Wavelets Transform

Dr. Su Su Maung
CEIT Department
Yangon Technological University

Outline of Lecture10

- Introduction
- A Simple Wavelet Transform
- Two-Dimensional waveletes
- Wavelets and Images
- The Daubechies wavelets
- Image compression using wavelets
- High pass filtering using wavelets
- Denoising using wavelets

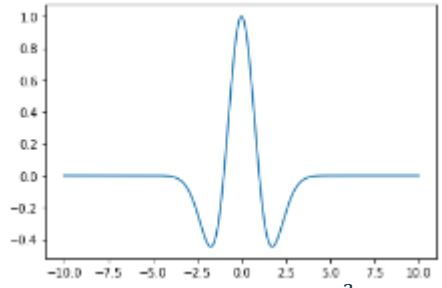
Introduction

- The idea of wavelets is to keep the wave idea, but drop the periodicity. So we may consider a *wavelet* to be a little part of a wave: a wave which is only non-zero in a small region.
- In the more general form $y = \frac{2}{\sqrt{3\sigma}\pi^{1/4}}(1 - \frac{x^2}{\sigma^2})e^{-x^2/(2\sigma^2)}$ is known as the *Mexican hat wavelet*.
- Suppose we are given a wavelet, we can:
  - **Dilate it** by applying a scaling factor to *x*: *f*(2*x*) would "squash" the wavelet; *f*(*x*/2) would expand it.
  - **Translate it** by adding or subtracting an appropriate value from x: f(x – 2) would shift the wavelet 2 to the right; f(x + 3) would shift the wavelet 3 to the left.
  - **Change its height** by simply multiplying the function by a constant.
- Given a suitable starting wavelet *w*(*x*), we can express any function *f*(*x*) as a sum of wavelets of the form. $aw(bx + c)$

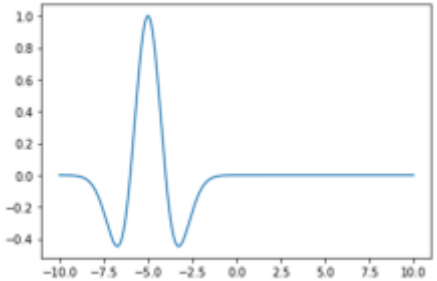
Introduction (cont.)

- Moving into two dimensions, we can apply wavelets to images.
- Using wavelets has provided a new class of very powerful image processing algorithms: wavelets can be used for noise reduction, edge detection, and compression.
- The use of wavelets has superseded the use of the DCT for image compression in the JPEG2000 image compression algorithm.

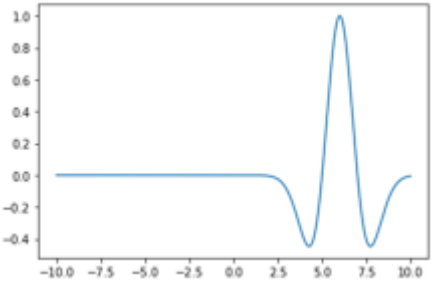
Introduction (cont.)



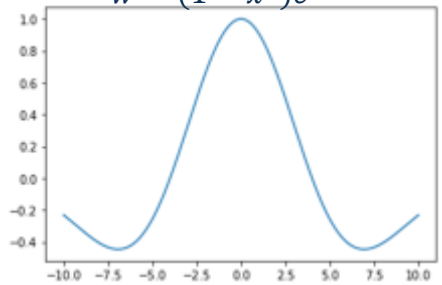
$w = (1 - x^2)e^{-x^2/2}$



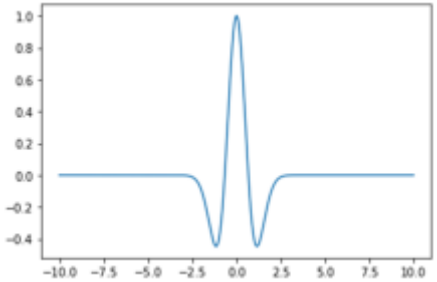
$w(x + 5)$



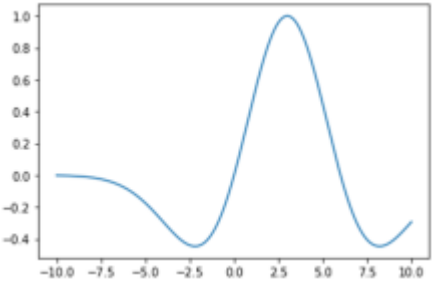
$w(x - 6)$



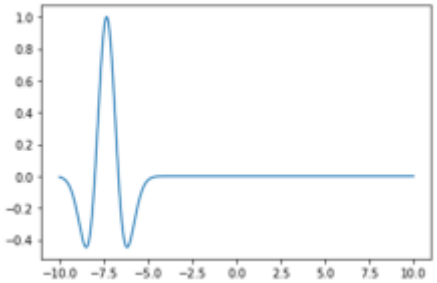
$w(x/4)$



$w(3x/2)$



$w(x/3 - 1)$



$w(1.5x + 11)$

Shifts and dilation of the wavelet w(x)

A Simple Wavelet Transform

- All wavelet transforms work by taking weighted averages of input values and providing any other necessary information to be able to recover the original input.
- we shall perform just two operations: adding two values and subtracting. For example, suppose we are given two numbers 14 and 22. Their sum and difference are

$$14 + 22 = 36$$
$$14 - 22 = -8$$

- To recover the original two values from the sum and difference, we take the sum and difference again, and divide by two:

$$36 - (-8))/2 = 22$$
$$36 + (-8))/2 = 14$$

A Simple Wavelet Transform (cont.)

- The forward operations (adding, subtracting) and the backward operations (adding and dividing by two, subtracting and dividing by two) differ only in the extra division. This function can be extended to any vector with an even number of elements. For example, suppose

$$v = [71, 67, 24, 26, 36, 32, 14, 18]$$

- Then define *s* to be the sum and d to be the differences of the elements in pairs:

$$\begin{aligned} s &= [71 + 67, \quad 24 + 26, \quad 36 + 32, \quad 14 + 18] \\ &= [138, \ 50, \ 68, \ 32] \end{aligned}$$

$$\begin{aligned} d &= [71 - 67, \quad 24 - 26, \quad 36 - 32, \quad 14 - 18] \\ &= [4, \quad -2, \quad 4, \quad -4] \end{aligned}$$

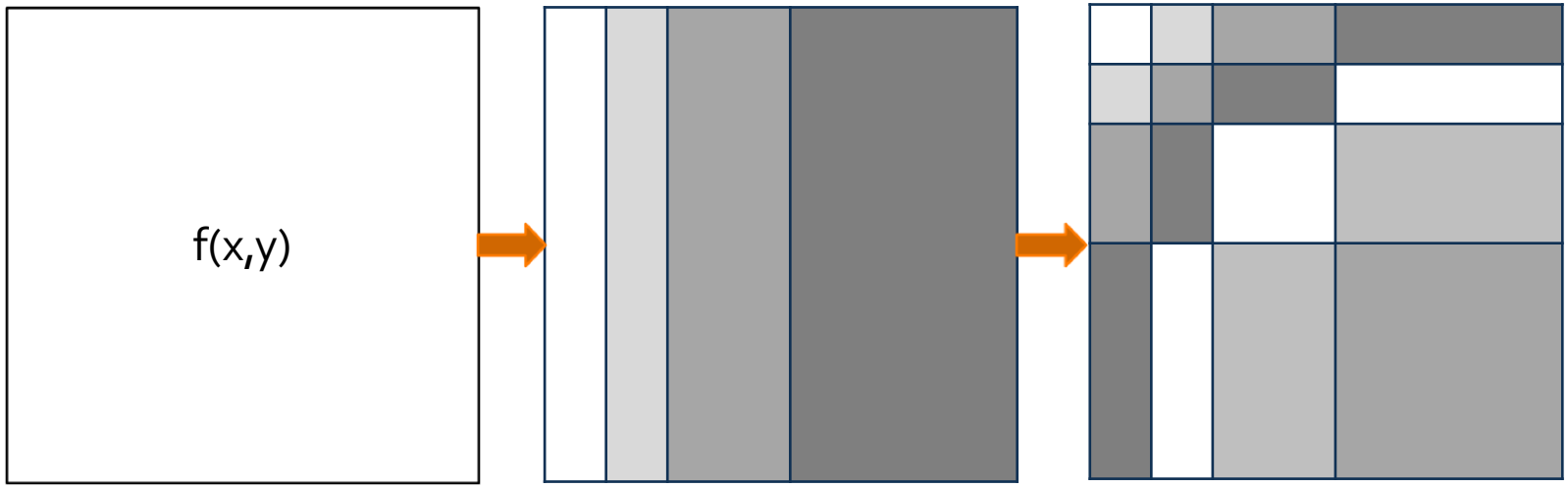
- The concatenation of these two vectors is the *discrete wavelet transform at 1 scale* of the original vector.

$$w = [138\,, 50\,, 68\,, 32\,, 4\,, -2\,, 4\,, -4]$$

Standard decomposition

- The two-dimensional wavelet transform is separable, which means we can apply a one-dimensional wavelet transform to an image in the same way as for the DFT.
- We apply a one-dimensional DWT to all the columns, and then one-dimensional DWTs to all the rows of the result. This is called the standard decomposition.

Standard decomposition (cont.)

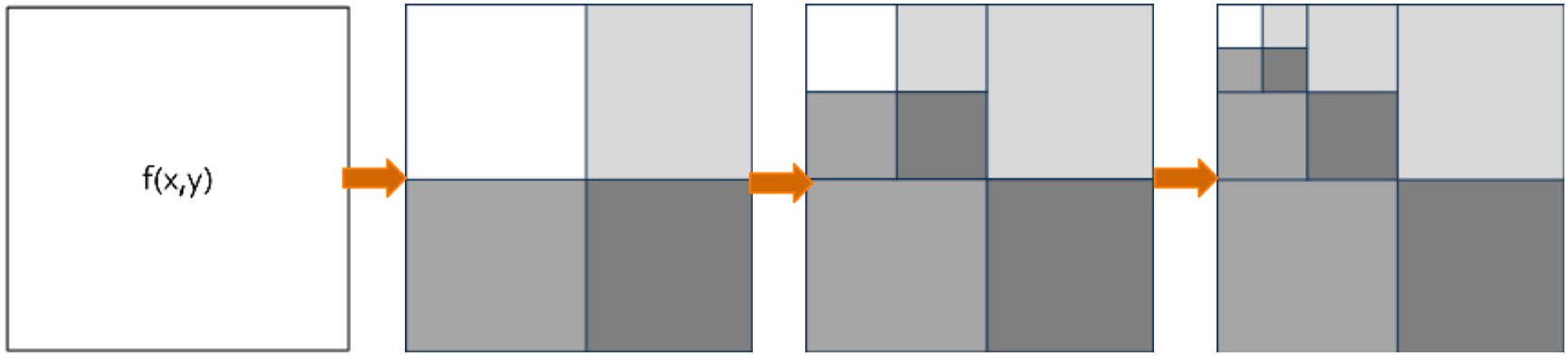


The standard decomposition of the two dimensional DWT

Nonstandard decomposition

- We can also apply a wavelet transform differently. Suppose we apply a wavelet transform (say, using the Haar wavelet) to an image by columns and then rows, but using our transform at 1 scale only.
- This will produce a result in four quarters: the top left will be a half-sized version of the image; the other quarters will be high pass filtered images. These quarters will contain horizontal, vertical, and diagonal edges of the image.
- We then apply a 1-scale DWT to the top left quarter, creating smaller images, and so on. This is called the *nonstandard decomposition*.

Nonstandard decomposition (cont.)



Original image | 1- scale DWT | 2- scale DWT | 3 - scale DWT

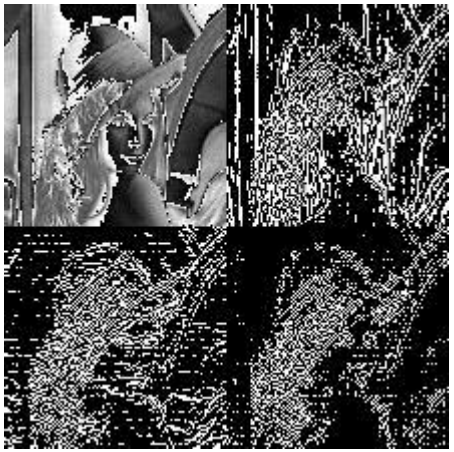
The nonstandard decomposition of the two dimensional DWT

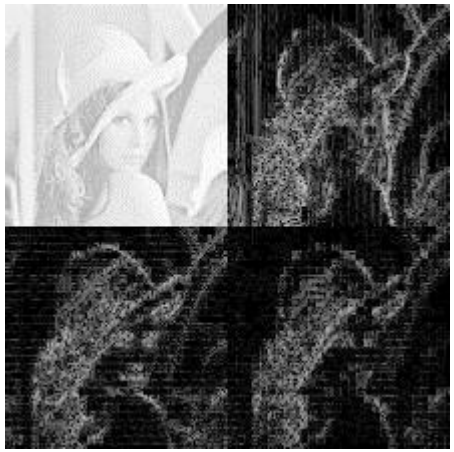
Wavelets and Images

- Python has some wavelet functionality, either built in or available through an added toolbox or package.
- However, the standard packages all seem to work a little differently.
- Let's try an image. We shall apply the Db2 wavelet to an image of size 256 × 256, first at one scale. For display, it will be necessary to scale parts of the image for viewing. Wavelet images can be displayed simply with io.imshow.

Wavelets and Images (cont.)

```
import pywt
import cv2
img = cv2.imread('lena.jpg')
Img=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
coeffs = pywt.wavedecn(img, wavelet='db2', level=2)
arr, coeff_slices = pywt.coeffs_to_array(coeffs)
io.imshow(arr)
coeffs =pywt.array_to_coeffs(arr)
img = pywt.waverec(coeffs, wavelet='db2')
io.imshow(img)
```

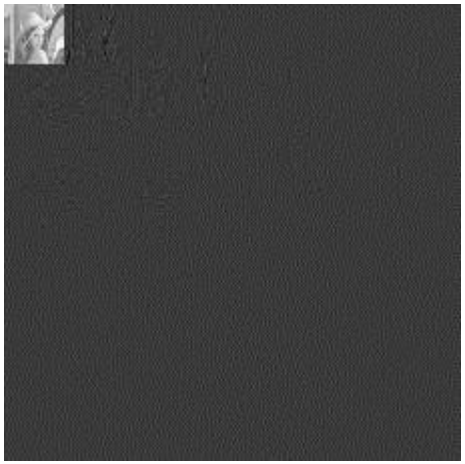


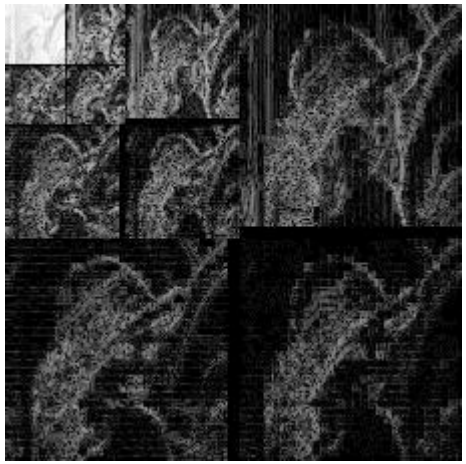


Different displays of a 1-scale DWT applied to an image

Wavelets and Images (cont.)

```
import pywt
import cv2
img = cv2.imread('lena.jpg')
Img=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
coeffs = pywt.wavedecn(img, wavelet='db2', level=3)
arr, coeff_slices = pywt.coeffs_to_array(coeffs)
io.imshow(arr)
coeffs =pywt.array_to_coeffs(arr)
img = pywt.waverec(coeffs, wavelet='db2')
io.imshow(img)
```



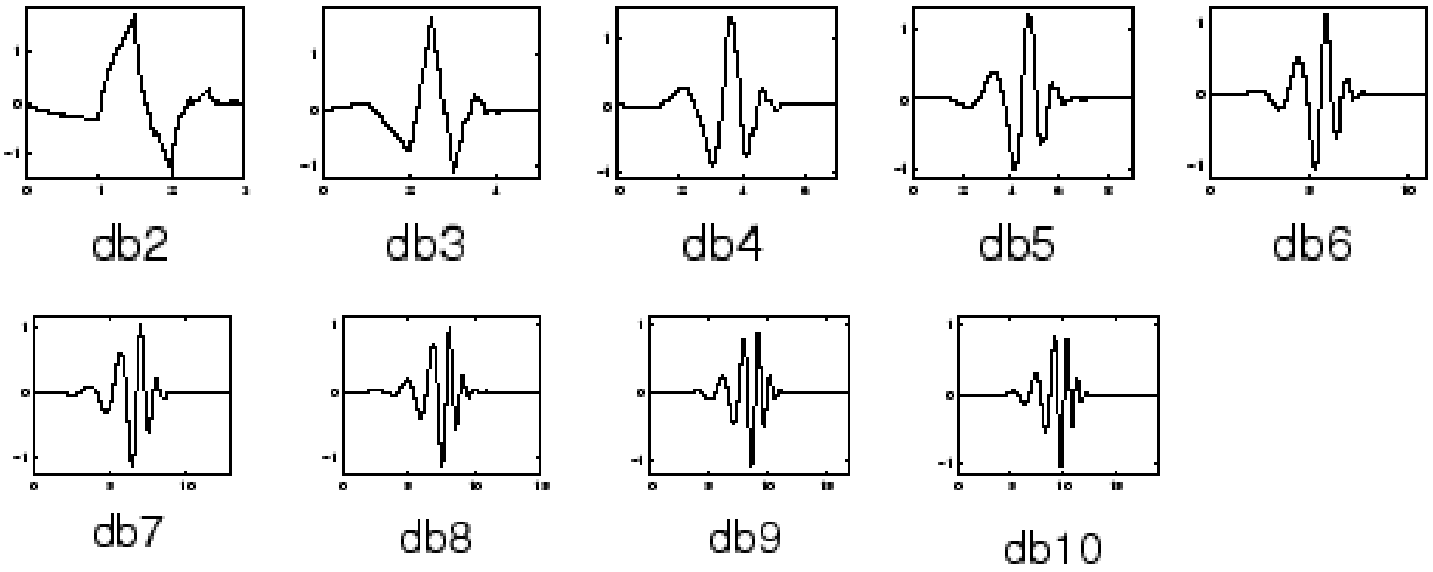


Different displays of a 2-scale DWT applied to an image

The Daubechies wavelets

- The names of the Daubechies family wavelets are written dbN, where N is the order, and db the "surname" of the wavelet.
- The db1 wavelet, as mentioned above, is the same as Haar wavelet.
- Here are the wavelet functions psi of the next nine members of the family:

The Daubechies wavelets (cont.)



the wavelet functions of the nine members of the Daubechies family

Image compression using wavelets

- Wavelets have replaced the use of the DCT in the JPEG2000 algorithm for image compression.
- The idea is to take the DWT of an image, and for a given value d, set all values x in the DWT for which $|x| \leq d$ to zero. This is at the basis of the JPEG2000 compression .
- We shall try this using the Daubechies-4 wavelets, and $d = 10$ removing nearly three quarters of the information from the DWT.
- We could obtain higher compression rates by removing more information by increasing the threshold value d. We will try with d = 30 and d = 50, and using the Daubechies-4 wavelet.
- With a threshold value of 50 we are throwing away the result is extremely good when compared with the results of JPEG compression.

Image compression using wavelets (cont.)

```
import cv2
import pywt
import skimage.util.noise as noise
img = cv2.imread('lena.png')
img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
coeffs = pywt.wavedecn(img, wavelet='db4', level=8)
arr, coeff_slices = pywt.coeffs_to_array(coeffs)
io.imshow(arr)
L=len(np.where(abs(arr)<=50)[0])
arr[np.where(abs(arr)<=50)]=0
arr = np.round(arr)
coeffs_from_arr = pywt.array_to_coeffs(arr, coeff_slices,output_format='wavedecn')
img1_recon = pywt.waverecn(coeffs_from_arr, wavelet='db4')
io.imshow(img1_recon)
io.imsave('50compressed_img.jpg',img1_recon)
```

Image compression using wavelets (cont.)

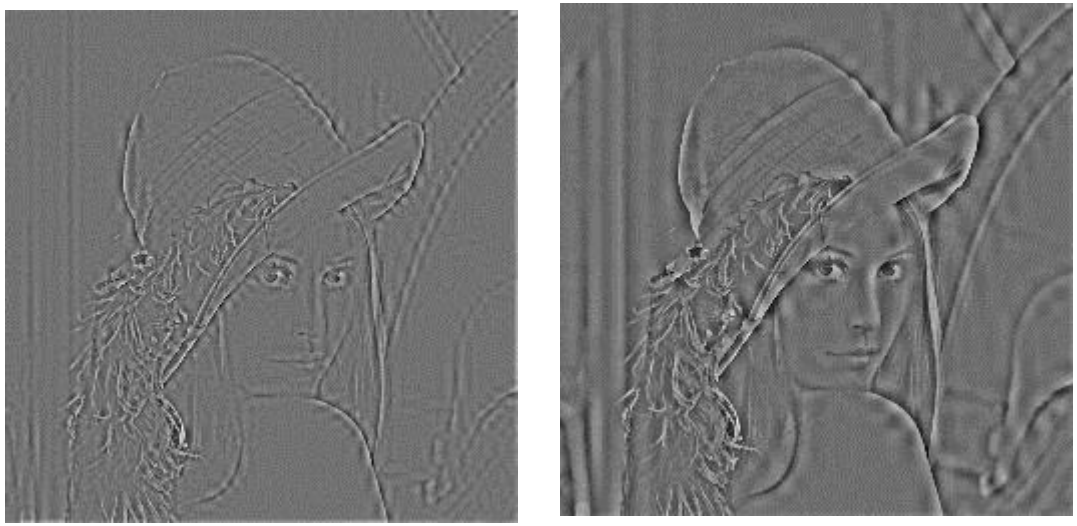
Results of wavelet compression with d = 10, d = 30 and d = 50

High pass filtering using wavelets

- Apart from the rescaled image in the top left of a wavelet transform(low frequency information), the rest is high frequency information.
- If we eliminate the top left corner, by setting all the transform values to zero, the result after inversion would be a high pass filtered version of the original image.
- We will perform a 2-scale and 3-sale decomposition (with Daubechies-4) on the image, and remove the image from the top left. These are shown in figure and the images can be thresholded to produce edge binary images.

High pass filtering using wavelets (cont.)

```
import cv2
import pywt
img = cv2.imread('lena.png')
coeffs = pywt.wavedecn(img, wavelet='db4', level=3)
arr, coeff_slices = pywt.coeffs_to_array(coeffs)
arr[0:32,0:32]=0
coeffs_from_arr = pywt.array_to_coeffs(arr,
coeff_slices,output_format='wavedecn')
img1_recon = pywt.waverecn(coeffs_from_arr, wavelet='db4')
img1_recon[np.where(abs(img1_recon)<=0.08)]=0
```



High pass filtering with wavelets (2-scale and 3-scale decomposition)

Denoising using wavelets

- Since noise is a high frequency component, some form of low pass filtering can be used to remove it.
- We can remove such components by thresholding.
- We shall assume that all the filter coefficients are those of the Daubechies-4 wavelet.

Denoising using wavelets (cont.)

```
import cv2
import pywt
import skimage.util.noise as noise
img = cv2.imread('images/lena.jpg')
img =
cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
img1=noise.random_noise(img,'gaussian')
coeffs = pywt.wavedecn(img1, wavelet='db4',
level=2)
arr, coeff_slices =
pywt.coeffs_to_array(coeffs)
arr[np.where(abs(arr<=0.5))]=0
coeffs_from_arr = pywt.array_to_coeffs(arr,
coeff_slices,output_format='wavedecn')
img1_recon =
pywt.waverecn(coeffs_from_arr,
wavelet='db4')
```

noisy image

threshold of 0.4

threshold of 0.3

threshold of 0.5

The noisy image and the result after wavelet filtering

Next Week Lecture (Week11)

- Lecture11:Extracting image features

Thank You