# Elaborazione delle Immagini

## Laboratorio 7

Obiettivi:

- Classificazione di immagini

Ricordate: per processare le immagini è sempre conveniente trasformale in valori double tra 0 e 1 con **im2double**.

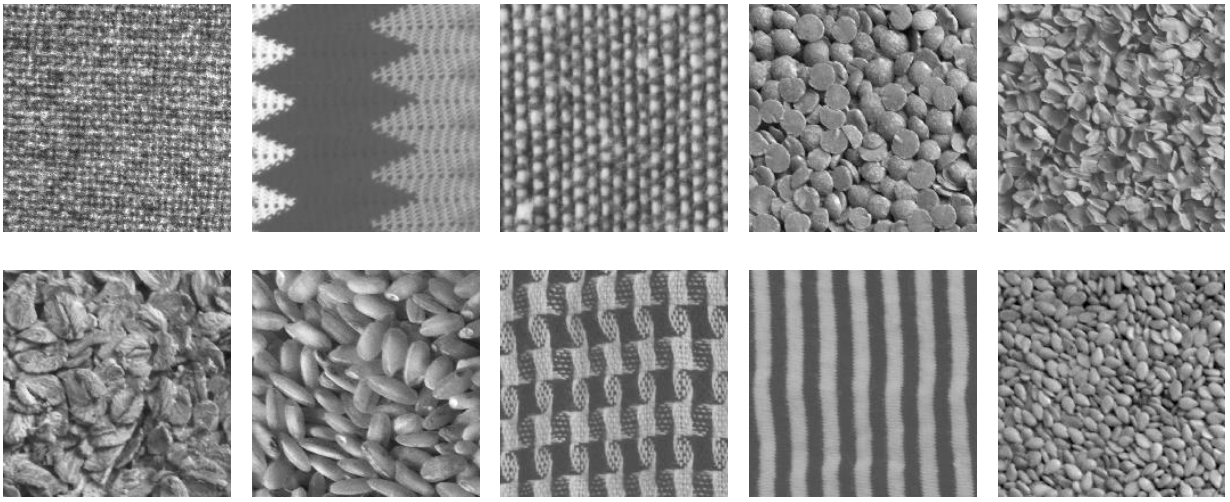
Ricordate: imshow visualizza le immagini in modo corretto se hanno valori tra 0 e 255 (uchar8), se hanno valori tra 0 e 1 (double) o sono valori logici.

Ricordate: se volete saperne di più sulle funzioni Matlab usate, consultate l'help o la documentazione con i seguenti comandi da console:
help <funzione>
doc <funzione>

Scrivete il codice di ogni esercizio in uno script separato (labX_1.m, labX_2.m, ...)

---

L'obiettivo è quello di classificare le 200 immagini di texture (**miniset.zip**), che compongono il dataset, nelle 10 classi: blanket1, blanket2, cushion1, lentils1, linseeds1, oatmeal1, rice2, scarf1, scarf2, sesameseeds.



Le immagini sono a livelli di grigio e rappresentano sostanzialmente 10 texture diverse. Per questo motivo si useranno principalmente descrittori di texture.

Sono dati due file di testo: **images.list** e **labels.list.** Questi file contengono la lista delle immagini del dataset e, per ognuna, la rispettiva etichetta (di tipo stringa) della classe.

Per poter classificare queste immagini è necessario un pre-processing per creare i set di descrittori di feature da usare per il training e per testare il classificatore creato.

A1. In uno script, usate la funzione "**readlists.m**", per leggere dai file di testo la lista delle immagini e le rispettive etichette. La funzione non prende parametri e ritorna due variabili **images** e **labels.** Guardate cosa contengono queste due variabili.

B1. Aggiungete il codice che calcola, per ognuna delle immagini in **images,** tre descrittori di texture tramite le funzioni date: **compute_lbp**, **compute_ghist**, **compute_glcm**. Concatenate ciascuno dei tre descrittori in un distinto array: **lbp**, **ghist**, **glcm**. Si deve avere un descrittore per riga.

C1. Usando la funzione save di Matlab, salvate su file il workspace contenente le variabili **images**, **labels**, **glcm**, **lbp** e **ghist**.

Il workspace salvato sarà la base per costruire i classificatori dei prossimi esercizi.

A2. In un nuovo script. Caricate il workspace salvato in precedenza.

B2. Usando la funzione Matlab **cvpartition**, partizionate le etichette in due insiemi: quello di training e quello di test. Usando l'opzione 'Holdout' e il valore 0.2, la funzione seleziona l'80% delle immagini come training set. Il restante 20% è selezionato come test set. Il risultato della funzione **cvpartition** memorizzatelo nella struttura **cv**. La struttura di output contiene due metodi: **cv.training(n)** e **cv.test(n)**. I metodi ritornano gli n-esimi array colonna di booleani che indicano quali elementi appartengono al training set e quali al test set rispettivamente per la partizione dei dati n-esima (nel nostro caso ne abbiamo solo una).

C2. Usando gli array **cv.training(1)** e **cv.test(1)** e gli array dei descrittori, creiamo il training set e il test set da usare per la classificazione. Create due strutture **train** e **test**. Ciascuna struttura deve contenere i seguenti campi: **images** l'array che contiene la lista delle immagini; **labels** l'array che contiene la lista delle etichette delle immagini; **ghist** l'array che contiene l'istogramma dei livelli di grigio; **lbp** l'array dei descrittori lbp; e **glcm** l'array che contiene il descrittore Gray-Level Co- occurence Matrix. Ciascun campo deve riferirsi ai soli dati di training (**train**) e ai soli dati di test (**test**).

D2. Salvate in un file '**data2.mat**' le due variabili **train** e **test**.

Con cvpartition è possibile creare diverse partizioni dei dati per testare diverse configurazioni di training e test.

A3. In un nuovo script. Caricate il workspace **data2.mat** salvato in precedenza.

B3. Addestrate un classificatore cart sulle feature **ghist.** Cosa notate?

C3. Testate il classificatore sui dati di training (mette le etichette predette in **predicted_train**) e usate la funzione **confmat** per calcolare la sua performance (**performance_train**). Analizzate la matrice di confusione.

D3. Testate il classificatore sui dati di test (mette le etichette predette in **predicted_test**) e usate la funzione **confma**t per calcolare la sua performance (**performance_test**). Analizzate la matrice di confusione.

A4. Nello stesso script precedente, testate il classificatore baysiano e il classificatore knn. Analizzate le performance.

B4. Testate diverse combinazioni di descrittori.

Può essere comodo creare una funzione **eval_cart** (**eval_knn**, **eval_bayes**) per addestrare, e testare in fase di training e test il classificatore sui dati delle strutture **train** e **test**. La funzione potrebbe ritornare le accuratezze di training e test.

In questo modo è possibile valutare diversi classificatori e memorizzare le varie accuratezze per determinare quello più performante.