where the window is applied, gradients are computed

window is evently partitioned into cells and each pixel of the cell contirbutes to cell gradient orientation histogram

orientation histograms for overlapping 2x2 blocks of cells are normalized and collected to form the final descriptor
This is called Contrast Normalization

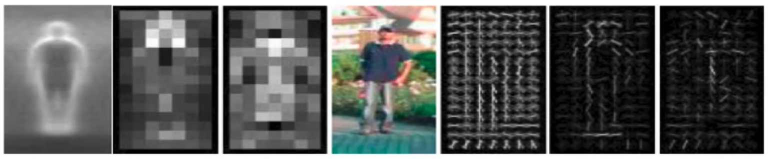
Final descriptor

[2]Dalal and Triggs, Histograms of Oriented Gradients for Human Detection, CVPR 2005



When you do this for all the cells in your image, you would get a final descriptor which looks like this where you have a set of histograms of oriented gradients across different locations in the image.

HOG: SVM Weights



Average gradient image over training examples

Maximum positive and negative SVM weights respectively centred on the block

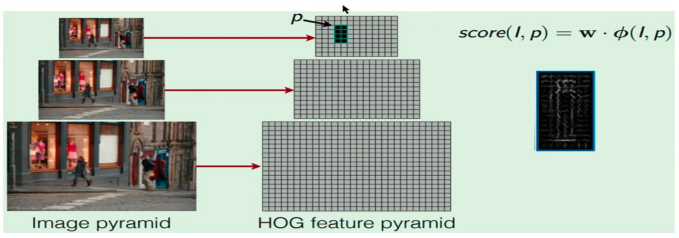
Test Image

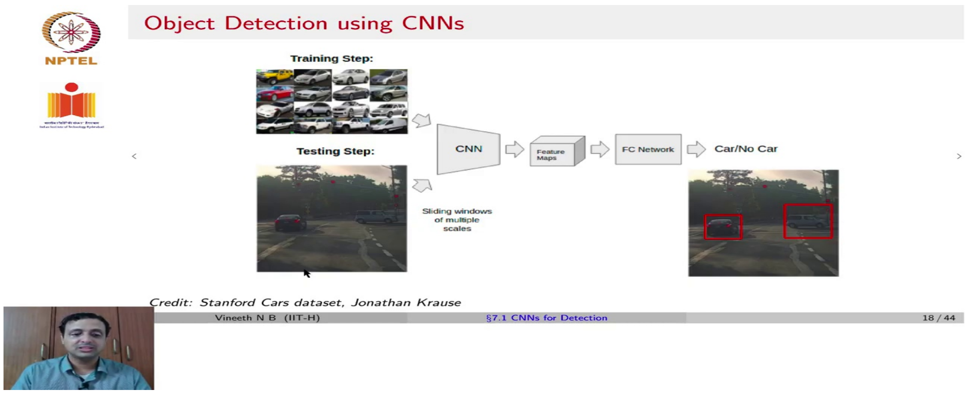
HOG descriptor for test image

HOG descriptor weighted by positive and negative SVM weights respectively

Image Pyramid

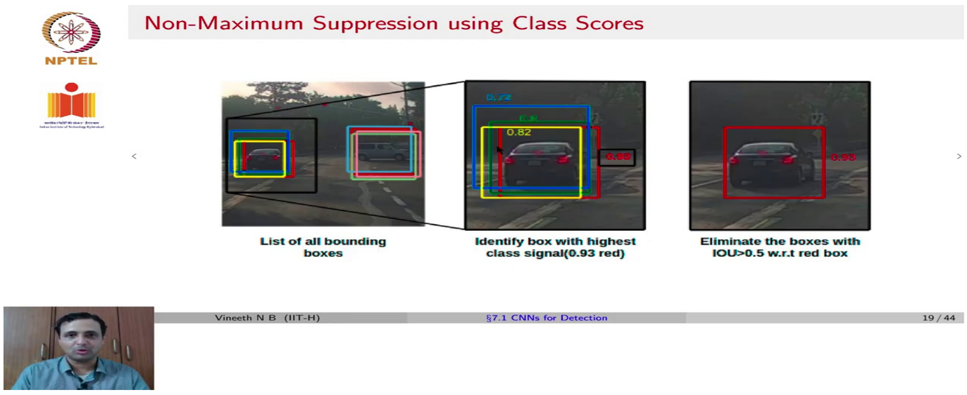
To identify images of all sizes, repeat the process across multiple image scales





Object Detection using CNNs

Credit: Stanford Cars dataset, Jonathan Krause



Non-Maximum Suppression using Class Scores

0.82
List of all bounding boxes
Identify box with highest class signal(0.93 red)
Eliminate the boxes with IOU>0.5 w.r.t red box

Now, you branch out your CNN into two parts, one part that gives you a classification score which can be learnt using a cross entropy loss, that is the standard CNN for classification that we have spoken about so far, but we could also do a bounding box regression.So, you could now try to see what are the exact coordinates of the bounding box say with respect to the entire image or with respect to any other box in the image.So, given an input which is say 14 cross 14 or you pad and make it 16 cross 16 you perform 5 cross 5 convolution, this is only an example for visualization, you get a 10 cross 10 output if you did not do padding and if you do 2 pixel padding it becomes 12 cross 12, then a 2 cross 2 pooling makes it 5 cross 5.