The goal of this assignment is to develop a face recognition machine. It is based on the recent work of Bruna and Mallat, 2012. The idea is to build a convolution-like Network, with various layers (typically 2 or 3 layers). At each layer an “image-like” is convolved with a low pass filter $\phi_\sigma$ (a blur-like filter at scale $\sigma$) and down sampled (reducing the size). We refer to “image-like” to structures that have the same pixels as the image and are non-negative quantities at each pixel. The “image-like” vary from layer to layer via a prescribed procedure as we describe next.

At the zero layer, we start the input image, $I$, i.e., $I^0=I$. At the next layer, layer-1, the image $I$ is convolved with a bank of $m$ wavelets at different directions and different scales, represented by $\psi_{\lambda_i}$, where $\lambda_i=(\sigma_i,\theta_i)$ represents a given scale and orientation, i.e., $i=1,\ldots,m$. The “images-like” at this layer becomes the magnitude of each convolution, $I^1_i=|\psi_{\lambda_i}*I^0|$. We have as many “images-like” at this layer as orientations and scales of wavelets. At the next layer, layer-2, we have $I^2_{i,j}=|\psi_{\lambda_i}*I^1_i|$, $k=(i,j)$. Again, the saved vector is the down sampled of: each “image-like” convolved with a low pass filter, i.e., the down sample of $v^2_k=\phi_\sigma*I^2_k$. The final representation, what is saved, is a set of layer vectors $v^L_k$ downsampled. This representation is created to be invariant to small deformation made to each input image. So if two images are similar, up to small deformations on the grid, then the final output of the scattered network will produce vectors that are close to each other in Euclidean distance.

Let us develop and test these ideas on face recognition.

**MORLET WAVELETS:** YOU HAVE DONE ALL OF THIS ALREADY, EXCEPT, I AM SUGGESTING TO USE FEWER ANGLES, ONLY 4, NOT 8, TO REDUCE COMPUTATIONS. SEE BELOW

**ZERO LAYER**

Convolve the image with a Gaussian Blur ($\phi_\sigma=G_{\sigma=6}$)

$$S^0(I) = G_{\sigma=6} * I(u)$$

We then downsample the output by a factor $2^3$ along the $x$ and $y$ axis. Let us refer the downsample by $2^k$ as $S^0_{k=3}(I)$. If the image $I(u)$ is of size $32 \times 32$, then $S^0_{k=3}(I)$ is of size $4 \times 4$, i.e., it contains 16 parameters (the values of $S^0_{k=3}(I)$ at each of the $4 \times 4$ locations). One can refer to the elements $[S^0_{k=3}(I)]_{pq}$, $p,q = 1,\ldots,4$. 
**FIRST LAYER**

For each image and each $\lambda_i$, perform a convolution with the real and imaginary part of the wavelet and produce the results,

$$W_{\lambda_i}I(u) = \psi_{\lambda_i} * I(u) \quad i = 1, \ldots, 12$$

or

$$W_{\lambda_i}^{\text{Real}}I(u) = \psi_{\lambda_i}^{\text{Real}} * I(u) = \sum_{x'} \sum_{y'} I(x', y') \psi_{\lambda_i}^{\text{Real}}(x' - x, y' - y) \quad i = 1, \ldots, 12$$

$$W_{\lambda_i}^{\text{Im}}I(u) = \psi_{\lambda_i}^{\text{Im}} * I(u) = \sum_{x'} \sum_{y'} I(x', y') \psi_{\lambda_i}^{\text{Im}}(x' - x, y' - y) \quad i = 1, \ldots, 12$$

Compute the magnitude of the wavelet transform, $|W_{\lambda_i}I(u)|$, where

$$|W_{\lambda_i}I(u)| = \sqrt{W_{\lambda_i}I(u) \cdot \text{conjugate}(W_{\lambda_i}I(u))} = \sqrt{\left(W_{\lambda_i}^{\text{Real}}I(u)\right)^2 + \left(W_{\lambda_i}^{\text{Im}}I(u)\right)^2}$$

and in order to down sample it, first convolve it with the Gaussian $G_{\sigma=6}$

$$S_{\lambda_i}^1(I) = G_{\sigma=6} * |W_{\lambda_i}I(u)|$$

Finally, we also down sample the output by a factor $2^3$ along the x and y axis, to make $S_{\lambda_i,k=3}^1(I)$. For the image $I$ of size $32 \times 32$, $S_{\lambda_i,k=3}^1(I)$ is of size $4 \times 4$ for each of the twelve parameters $\lambda_i$. $S_{\lambda_i,k=3}^1(I)$ can be thought as a vector of size $4 \times 4 = 16$ where each entry value is the quantity $S_{\lambda_i,k=3}^1(I)$ at the corresponding location in the $4 \times 4$ grid, i.e.,

$$[S_{\lambda_i}^1(I)]_{p,q} \quad p,q = 1, \ldots, 4 \quad \text{or} \quad (S_{\lambda_i}^1(I))_l \quad l = 1, \ldots, 16$$

We will be using the vector notation $(S_{\lambda_i}^1(I))_l$. All together, we have 12 vector image-like structures $\{S_{\lambda_i,k=3}^1(I), i = 1, \ldots, 12\}$ in layer 1, each of size 16.

**PROBLEM 1: PROCESS THE DATABASE BY THE SCATTERED WAVELET NETWORK**

Let us work first with just these two layers, $\tilde{S}(I) = \{S^0(I), S_{\lambda_i}^1(I), i = 1, \ldots, 12\}$, where $S^0(I)$ is of size 16 and the 12 scattered $S_{\lambda_i}^1(I)$ are also each of size 16 One can think of $\tilde{S}(I)$ as 13 vectors of size 16 with entries given by the scattered network described above.
Alternatively, we can write \( \vec{s}(I) \) as a long vector of size \( 208 = (1 + 12) \times 16 \), a concatenation of the vectors \( \{S_{1t}(I)\}_t \) into one long vector. This is our new image representation. Every image of a face on the database becomes a vector \( \vec{s}(I) \) of size \( 208 \). Compute them for the image database of faces.

**Problem 2: Face Training and Recognition**

a. **PCA to improve and simplify representation**

We could in “first approximation” represent a class of all \( N \) images of faces by their average value, i.e.,

\[
\bar{A}S_{F} = \frac{1}{N} \sum_{k=1}^{N} \vec{s}(I_k)
\]

Then, the covariance associated to the \( N \) image face vectors \( \vec{s}(I_k) \) is defined as the matrix

\[
CovS_{mn} = \frac{1}{N - 1} \sum_{k=1}^{N} (S_m(I_k) - \bar{A}S_m)(S_n(I_k) - \bar{A}S_n)
\]

Where \( m, n = 1, \ldots, 208 \), and \( \bar{A}S_m \) is the \( m \)-th component of \( \bar{A}S \). Likewise, \( S_m(I_k) \) is the \( m \)-th component of \( \vec{s}(I_k) \). From the covariance matrix we can compute the principal components as the normalized eigenvectors of it, say \( \{e_j; j = 1, \ldots, m\} \) are all the normalized eigenvectors of \( CovS \) and \( \{\alpha_j; j = 1, \ldots, m\} \) the corresponding eigenvalues, i.e.,

\[
CovS e_j = \alpha_j e_j
\]

For the covariance matrix the eigenvectors are also orthogonal. The eigenvalues capture the variance of the data along the eigenvectors. We consider the eigenvectors associated to the higher eigenvalues, since if the variance is very small we can neglect these directions. Say we choose the top \( L \) eigenvectors, normalized, i.e., we keep \( (e_1, \ldots, e_L) \). We build the matrix

\[
W = \begin{pmatrix} e_1, \ldots, e_L \end{pmatrix}
\]

where each column of \( W \) represents one eigenvector. The matrix \( W \) is of size \( 208 \times L \). We order the eigenvector from left to right, in descending order, according to the eigenvalues associated to them.

Each new image \( I \) can then be processed by the scattered network. First we produce a vector \( \vec{s}(I) \). Then, for each vector \( \vec{s}(I) \) we remove the average vector \( \bar{A}S \), i.e., create \( \vec{x}(I_k) = \vec{s}(I_k) - \bar{A}S \). Then, we transform \( \vec{x}(I_k) \) to the new basis given by the matrix \( W \), i.e.,
\[
\tilde{Y}(I_k) = W^T \tilde{X}(I_k) = \begin{pmatrix}
e_1 \\
\vdots \\
e_L
\end{pmatrix} \tilde{X}(I_k)
\]

The vector \(\tilde{Y}(I_k)\) has dimensions \(L\) and each component is the dot product of the vector \(\tilde{X}(I_k)\) with the PCA eigenvector associated to that component.

Each component of \(\tilde{Y}(I_k)\) is expected to have different variance, according to the eigenvalues \(\alpha_j\). We may then compute the distance

\[
d^2_F = Y^T(I_k) \begin{bmatrix}
1 & 0 & \ldots & 0 \\
\frac{1}{\alpha_1} & \ldots & \ldots & 0 \\
0 & \frac{1}{\alpha_j} & \ldots & 0 \\
0 & \ldots & \frac{1}{\alpha_L} & 0
\end{bmatrix} Y(I_k)
\]

and if it is below some threshold \(T\), we accept as a face.

How to choose \(L\)? one may simply keep only the top \(~10\%\) of the eigenvectors, i.e., \(L=20\). One may keep the eigenvectors such that the eigenvalue is more than \(10\%\) of the largest eigenvalue, \(\alpha_1\). So if \(\alpha_j > 0.1 \alpha_1\) then \(e_j\) is included. You can choose a criteria and perform the same test as in Problem 2. Report performance.

**EXTRA: OPTIONAL. IMPROVING WITH THE SECOND LAYER**

Reapply the same concept of convolution with wavelets followed by a magnitude measure, followed by averaging with Gaussian, i.e.,

\[
S^2_{\lambda_j \lambda_i}(u) = G_{\sigma=6} * |\psi_{\lambda_j} * |\psi_{\lambda_i} * I(u)|
\]

But now \(\lambda_j = (\sigma = 6, \theta_j)\), i.e., there are only 4 different values for the parameter set \(\lambda_j\). Finally, we also downsample the output by a factor \(2^3\) along the x and y axis.

In total we have new \(48 = 4 \times 12\) different \(S^2_{\lambda_j \lambda_i}(u)\) downsampled “scattered images” in layer 2.

Redo Extra Problem 1, now adding the second Layer to the representation and create the much longer vector. When doing the PCA reduce the representation further (instead 10% use 5% or even less.)