

Laboratorio di Elaborazione delle Immagini 2

**Classificazione di segnali ECG con
Wavelet e SVM**

**Classificazione di features con WL e
PCA**



Esempio

- Questo esempio vi farà vedere come classificare in Matlab segnali EEG (elettrocardiogramma) usando features ricavate dalle WL e un classificatore SVM
- Toolbox necessari:
 - Wavelet Toolbox
 - Signal Processing Toolbox
 - Statistics and Machine Learning Toolbox
- Tutorial di riferimento:
<https://it.mathworks.com/help/wavelet/ug/ecg-signal-classification-using-wavelet-time-scattering.html>

Data

- Il dato utilizzato è disponibile pubblicamente ed è composto dall'ECG di tre gruppi di persone:
 - Soggetti normali (NSR)
 - Soggetti con aritmia cardiaca (ARR)
 - Soggetti con insufficienza cardiaca (CHF)
- Useremo 162 ECG provenienti da 3 database Physionet
 - [MIT-BIH Arrhythmia Database](#)
 - [MIT-BIH Normal Sinus Rhythm Database](#)
 - [The BIDMC Congestive Heart Failure Database](#)
- In totale avremo 96 ECG di persone con aritmia, 30 di persone con insufficienza cardiaca e 36 di persone normali.



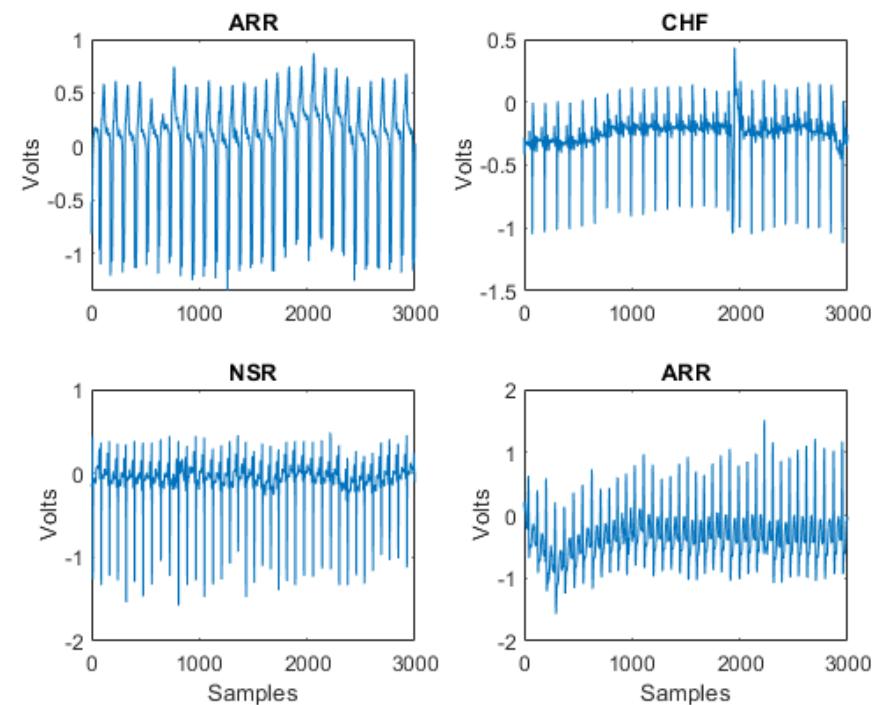
Download the data

- Come primo step è necessario scaricare il dataset dalla repository github
https://github.com/mathworks/physionet_ECG_data/
- Alternativamente si può scaricare da linea di comando con
git clone https://github.com/mathworks/physionet_ECG_data/
- Il file .zip contiene:
 - ECGData.zip
 - ECGData.mat – è il dato che useremo nell'esempio
 - Modified_physionet_data.txt
 - License.txt
 - README.md



Load files

```
unzip(fullfile(tempdir,'physionet_ECG_data-master.zip'),tempdir)  
  
unzip(fullfile(tempdir,'physionet_ECG_data-master','ECGData.zip'),...  
      fullfile(tempdir,'ECGData'))  
  
load(fullfile(tempdir,'ECGData','ECGData.mat'))  
  
helperPlotRandomRecords(ECGData,14)
```



Create training and testing sets

- Splittiamo randomicamente il dataset in training e testing (70% training e 30% testing). La funzione `helperRandomSplit` ritorna due dataset con le rispettive labels.
 - Ogni riga di `trainData` e `testData` è un segnale ECG
 - Ogni elemento di `trainLabels` e `testLabels` contiene la classe corrispondente

```
percent_train = 70;
```

```
[trainData,testData,trainLabels,testLabels] = ...  
    helperRandomSplit(percent_train,ECGData);
```

```
Ctrain = countcats(categorical(trainLabels))./numel(trainLabels).*100  
Ctest = countcats(categorical(testLabels))./numel(testLabels).*100
```

Feature extraction – Wavelet time scattering

- La Wavelet Scattering in MATLAB si calcola con la funzione `waveletScattering`, i cui parametri sono:

Properties

✓ **SignalLength — Signal length**
1024 (default) | positive integer ≥ 16

Signal length, specified as a positive integer ≥ 16 .

Data Types: double

✓ **SamplingFrequency — Sampling frequency**
1 (default) | positive scalar

Sampling frequency in hertz, specified as a positive scalar. If unspecified, frequencies are in cycles/sample and the Nyquist frequency is $\frac{1}{2}$.

Data Types: double

✓ **InvarianceScale — Scattering transform invariance scale**
one-half of `SignalLength` (default) | positive scalar

Scattering transform invariance scale, specified as a positive scalar. `InvarianceScale` specifies the translation invariance of the scattering transform. If you do not specify `SamplingFrequency`, `InvarianceScale` is measured in samples. If you specify `SamplingFrequency`, `InvarianceScale` is measured in seconds. By default, `InvarianceScale` is one-half the `SignalLength` in samples.

`InvarianceScale` cannot exceed `SignalLength` in samples.

Example: `sf = waveletScattering('SignalLength',1000,'SamplingFrequency',200,'InvarianceScale',5)` has the largest possible `InvarianceScale`.

Data Types: double

✓ **QualityFactors — Scattering filter bank Q factors**
[8 1] (default) | positive integer | vector of positive integers

Scattering filter bank Q factors, specified as a positive integer or a vector of positive integers. A filter bank Q factor is the number of wavelet filters per octave. Quality factors cannot exceed 32 and must be greater than or equal to 1.

If `QualityFactors` is specified as a vector, the elements of `QualityFactors` must be strictly decreasing.

Example: `sf = waveletScattering('QualityFactors',[8 2 1])` creates a wavelet scattering framework with three filter banks.

Data Types: double



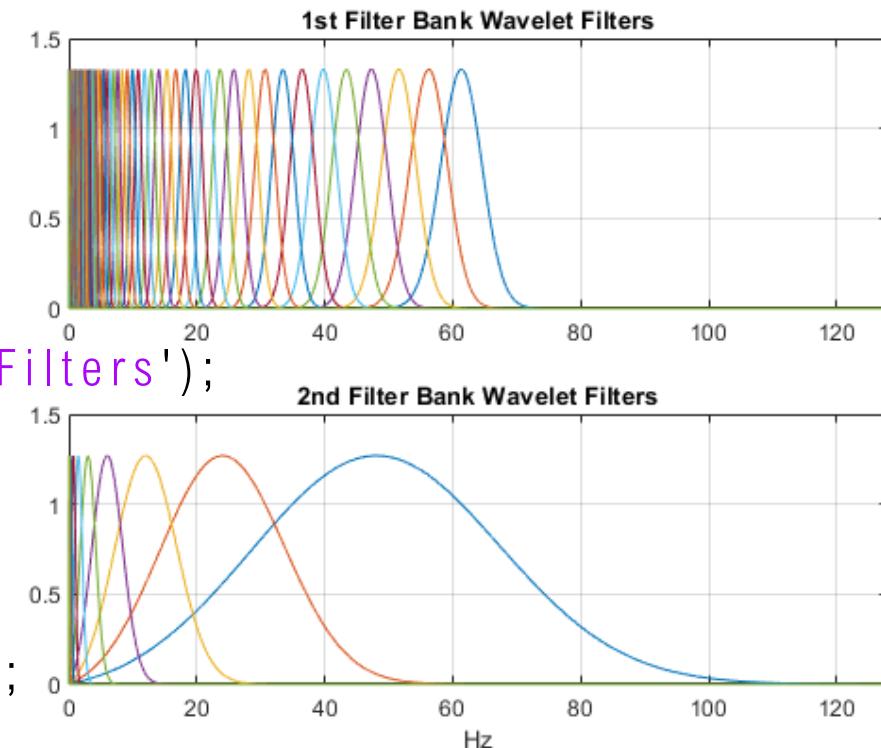
Wavelet time scattering

- In generale l'utilizzo di 2 banchi di filtri a cascata è sufficiente per ottenere buone performance.
- In questo esempio usiamo i banchi di filtri di default che contengono 8 WL per ottava per il primo banco di filtri e 1 WL per ottava nel secondo banco di filtri.
- La scala di invarianza è settata a 150 secondi (visto che stiamo analizzando una serie temporale la scala rappresenta una durata)

Wavelet time scattering

```
N = size(ECGData.Data,2);
sf = waveletScattering('SignalLength',N, ...
    'InvarianceScale',150,'SamplingFrequency',128);
```

```
%To visualize filter banks
[fb,f,filterparams] = filterbank(sf);
subplot(211)
plot(f,fb{2}.psift)
xlim([0 128])
grid on title('1st Filter Bank Wavelet Filters');
subplot(212)
plot(f,fb{3}.psift)
xlim([0 128])
grid on
title('2nd Filter Bank Wavelet Filters');
xlabel('Hz');
```



Wavelet Scattering – Invariance Scale

- Per dimostrare l'invarianza di scala possiamo calcolare l'inversa della FFT della scaling function e centralarla in 0 secondi.
- Poi possiamo plottare la parte reale e la parte immaginaria della WL alla scala più a bassa frequenza (coarse) del primo banco di filtri.
- Vedremo che le WL alla scala più grossolana non eccede la scala di invarianza determinata dal supporto temporale della scaling function [-75 secondi : 75 secondi]

Wavelet Scattering – Invariance Scale

```

figure;
phi = ifftshift(fftshift(fb{1}.phift));
psiL1 = ifftshift(fftshift(fb{2}.psift(:,end)));
t = (-2^15:2^15-1).*1/128;

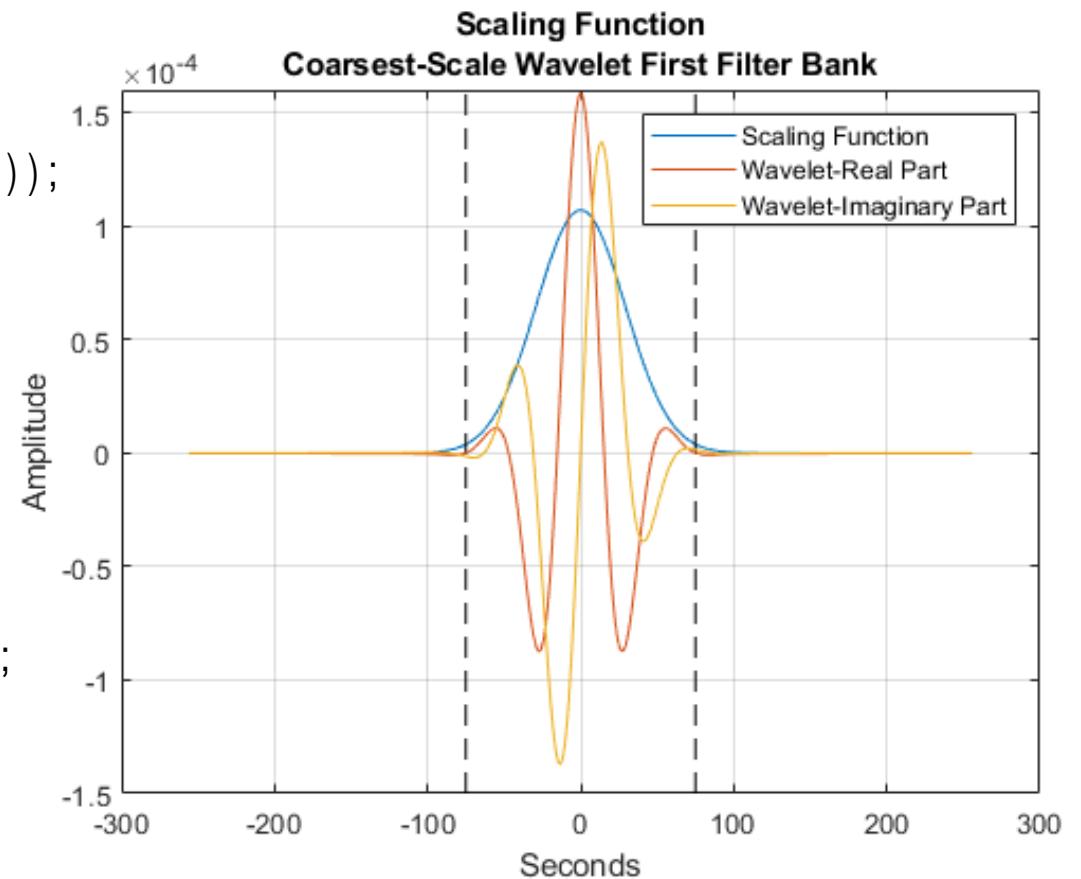
scalplt = plot(t,phi);

hold on
grid on
ylim([-1.5e-4 1.6e-4]);

plot([-75 -75],[-1.5e-4 1.6e-4],'k--');
plot([75 75],[-1.5e-4 1.6e-4],'k--');
xlabel('Seconds');

ylabel('Amplitude');
wavplt = plot(t,[real(psiL1) imag(psiL1)]); legend([scalplt wavplt(1)
wavplt(2)],{'Scaling Function','Wavelet-Real Part','Wavelet-Imaginary
Part'});
title({'Scaling Function';'Coarsest-Scale Wavelet First Filter Bank'})

```



Estrazione delle features

% La funzione featureMatrix permette di ottenere i coefficienti di scattering per i dati di training. L'output nel nostro caso è 461x16x113

```
scat_features_train = featureMatrix(sf,trainData');
```

%È necessario fare un reshape della matrice, per utilizzarla nelle SVM. Ogni colonna corrisponde a uno scattering path e ogni riga a una finestra temporale di scatterig. Si ottengono 1808 righe in quanto ci sono 16 finestre temporali e 113 segnali nel dato di training

```
Nwin = size(scat_features_train,2);  
scat_features_train = permute(scat_features_train,[2 3 1]);  
scat_features_train = reshape(scat_features_train,...  
    size(scat_features_train,1)*size(scat_features_train,2),[]);
```

```
scat_features_test = featureMatrix(sf,testData');  
scat_features_test = permute(scat_features_test,[2 3 1]);  
scat_features_test = reshape(scat_features_test,...  
    size(scat_features_test,1)*size(scat_features_test,2),[]);
```

%Infine dato che per ogni segnale abbiamo ottenuto 16 finestre di scattering bisogna creare delle labels che corrispondano al numero di finestre, utilizzando la funzione createSequenceLabels

```
[sequence_labels_train,sequence_labels_test] = ...  
    createSequenceLabels(Nwin,trainLabels,testLabels);
```



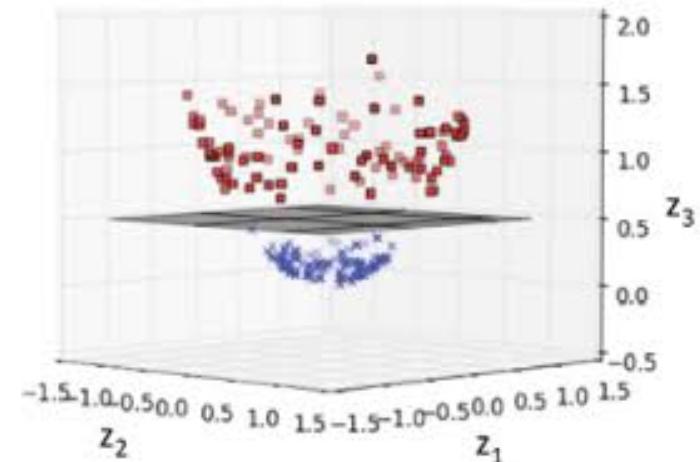
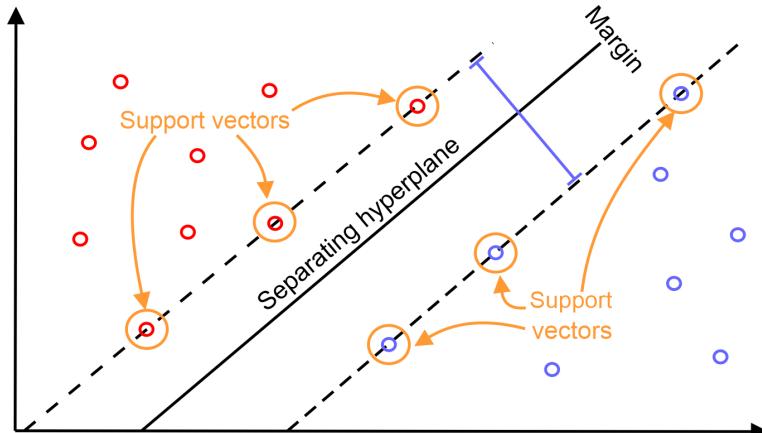
SVM - Cenni

- Approccio di ML lineare che cerca di determinare i decision boundaries tra le classi massimizzando il margine di separazione tra le classi stesse.

Se $\{x_1 \dots x_n\}$ è il dataset e $y_i \in \{1, -1\}$ contiene le labels di x_i

La soglia di decisione può essere trovata risolvendo:

$$\min\{\|w\|\} \quad \text{s.t.} \quad y_i(w^T x_i + b) \geq 1 \quad \forall i \in 1, \dots, n$$



- Kernel SVM: Si passa attraverso una trasformazione che mappa le features in uno spazio a dimensione maggiore dove si ipotizza siano maggiormente separabili

Classificazione con SVM – Cross validation

Come prima analisi usiamo **tutto il dato** con una SVM con kernel quadratico. In totale ci sono 2592 sequenze di scattering in tutto il dataset, 16 per ognuno dei 162 segnali. L'error rate, o loss, è stimata usando una 5-fold cross validation

```
scat_features = [scat_features_train; scat_features_test];
allLabels_scat = [sequence_labels_train; sequence_labels_test];

rng(1); template = templateSVM(...  

    'KernelFunction', 'polynomial', ...  

    'PolynomialOrder', 2, ...  

    'KernelScale', 'auto', ...  

    'BoxConstraint', 1, ...  

    'Standardize', true);
classificationSVM = fitcecoc(...  

    scat_features, ...  

    allLabels_scat, ...  

    'Learners', template, ...  

    'Coding', 'onevsone', ...  

    'ClassNames', {'ARR';'CHF';'NSR'});
kfoldmodel = crossval(classificationSVM, 'KFold', 5);
```



Confusion Matrix

```
%calcolo della Loss e della confusion matrix
predLabels = kfoldPredict(kfoldmodel);
loss = kfoldLoss(kfoldmodel)*100;
confmatCV = confusionmat(allLabels_scat,predLabels)
fprintf('Accuracy is %2.2f percent.\n',100-loss);

%visto che abbiamo 16 classificazioni diverse per ogni segnale si può usare un
majority vote per ottenere una predizione della singola classe per ogni
rappresentazione scattering
classes = categorical({'ARR','CHF','NSR'});
[ClassVotes,ClassCounts] =...
    helperMajorityVote(predLabels,[trainLabels;testLabels],classes);

%Ricalcolo della vera cross-validation basata sulla moda della predizione della classe
per ogni insieme di finestre temporali di scattering
CVaccuracy = sum(eq(ClassVotes,categorical([trainLabels;...
    testLabels])))/162*100;
fprintf('True cross-validation accuracy is %2.2f percent.\n',CVaccuracy);
MVconfmatCV = confusionmat(ClassVotes,categorical([trainLabels;...
    testLabels]));
```



Classificazione con SVM – senza Cross validation

Come successiva analisi fittiamo la SVM con kernel quadratico solo sul dato di training (70%) e successivamente usiamo quel modello per predire il testing set (30%).

```
model = fitcecoc(...  
    scat_features_train, ...  
    sequence_labels_train, ...  
    'Learners', template, ...  
    'Coding', 'onevsone', ...  
    'ClassNames', {'ARR','CHF','NSR'});  
  
predLabels = predict(model,scat_features_test);  
  
[TestVotes,TestCounts] = ...  
    helperMajorityVote(predLabels,testLabels,classes);  
testaccuracy = ...  
    sum(eq(TestVotes,categorical(testLabels)))/numel(testLabels)*100;  
  
fprintf('The test accuracy is %2.2f percent. \n',testaccuracy);  
testconfmat = confusionmat(TestVotes,categorical(testLabels));
```



Esercizio 1

- Nell'esempio abbiamo visto come usare la scattering transform per la classificazione di dati ECG.
- Provare a cambiare i parametri della funzione scatteringTransform in modo da vedere come variano le performance.
 - Cambiare il numero di filtri a cascata
 - Cambiare il numero di WL in ogni banco di filtri
 - ...
- Sarà necessario controllare ed aggiustare le dimensioni delle matrici a seconda dei test che si fanno
- Testare le differenze facendo il training sul 70% dei dati e il testing sul 30%

Esercizio 2

- Provare a sostituire la scattering transform con la Discrete Wavelet Transform (DWT)
- Il framework dovrà essere leggermente modificato in quanto sarà necessario calcolare la dwt separatamente per ogni segnale EEG. Si può provare ad usare direttamente il vettore risultante come feature in input oppure calcolare varie metriche come:
 - Entropia
 - Varianza
 - Standard deviation
 - Mediana
 - Gli altri percentili (5,25,75,95)
 - Errore quadratico medio
 - ...

Usandole tutte come feature matrix per la SVM



Esempio immagini – Classificazione di textures

- Per questo esempio useremo il dataset pubblico KTH-TIPS (Textures under varying Illumination, Pose and Scale) nella versione greyscale.
- Ci sono 810 immagini 200x200 suddivise in 10 classi bilanciate di texture
- È possibile scaricare il dataset da
<https://www.csc.kth.se/cvap/databases/kth-tips/>

(Download -> KTH-TIPS -> greyscalePNGimages(23MB))

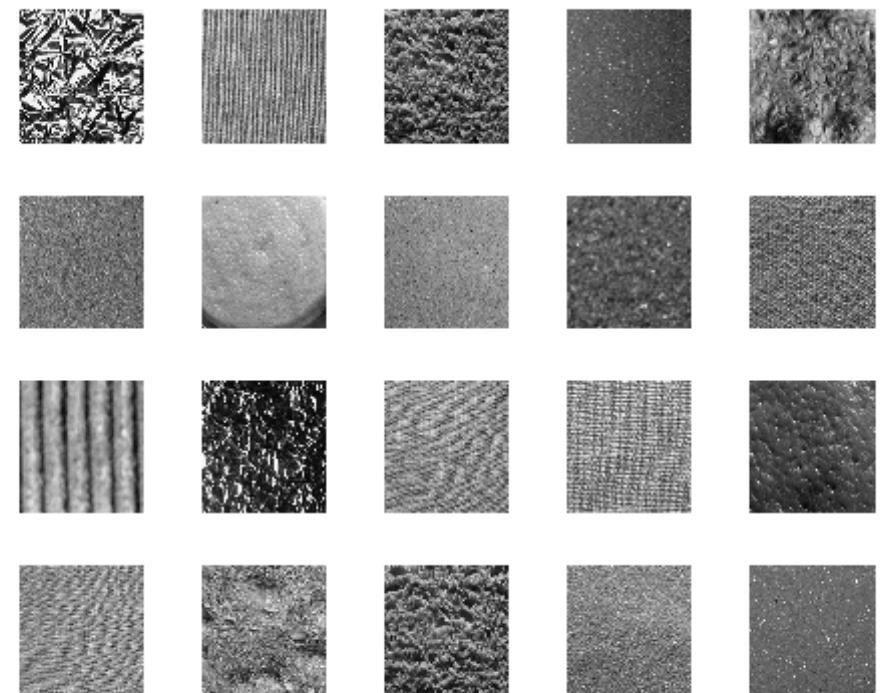
- Toolbox Matlab necessari:
 - Image processing toolbox
 - Wavelet toolbox



Load the data

```
%modificare in base al path del dato
location = fullfile(tempdir,'kth_tips_grey_200x200','KTH_TIPS');
Imds = imageDatastore(location,'IncludeSubFolders',true, ...
    'FileExtensions','.png','LabelSource','foldernames');
```

```
%visualizzazione di 20 immagini dal dataset
numImages = 810;
perm = randperm(numImages,20);
for np = 1:20
    subplot(4,5,np)
    im = imread(Imds.Files{perm(np)} );
    imagesc(im);
    colormap gray; axis off;
end
```



Training and testing sets

```
% Random split del dataset in training e testing, approssimativamente 80% training e  
% 20% testing  
Imds = shuffle(Imds);  
[trainImds,testImds] = splitEachLabel(Imds,0.8);  
  
%Abbiamo ottenuto il training di 650 immagini (65 per classe) e il testing di 160  
immagini (16 per classe)  
countEachLabel(trainImds)  
countEachLabel(testImds)  
  
%Trasformiamo gli array in tall arrays (utile se si usa il parallel toolbox)  
Ttrain = tall(trainImds);  
Ttest = tall(testImds);
```

Wavelet scattering

%Creiamo il framework di scattering dando in input l'image size e l'invariance scale e lasciando gli altri iperparametri di default

```
sn = waveletScattering2('ImageSize',[200 200],'InvarianceScale',150);
```

%Per estrarre le features usiamo la funzione HelpScatImages_mean che prima fa un rescaling di tutte le immagini a 200x200 e poi calcola la feature matrix, in questo caso ogni feature matrix è 391x7x7 (Ci sono 391 scattering paths e ogni scattering coefficient è 7x7). La funzione fa anche una media nella seconda e terza dimensione per ottenere un vettore di 391 elementi per ogni immagine.

```
trainfeatures = cellfun(@(x)helperScatImages_mean(sn,x),Ttrain,'Uni',0);  
testfeatures = cellfun(@(x)helperScatImages_mean(sn,x),Ttest,'Uni',0);
```

%A questo punto, sfruttando la funzione di gathering dei array tall uniamo tutti i vettori di fette di training e testing e li concateniamo. Si ottengono due matrici con 391 righe e numero di colonne pari al numero di immagini del training e del testing set

```
Trainf = gather(trainfeatures);  
trainfeatures = cat(2,Trainf{:});  
Testf = gather(testfeatures);  
testfeatures = cat(2,Testf{:});
```



PCA model and prediction

% Il classificatore usato per questo esempio si basa sulla PCA del vettore di feature scattering di ogni classe. Con la funzione helperPCAModel si determinano le componenti principali per ogni classe basata sulle feature di scattering. La funzione helperPCAClassifier invece classifica il testing set cercando il match migliore (ovvero la migliore proiezione) tra le componenti principali di ogni vettore di test con il training, assegnano la relativa classe.

```
model = helperPCAModel(trainfeatures,30,trainImds.Labels);  
predlabels = helperPCAClassifier(testfeatures,model);
```

```
%Calcolo dell'accuracy  
accuracy = sum(testImds.Labels ==  
predlabels)./numel(testImds.Labels)*100
```

```
%plot della confusion matrix  
figure  
confusionchart(testImds.Labels,predlabels)
```

Esercizio 3

- Provare ad utilizzare la SVM come classificatore al posto della PCA
- Sostituire la scattering transform con la dwt2 calcolando le feature descritte nell'esercizio precedente e vedere come varia l'accuracy di classificazione.