Laboratorio di Elaborazione delle Immagini 2

# Classificazione di segnali ECG con Wavelet e CNN

# Esempio

- Questo esempio vi farà vedere come classificare in Matlab segnali EEG (elettrocardiogramma) usando features ricavate dalle WL e una Convolutional Neural Network (CNN)

- Toolbox necessari:
  - Wavelet Toolbox
  - Image Processing Toolbox
  - Deep Learning Toolbox Model for GoogLeNet Network
  - Deep Learning Toolbox

- Tutorial di riferimento: https://it.mathworks.com/help/wavelet/ug/classify-time-series-using-wavelet-analysis-and-deep-learning.html

# Data

- Il dato utilizzato è lo stesso dell'esercitazione precedente, disponibile pubblicamente. È composto dall'ECG di tre gruppi di persone:
  - Soggetti normali (NSR)
  - Soggetti con aritmia cardiaca (ARR)
  - Soggetti con insufficienza cardiaca (CHF)

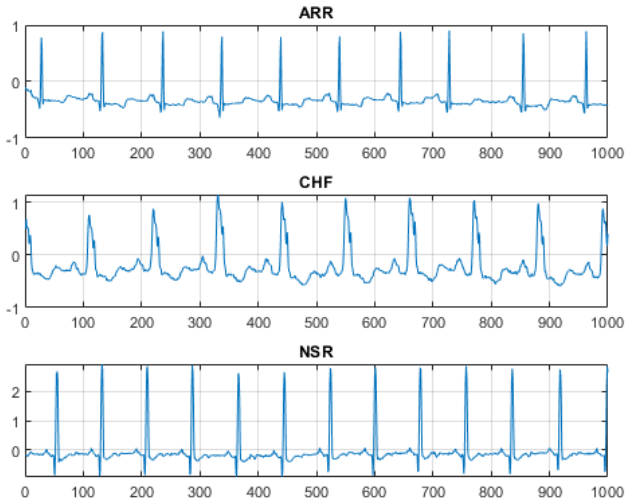
- Useremo 162 ECG provenienti da 3 database Physionet
  - MIT-BIH Arrhythmia Database
  - MIT-BIH Normal Sinus Rhythm Database
  - The BIDMC Congestive Heart Failure Database

- In totale avremo 96 ECG di persone con aritmia, 30 di persone con insufficienza cardiaca e 36 di persone normali.

# Load files

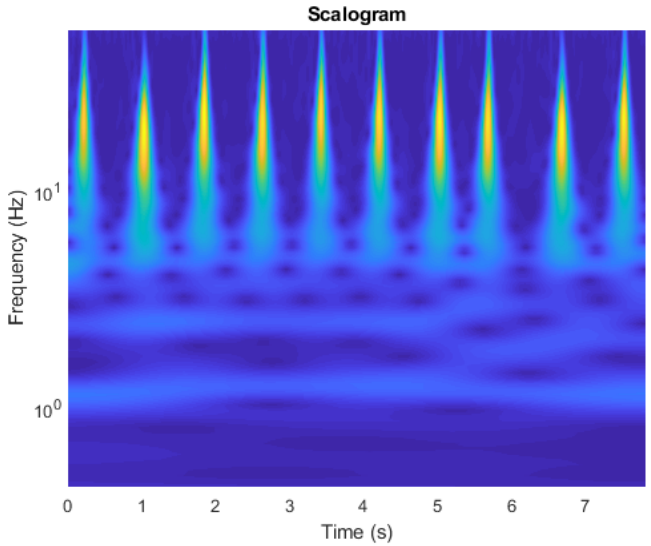
```
unzip(fullfile(tempdir,'physionet_ECG_data-master.zip'),tempdir)

unzip(fullfile(tempdir,'physionet_ECG_data-master','ECGData.zip'),...
        fullfile(tempdir,'ECGData'))

load(fullfile(tempdir,'ECGData','ECGData.mat'))

parentDir = tempdir; dataDir = 'data';
helperCreateECGDirectories(ECGData,parentDir,dataDir)
helperPlotReps(ECGData)
```

ECGData è un array di tipo structure con due campi: Data e Labels. Data è 162x65536 in cui ogni riga è un EEG campionato a 128Hz. Labels è un cell array 162x1 che contiene le etichette diagnostiche del dato



# Creazione delle rappresentazioni tempo - frequenza

```
%Come feutures per la CNN utilizzeremo gli scalogrammi, che rappresentano il valore
assoluto dei coefficient della CWT di un segnale

Fs = 128; fb = cwtfilterbank('SignalLength',1000,...
        'SamplingFrequency',Fs,...
        'VoicesPerOctave',12);
sig = ECGData.Data(1,1:1000);

[cfs,frq] = wt(fb,sig);
t = (0:999)/Fs;

figure;
pcolor(t,frq,abs(cfs)) set(gca,'yscale','log');
shading interp;axis tight;
title('Scalogram');
xlabel('Time (s)');
ylabel('Frequency (Hz)')
```




```
%Calcolo di tutti gli scalogrammi di ECGData. Ogni imagine RGB derivante sarà
224x224x3 in modo da essere compatibile con la GoogLeNet
helperCreateRGBfromTF(ECGData,parentDir,dataDir)
```

# Training and Validation

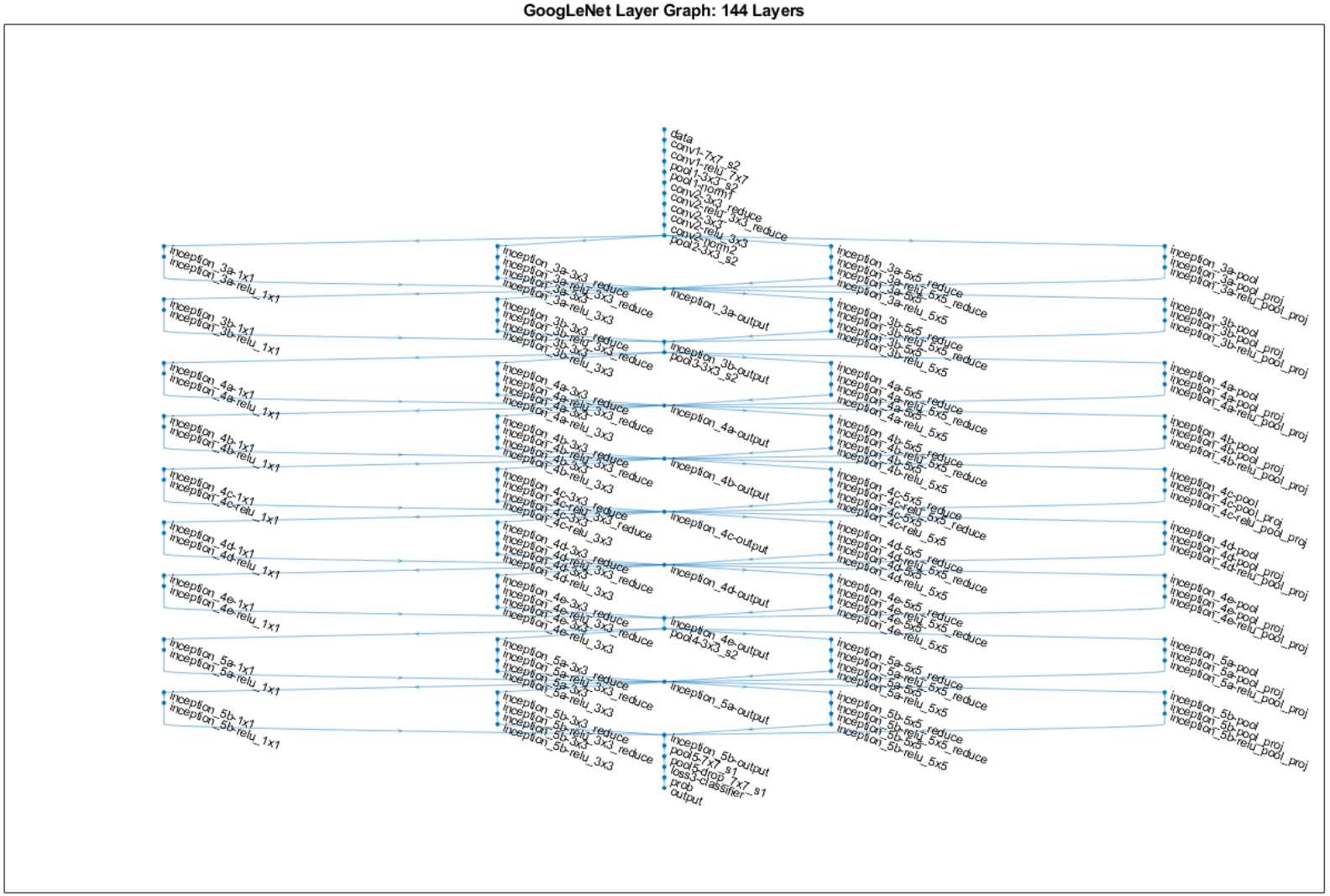
```
%carichiamo tutti gli scalogrammi come un datastore. La funzione imageDatastore
etichetta autmaticamente tutte le immagini in base al nome della cartella e li salva
come oggetto datastore
allImages = imageDatastore(fullfile(parentDir,dataDir),...
        'IncludeSubfolders',true,...
        'LabelSource','foldernames');
%divisione randomica delle immagini in 2 gruppi. 80% training e 20% validation
rng default
[imgsTrain,imgsValidation] = splitEachLabel(allImages,0.8,'randomized');

disp(['Number of training images: ',num2str(numel(imgsTrain.Files))]);
%Number of training images: 130

disp(['Number of validation images ',num2str(numel(imgsValidation.Files))]);
%Number of training images: 130
```

# Transfer learning – Google Net

```
%Caricamento della GoogLeNet pretrained
net = googlenet;

%Visualizzazione della rete
lgraph = layerGraph(net);
numberOfLayers = numel(lgraph.Layers);
figure('Units','normalized','Position',[0.1 0.1 0.8 0.8]);
plot(lgraph)
title(['GoogLeNet Layer Graph: ',num2str(numberOfLayers),' Layers']);

%Ispezione del primo layer
net.Layers(1)
```

```
ans =
        ImageInputLayer with properties:
                Name: 'data'
                InputSize: [224 224 3]
        Hyperparameters
                DataAugmentation: 'none'
                Normalization: 'zerocenter'
                Mean: [224×224×3 single]
```

# GoogLeNet


GoogLeNet Layer Graph: 144 Layers
data
conv1-7x7_s2
conv1-relu_7x7
pool1-3x3_s2
pool1-norm1
conv2-3x3_reduce
conv2-relu_3x3_reduce
conv2-3x3
conv2-relu_3x3
conv2-norm2
pool2-3x3_s2
inception_3a-output
inception_3b-output
pool3-3x3_s2
inception_4a-output
inception_4b-output
inception_4c-output
inception_4d-output
inception_4e-output
pool4-3x3_s2
inception_5a-output
inception_5b-output
pool5-7x7_s1
pool5-drop_7x7_s1
loss3-classifier
prob
output

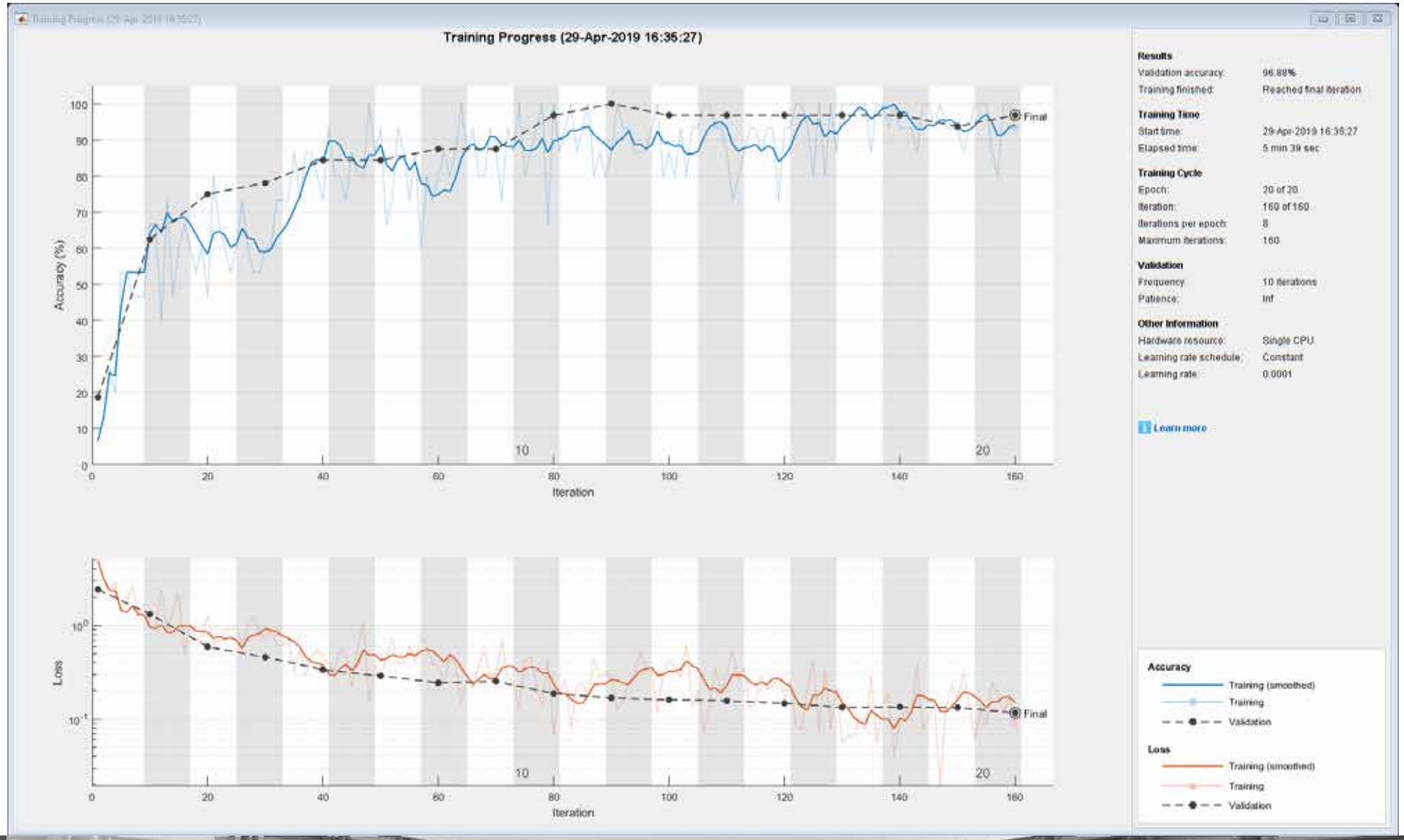
# Modify GoogLeNet parameters

```
%Ogni layer può essere considerate come filtro. I primi layer identificano le
features più comuni come contorni e colori. I layer successive sono più sensibili a
features più specifiche che differenziano le cateogrie. La rete è pretrained su
1000 classi quindi la dobbiamo riallenare per I nostril ECG. Per evitare overfitting
aggiungiamo un layer di dropout che sostituiamo al layer 'pool5-drop_7x7_s1
newDropoutLayer = dropoutLayer(0.6,'Name','new_Dropout');
lgraph = replaceLayer(lgraph,'pool5-drop_7x7_s1',newDropoutLayer);

%I layer loss3-classifier e output definiscono come combinare le feature
estratte nei layer convoluzionali in probabilità di classificazione, loss e
etichette predette. Vanno quindi modificati in base al dato che stiamo usando.
numClasses = numel(categories(imgsTrain.Labels));
newConnectedLayer = fullyConnectedLayer(numClasses,'Name','new_fc',...
 'WeightLearnRateFactor',5,'BiasLearnRateFactor',5);
lgraph = replaceLayer(lgraph,'loss3-classifier',newConnectedLayer);

% Il layer di classificazione specifica la classe di output della rete. Lo sostituiamo
con un layer senza etichette, che saranno automaticamente date durante il training
newClassLayer = classificationLayer('Name','new_classoutput'); lgraph =
replaceLayer(lgraph,'output',newClassLayer);
```

# Set training options and train GoogLeNet

```
%Il training delle CNN è un processo iterative che minimizza la loss. Ad ogni
iterazione si calcola il gradiente della loss e I pesi vengono aggiornati.
%InitialLearnRate specifica lo step size iniziale nella direzione del gradiente della
loss. MiniBatchSize specifica la dimensione del subset di training usato in ogni
iterazione. MaxEpoch specifica il numero Massimo di epoche da usare per il
training

options = trainingOptions('sgdm',...
        'MiniBatchSize',15,...
        'MaxEpochs',20,...
        'InitialLearnRate',1e-4,...
        'ValidationData',imgsValidation,...
        'ValidationFrequency',10,...
        'Verbose',1,...
        'ExecutionEnvironment','cpu',...
        'Plots','training-progress');
rng default

%train the network
trainedGN = trainNetwork(imgsTrain,lgraph,options);
```

# Training process



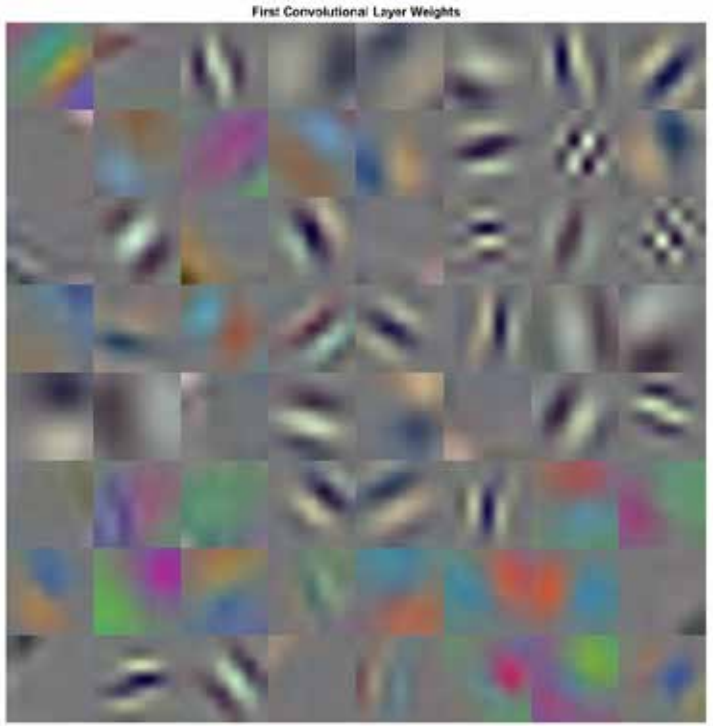
# Evaluate GoogLeNet accuracy

```
%Valutare le performance di classificazione sul set di validazione

[YPred,probs] = classify(trainedGN,imgsValidation);
accuracy = mean(YPred==imgsValidation.Labels);
disp(['GoogLeNet Accuracy: ',num2str(100*accuracy),'%'])
        GoogLeNet Accuracy: 96.875%


%La situazione ottimale sarebbe quella di divider il dataser in training/validation e
testing e usare il testing per quest'ultima operazione
```

# Visualize the activations

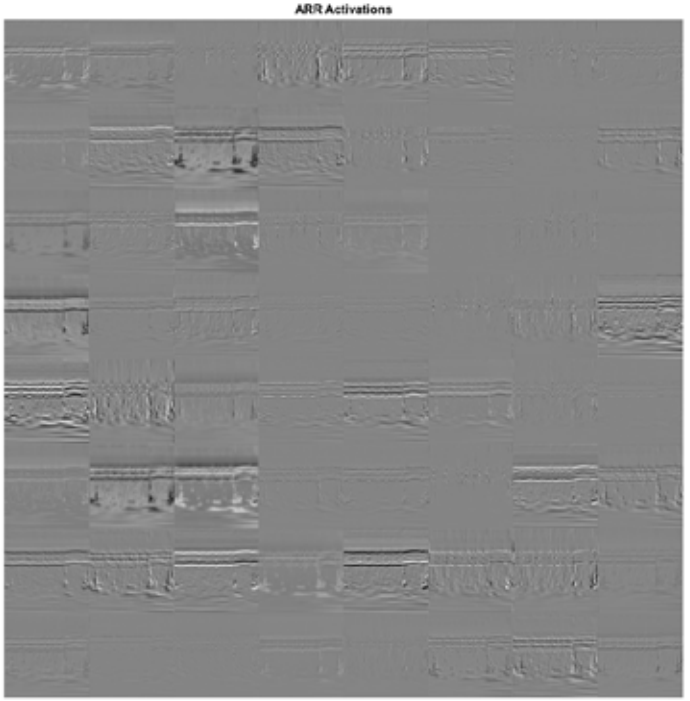
```
%Ogni layer della CNN produce una risposta, o attivazione, all'imagine in input
corrispondenti alle features estratte. Nel primo latyer ci sono 64 set individuali di
pesi che corrispondono ai filtra della CNN
```

```
wghts = trainedGN.Layers(2).Weights;
wghts = rescale(wghts);
wghts = imresize(wghts,5);
figure
montage(wghts)
title('First Convolutional Layer Weights')
```



# Feature visualization

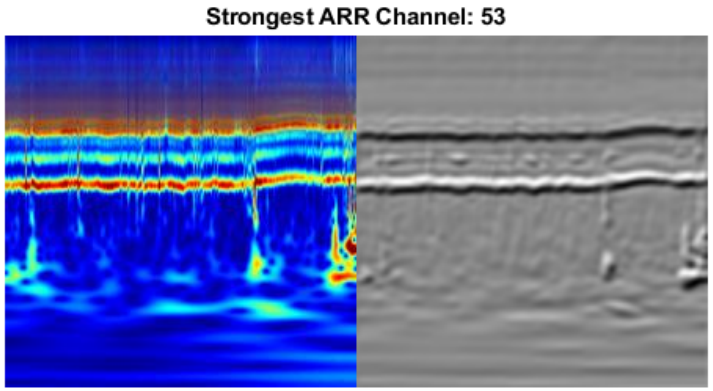
```
convLayer = 'conv1-7x7_s2';
imgClass = 'ARR';
imgName = 'ARR_10.jpg';
imarr =
imread(fullfile(parentDir,dataDir,imgClass,...
        imgName));
trainingFeaturesARR =...
activations(trainedGN,imarr,convLayer);
sz = size(trainingFeaturesARR);
trainingFeaturesARR =...
reshape(trainingFeaturesARR,[sz(1) sz(2) 1
sz(3)]);
figure
montage(rescale(trainingFeaturesARR),...
'Size',[8 8])
title([imgClass,' Activations'])
```




Si possono esaminare le feature che la rete prende in considerazione comparando le aree di attivazione con l'immagine originale.
Facciamo passare un'immagine di prova lungo la rete ed esaminiamo per esempio l'output di attivazione del primo layer convolutivo

# Feature visualization

```
% Visualizzazione del canale più forte per l'imagine di prova
imgSize = size(imarr);
imgSize = imgSize(1:2);
[~,maxValueIndex] = max(max(max(trainingFeaturesARR)));
arrMax = trainingFeaturesARR(:,:,:,maxValueIndex);
arrMax = rescale(arrMax);
arrMax = imresize(arrMax,imgSize);
figure; imshowpair(imarr,arrMax,'montage')
title(['Strongest ',imgClass,' Channel: ',num2str(maxValueIndex)])
```



# ScatNet e progetti per il Lab

# ScatNet

- Software MATLAB che include l'analisi e la classificazione di segnali 1D e di immagini implementando la trasformata scattering
- Permette la creazione di una scattering network, ovvero uno stack di signal processing layers di dimensione via via maggiori, che fornirà in output le feature scattering utili per la classificazione

E' possibile scaricare la libreria e tutta la relativa documentazione al sito:

https://www.di.ens.fr/data/software/scatnet/

# Esempio con una sola immagine

```
x = uiuc_sample;
imagesc(x);
colormap gray;
```

# Wavelet operators

Il software mette a disposizione alcune funzioni per precomputare gli operatori delle WL, ad hoc per del tipo di segnale che si vuole analizzare.

```
Wop = wavelet_factory_2d(size(x));
```

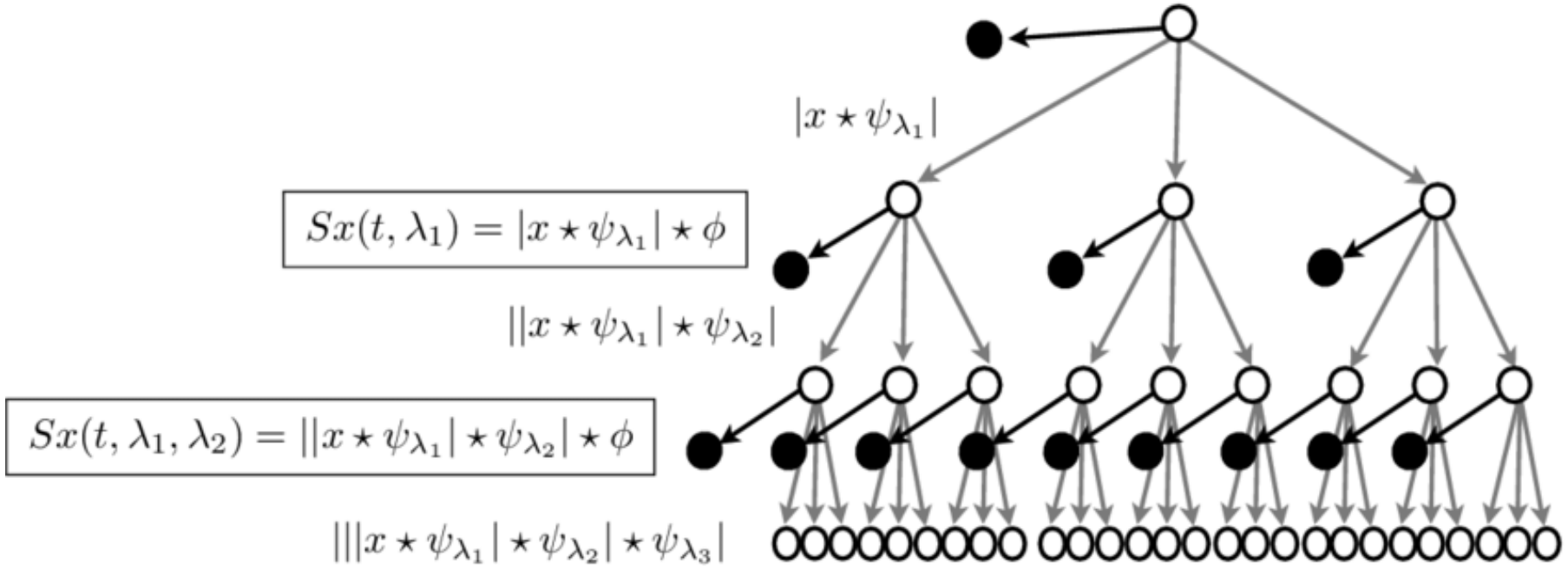
La `wavelet_factory_2d` rappresenta i banchi di filtri delle WL 2D per avere una rappresentazione scattering dell'immagine invariante fino alla traslazione planare

Esistono altre wavelet factory per le immagini che vi invitiamo a testare (`wavelet_factory_2d_pyramid`, `wavelet_factory_3d`)

# Wop

Ogni operatore `wop{1+m}` restituisce:

1. Una media pesata secondo la scala maggiore, attraverso il filtro lowpass $\boldsymbol{\Phi}$
2. Uno scattering lungo le scale attraverso il filtro passabanda $\boldsymbol{\Psi}$



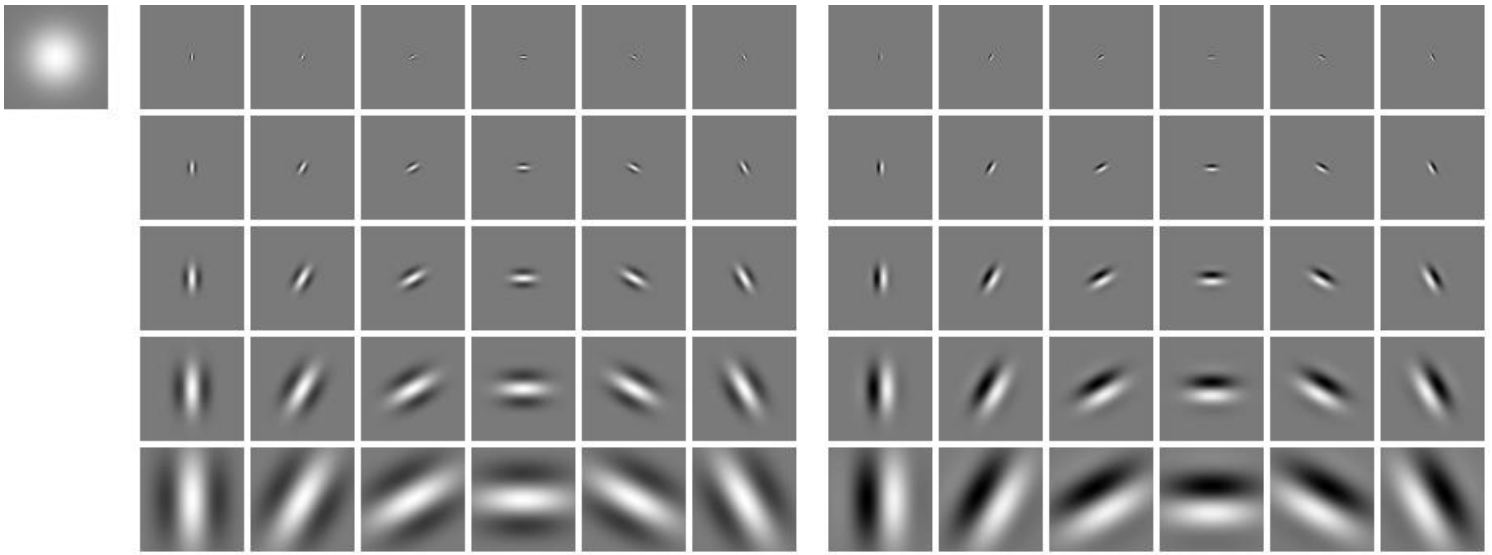
# Filtri

- I filtri usati nel calcolo della trasformata scattering si possono ottenere come secondo argomento della funzione

```
[Wop, filters] = wavelet_factory_2d(size(x));
```

# Visualizzazione dei filtri

```
filt_opt.J = 5;
filt_opt.L = 6;
[Wop, filters] = wavelet_factory_2d([512, 512],
filt_opt);
% display all the filters
display_filter_bank_2d(filters);
```



# Calcolo dei coefficienti scattering

- Una volta definiti gli operatori da applicare possiamo utilizzare la funzione per calcolare la trasformata scattering

```
S = scat(x, Wop);
```

- Il risultato è una variabile di tipo struct di lunghezza M+1 dove M indica l'ordine massimo della scattering transform. S{m} contiene i coefficienti di scattering del layer m.
- E' possibile ottenere in output anche i coefficienti U risultanti dall'applicazione di $\boldsymbol{\Psi}$

```
[S,U] = scat(x, Wop);
```

# Reformat in a 3D array

- La S in formato struct non permette una manipolazione numerica diretta per cui si può usare la funzione

```
S_mat = format_scat(Sx);
```

Per generare una matrice 3D. Nell'esempio viene generata una matrice 417x60x80.

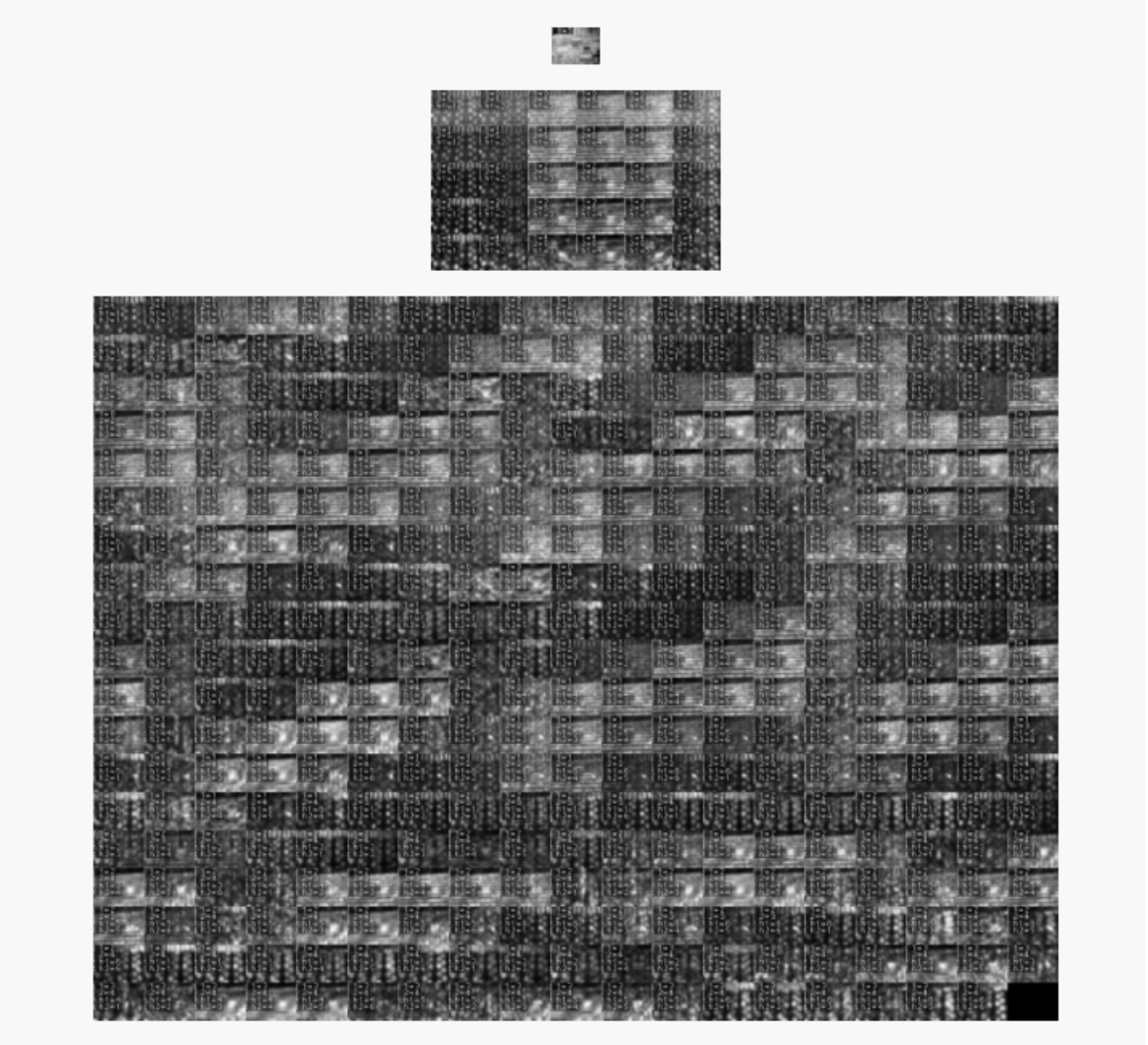
La prima dimensione è l'indice del path mentre la seconda e la terza corrispondono alle coordinate spaziali sottocampionate rispetto all'immagine originale

# Visualizzazione dei coefficienti

- Per visualizzare tutti i coefficienti di scattering

```
% compute scattering with 5 scales, 6 orientations
% and an oversampling factor of 2^2
x = uiuc_sample;
filt_opt.J = 5;
filt_opt.L = 6;
scat_opt.oversampling = 2;
Wop = wavelet_factory_2d(size(x), filt_opt,
scat_opt);
Sx = scat(x, Wop);
% display scattering coefficients
image_scat(Sx)
```

# Visualizzazione dei coefficienti



# Customizzazione

- E' possibile personalizzare ogni aspetto del calcolo della scattering transform attraverso la modifica delle variabili `filt_opt` e `scat_opt`
- `Filt_opt.J` rappresenta il numero di scale j nel banco di filtri
- `Filt_opt.L` rappresenta il numero di orientazioni
- `Filt_opt.Q` rappresenta il numero di scale per ottava

- `Scat_opt.M` è l'ordine massimo di scattering. La funzione scat calcolerà i coefficienti dall'ordine 0 a M
- `Scat_opt.oversampling` è il livello di sovracampionamento della trasformata che può essere settato fino a una potenza di 2 rispetto al campionamento dell'immagine

# Esempio classificazione

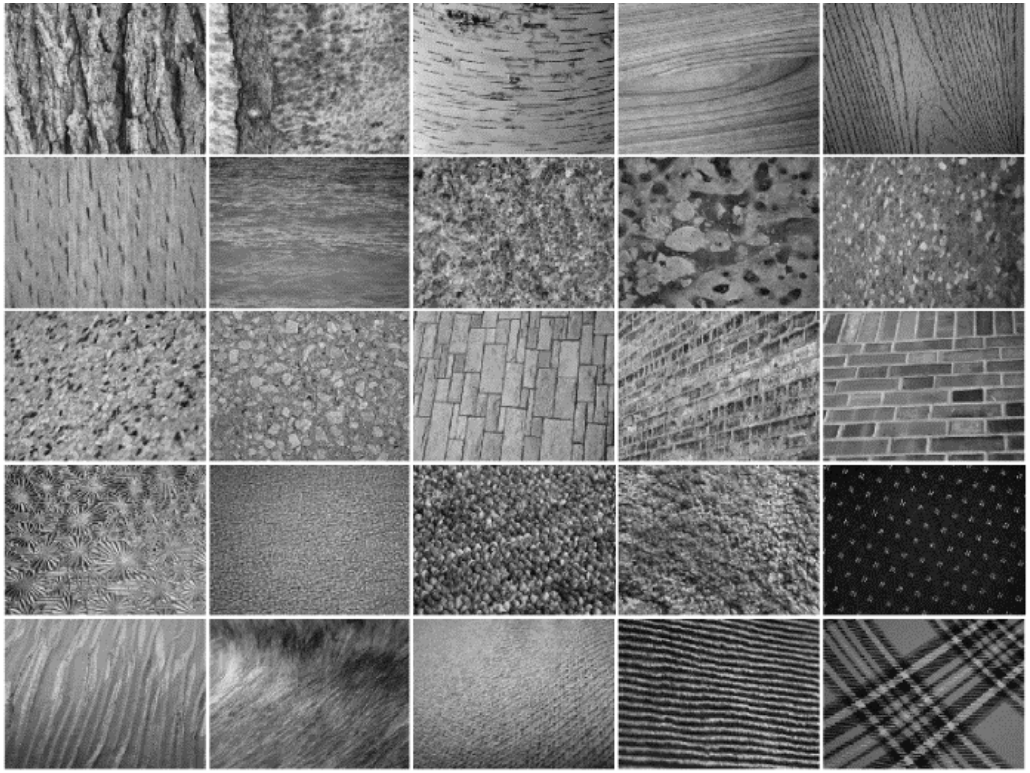
```
src = uiuc_src(path/to/dataset);
filt_opt.J = 5;
scat_opt.oversampling = 0;
Wop = wavelet_factory_2d([480, 640], filt_opt,
scat_opt);
features{1} =
@(x)(sum(sum(format_scat(scat(x,Wop)),2),3));
options.parallel = 0;
db = prepare_database(src, features, options);
```

# SVM classification

```
% proportion of training example
prop = 0.5;
% split between training and testing
[train_set, test_set] = create_partition(src,
prop);
% kernel type
train opt.kernel type = 'linear';
train opt.C = 8;
% training
model = svm_train(db, train_set, train_opt);
% testing
labels = svm_test(db, model, test_set);
% compute the error

error  = classif_err(labels, test_set, src);
```

# Progetti per il LAB

- Prendiamo in considerazione il dataset UIUC:
  - 1000 textures 256x256 in scala di grigi
  - 25 classi



# Progetti LAB - Task

Confrontare la performance di classificazione che si ottiene utilizzando la ScatNet per l'estrazione di feature seguito dalla classificazione tramite SVM con:

1. Feature derivate dalla FFT2 e SVM come classificatore;
2. Feature derivate dalla dwt2D e SVM come classificatore;
3. Scatterogramma delle texture come input a una CNN (tipo esempio GoogLeNet)
4. Scalogramma della dwt2 delle texture come input a una CNN (tipo esempio GoogLeNet)

Il progetto può essere svolto in gruppi di 2 persone. Potete scegliere una sola metodologia (1-4) rispetto alla quale fare il confronto.

# Hints

- Per la scatNet potete provare ad utilizzare anche le feature derivanti dalle diverse wavelet factories introdotte nelle slide precedenti.

- Se scegliete il confronto con le CNN potete effettuare anche una visualizzazione delle feature in modo da confrontare, anche solo visivamente con i coefficienti di scattering, se c'è differenza tra le feature più importanti per un metodo e per l'altro. La visualizzazione potete effettuala anche solo rispetto a qualche immagine di esempio del validation set.

# Database + istruzioni

- Abbiamo creato una cartella condivisa sul drive in cui è presente
  - Il software per la scatnet con la relativa documentazione
  - Il database di texture sul quale lavorare

Cartella OneDrive

N.B. Se notate che il calcolo delle features per tutto il dataset richiede troppo tempo potete ovviamente diminuire il numero di classi del task