Lecture 3
Fourth stage

Medical Imaging Processing II

Histogram of Gradient Feature Extractor

By

Asst. Prof. Dr. Mehdi Ebady Manaa
Asst. Lect. Lubna ali jalil

Feature Extraction

Feature extraction is a game-changer in the world of machine learning algorithms. It's actually one of favorite aspects of being a data scientist! This is where we get to experiment the most – to engineer new features from existing ones and improve our model's performance. We have used various feature extraction techniques on structured data.

What is a Feature Descriptor?

let's clear that up first before we jump into the HOG part, see the two images shown below. Can you differentiate between the objects in the image?

It is clearly see that the right image here has a dog and the left image has a car. Now, let make this task slightly more complicated – identify the objects shown in the image below:

Still easy, right? Can you guess what was the difference between the first and the second case? The first pair of images had a lot of information, like the shape of the object, its color, the edges, background, etc. On the other hand, the second pair had much less information (only the shape and the edges) but it was still enough to differentiate the two images. We were easily able to differentiate the objects in the second case because it had the necessary information we would need to identify the object. And that is exactly what a feature descriptor does:

It is a simplified representation of the image that contains only the most important information about the image.

There are a number of feature descriptors out there. One of them is HOG: Histogram of Oriented Gradients

HOG Feature Descriptor

HOG, or Histogram of Oriented Gradients, is a feature descriptor that is often used to extract features from image data. It is widely used in computer vision tasks for object detection. An Important aspects of HOG that makes it different from other feature descriptors:

- The HOG descriptor focuses on the structure or the shape of an object. This is done by extracting the gradient and orientation (or you can say magnitude and direction) of the edges
- Additionally, these orientations are calculated in 'localized' portions. This means that the complete image is broken down into smaller regions and for each region, the gradients and orientation are calculated.
- Finally, the HOG would generate a Histogram for each of these regions separately. The histograms are created using the gradients and orientations of the pixel values, hence the name 'Histogram of Oriented Gradients'

To put a formal definition to this:

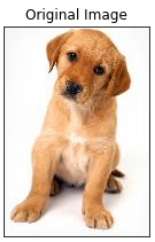
The HOG feature descriptor counts the occurrences of gradient orientation in localized portions of an image.

Calculating the Histogram of Oriented Gradients (HOG)

Consider the below image of size (180 x 280). Let us take a detailed look at how the HOG features will be created for this image:

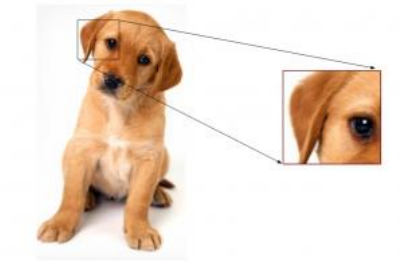
Step 1: Preprocess the Data (64 x 128)

Preprocess the image and bring down the width to height ratio to 1:2 for example. The image size should preferably be 64 x 128. This is because we will be dividing the image into 8*8 and 16*16 patches to extract the features.



Step 2: Calculating Gradients (direction x and y)

The next step is to calculate the gradient for every pixel in the image. **Gradients are the small change in the x and y directions.** We need take a small patch from the image and calculate the gradients on that:



We will get the pixel values for this patch. Let's say we generate the below pixel matrix for the given patch (as an example not real values)

121	10	78	96	125
48	152	68	125	111
145	78	85	89	65
154	214	56	200	66
214	87	45	102	45

I have highlighted the pixel value 85. Now, to determine the gradient (or change) in the x-direction, we need to subtract the value on the left from the pixel value on the right. Similarly, to calculate the gradient in the y-direction, we will subtract the pixel value below from the pixel value above the selected pixel.

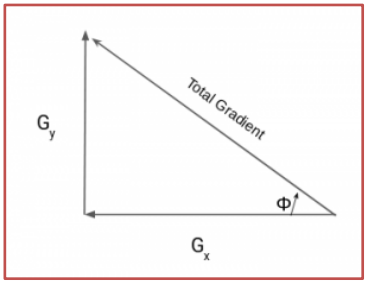
Hence the resultant gradients in the x and y direction for this pixel are:

- Change in X direction(G_x) = 89 – 78 = 11
- Change in Y direction(G_y) = 68 – 56 = 8

This process will give us two new matrices – one storing gradients in the x-direction and the other storing gradients in the y direction. This is similar to using kernel in the previous lecture. **The magnitude would be higher when there is a sharp change in intensity, such as around the edges.** the gradients in both x and y direction separately is calculated. The same process is repeated for all the pixels in the image. The next step would be to find the magnitude and orientation using these values.

Step 3: Calculate the Magnitude and Orientation

Using the gradients we calculated in the last step, we will now determine the **magnitude and direction** for each pixel value. For this step, we will be using the Pythagoras theorem Take a look at the image below:



The gradients are basically the base and perpendicular here. So, for the previous example, we had Gx and Gy as 11 and 8. Let's apply the Pythagoras theorem to calculate the **total gradient magnitude:**

- Total Gradient Magnitude = $\sqrt{[(Gx)2+(Gy)2]}$
- Total Gradient Magnitude = $\sqrt{[(11)^2+(8)^2]}$ = 13.6

Next, calculate the **orientation (or direction)** for the same pixel. We know that we can write the tan for the angles:

$$\tan(\Phi) = Gy / Gx$$

Hence, the value of the angle would be:

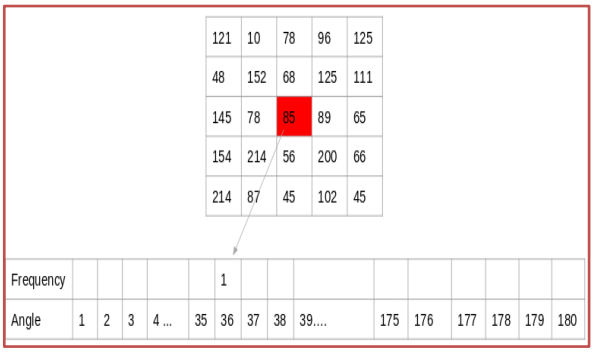
$$\Phi = \text{atan}(Gy / Gx)$$

The orientation comes out to be **36** when we plug in the values. So now, for every pixel value, we have the total gradient (magnitude) and the orientation (direction). **We need to generate the histogram using these gradients and orientations.**

Methods to Create Histograms using Gradients and Orientation

Method 1:

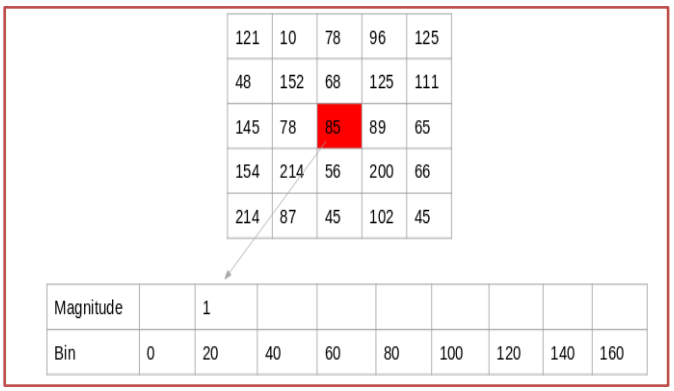
We will take each pixel value, find the orientation of the pixel and update the frequency table. Here is the process for the highlighted pixel (85). Since the orientation for this pixel is 36, we will add a number against angle value 36, denoting the frequency:



The same process is repeated for all the pixel values, and **we end up with a frequency table that denotes angles and the occurrence of these angles in the image.** This frequency table can be used to generate a histogram with angle values on the x-axis and the frequency on the y-axis. That's one way to create a histogram. Note that here the bin value of the histogram is 1. Hence we get about 180 different buckets, each representing an orientation value. Another method is to create the histogram features for higher bin values.

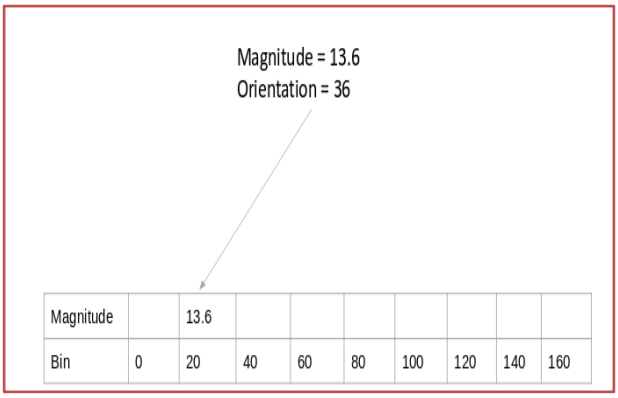
Method 2:

This method is similar to the previous method, except that here we have a bin size of 20. So, the number of buckets we would get here is 9. Again, for each pixel, we will check the orientation, and store the frequency of the orientation values in the form of a 9 x 1 matrix. Plotting this would give us the histogram:



Method 3:

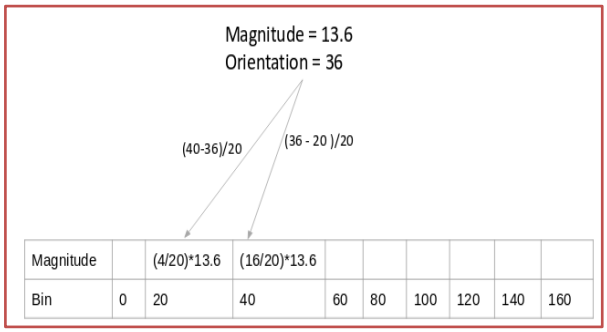
The above two methods use only the orientation values to generate histograms and do not take the gradient value into account. Here is another way in which we can generate the histogram – **instead of using the frequency, we can use the gradient magnitude to fill the values in the matrix.** Below is an example of this:



You might have noticed that we are using the orientation value of 36, and updating the bin 20 only. Additionally, we should give some weight to the other bin as well.

Method 4:

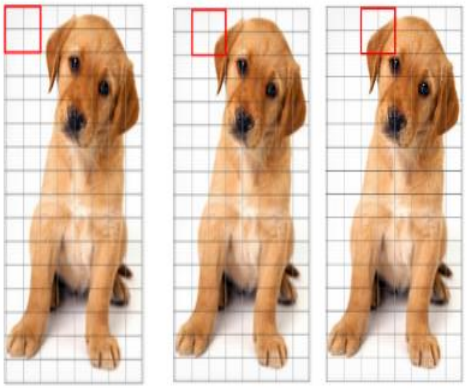
Here, we will add the contribution of a pixel's gradient to the bins on either side of the pixel gradient. Remember, the higher contribution should be to the bin value which is closer to the orientation.



Step 4: Calculate Histogram of Gradients in 8×8 cells (9×1)

The histograms created in the HOG feature descriptor are not generated for the whole image. Instead, the image is divided into 8×8 cells, and the histogram of oriented gradients is computed for each cell. Why do you think this happens? By doing so, we get the features (or histogram) for the smaller patches which in turn represent the whole image. We can certainly change this value here from 8 x 8 to 16 x 16 or 32 x 32. If we divide the image into 8×8 cells and generate the histograms, we will get a 9 x 1 matrix for each cell. This matrix is generated using method 4 that we discussed in the previous section. Once we have generated the HOG for the 8×8 patches in the image, the next step is to normalize the histogram.

Step 5: Normalize gradients in 16×16 cell (36×1)



Before we understand how this is done, it's important to understand why this is done in the first place. Although we already have the HOG features created for the 8×8 cells of the image, the gradients of the image are sensitive to the overall lighting. This means that for a particular picture, some portion of the image would be very bright as compared to the other portions. We cannot completely eliminate this from the image. But we can reduce this lighting variation by normalizing the gradients by taking 16×16 blocks. Here is an example that can explain how 16×16 blocks are created: Here, we will be combining four 8×8 cells to create a 16×16 block. And we already know that each 8×8 cell has a 9×1 matrix for a histogram. So, we would have four 9×1 matrices or a single 36×1 matrix. To normalize this matrix, we will divide each of these values by the square root of the sum of squares of the values. Mathematically, for a given vector V:

$$V = [a1, a2, a3, \ldots a36]$$

We calculate the root of the sum of squares:

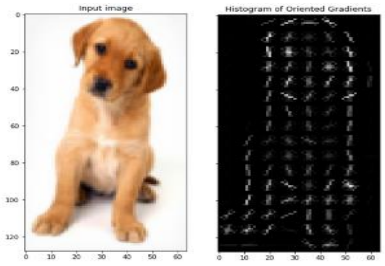
$$k = \sqrt{(a1)^2 + (a2)^2 + (a3)^2 + \ldots (a36)^2}$$

And divide all the values in the vector V with this value k:

$$\text{Normalised Vector} = \left[\frac{a_1}{k}, \frac{a_2}{k}, \frac{a_3}{k}, \ldots \frac{a_{36}}{k}\right]$$

The resultant would be a normalized vector of size 36×1.

Step 6: Features for the complete image



We are now at the final step of generating HOG features for the image. So far, we have created features for 16×16 blocks of the image. Now, we will combine all these to get the features for the final image. Can you guess what would be the total number of features that we will have for the given image? We would first need to find out how many such 16×16 blocks would we get for a single 64×128 image:

We would have 105 (7×15) blocks of 16×16. Each of these 105 blocks has a vector of 36×1 as features. Hence, the total features for the image would be 105 x 36×1 = 3780 features.