Classificazione di cellule epiteliali HEp-2 mediante l'utilizzo dei tensori di Fisher

Lorenzo Cioni

lore.cioni@gmail.com

18 Settembre 2015

Sommario

Analizzare e classificare le cellule epiteliali di tipo 2 (HEp-2) mediante l'utilizzo della tecnica della immunofluorescenza indiretta è uno standard per rilevare malattie al tessuto connettivo umano, come ad esempio l'Artrite Reumatoide. Purtroppo questo metodo è molto costoso in termini di tempo e particolarmente soggettivo.

Questo elaborato ha come finalità quella di implementare un metodo per la classificazione di questo tipo di cellule basato sull'utilizzo del descrittore di covarianza e dei tensori di Fisher per l'estrazione di *features* dalle immagini.

Indice

1 Introduzione

Una delle procedure standard per il rilevamento di malattie al tessuto connettivo umano, come ad esempio l'Artrite Reumatoide o il Lupus, è l'utilizzo di Immunofluorescenza Indiretta sulle cellule epiteliali di tipo 2, altrimenti conosciute come HEp-2.

Questo tipo di analisi ha due principali svantaggi: è molto soggettiva e richiede un gran numero di ore lavorative. Si è così pensato ad un metodo per automatizzare il processo per ottenere risultati migliori sia sotto il profilo medico che dal punto di vista di tempo impiegato.

Il metodo proposto e implementato è tratto da un articolo pubblicato in occasione del contest di *Pattern Recognition 2014* [1] [1]. Per la classificazione delle cellule si procede inizialmente all'estrazione di un adeguato numero di *features* attraverso l'utilizzo del *Descrittore di Covarianza* [2], vengono poi utilizzati i *Tensori di Fisher* che codificano informazioni addizionali rispetto alla distribuzione delle *features* ed infine le cellule vengono classificate tramite un SVM multiclasse.

I test per la valutazione della bontà del metodo sono stati effettuati sul dataset della competizione[2].

2 Tecniche utilizzate

Vengono ora presentate le tecniche utilizzate all'interno del programma: l'estrazione di *features* dall'immagine mediante l'utilizzo dei *filtri di Gabor* e il Descrittore di Covarianza (*Covariance Descriptor*).

2.1 Il filtro di Gabor

I filtri di Gabor[3] sono filtri passa-banda usati nell'analisi di immagini principalmente per l'estrazione di *features* e l'analisi basata sulla tessitura.

La risposta finita all'impulso di questi filtri è calcolata come prodotto di uno sviluppo Gaussiano con oscillazione complessa. Estendendo queste funzioni a due dimensioni è possibile creare filtri sensibili all'orientazione[4] e sotto certe condizioni è possibile approssimare linearmente la fase.

[1] ICPR Contest 2014 - HEp-2 Cells Classification Contest

[2] HEp-2 Dataset

[3] Gabor, D.: *Theory of communication.* In J. IEE, vol. 93, pp. 429-457, Londra, 1946.

[4] Daugman, J. G.: *Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters* J. Optical Society of America A, vol. 2, no. 7, pp. 1160-1169, July 1985.

Sia (x, y) un punto dell'immagine. L'equazione per il filtro di Gabor 2D è la seguente:

$$G(x, y) = e^{(-\frac{(x')^2+\gamma^2(y')^2}{2\sigma^2})} \cos(2\pi\frac{x'}{\lambda})$$

con

$$x' = x\cos\theta + y\sin\theta$$
$$y' = -x\cos\theta + y\sin\theta$$

I parametri del filtro sono:

- θ: orientazione del filtro, espressa in gradi.
- λ: lunghezza d'onda del fattore coseno, espressa in pixels.
- γ: specifica l'ellitticità del supporto della funzione di Gabor.
- σ: è il parametro che regola l'inviluppo Gaussiano.

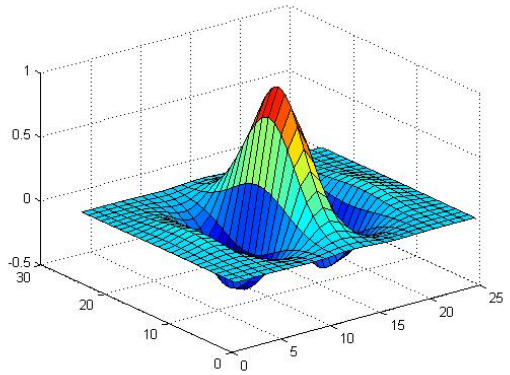


Figura 1: *Esempio di filtro di Gabor 2D*

Al fine di estrarre utili *features* da un'immagine è utile utilizzare un set di filtri di Gabor con parametri diversi, ad esempio la scala e l'orientazione.

Nel programma viene utilizzato un set di filtri di Gabor con 4 diverse angolazioni e 3 diverse scale per un totale di 12 filtri:

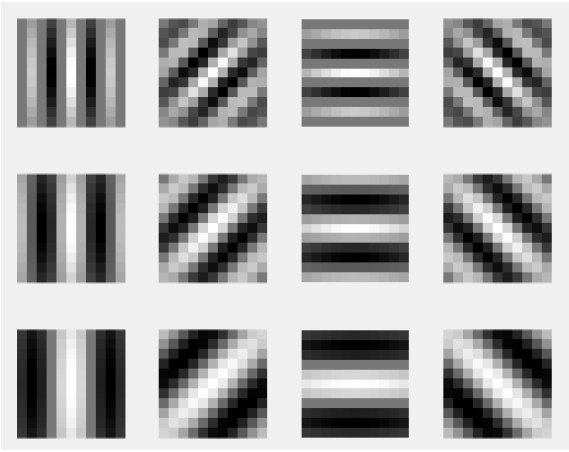


Figura 2: *Set di filtri di Gabor con 4 angolazioni e 3 scale*

Per ciascun pixel dell'immagine viene dunque calcolato il vettore delle *features* ottenute mediante la convoluzione dell'immagine con i filtri di Gabor (dividendo la parte reale dalla parte immaginaria).

2.2 Descrittore di Covarianza

Il Descrittore di Covarianza (Covariance Descriptor[5], CovD) viene utilizzato in letteratura per descrivere e caratterizzare regioni di immagini.

Data un'immagine con associato, a ciascun pixel, un vettore di *features*, il Descrittore di Covarianza codifica informazioni che riguardano le *varianze* di queste ultime all'interno di una regione, la loro correlazione e la loro distribuzione nello spazio. Questo metodo è molto utilizzato in particolare modo quando si ha bisogno di rimanere invarianti rispetto all'illuminazione o alla rotazione.

Data un'immagine I con associato, per ciascun pixel (x, y), il vettore di features di dimensione d, si considera F come l'immagine di features a d livelli. Viene dunque considerata una regione di F tale che $R \subset F$ e consideriamo $\{f_i\}_{i=1,\dots,n} \in R$ la i-esima features, con n numero di pixels della regione.

A questo punto la regione R viene rappresentata da una matrice di covarianza di dimensione dXd dei *features points*:

$$C = \frac{1}{n-1} \sum_{k=1}^{n} (f_k - \mu)(f_k - \mu)^T$$

con μ la media dei *features points*.

Poichè la matrice C è definita positiva e simmetrica può essere considerata un *tensore*. Il problema principale è che lo spazio del tensore così definito non è uno spazio Euclideo. Per questo la varietà della covarianza viene spesso definita Riemanniana per determinare un insieme di strumenti efficaci della geometria differenziale.

[5]Oncel Tuzel, Fatih Porikli, Peter Meer, *Region Covariance: A Fast Descriptor for Detection and Classification* Mitsubishi Electric Research Laboratories, Inc., 2006.

2.3 Tensori di Fisher

I *tensori di Fisher* rappresentano la controparte definita su varietà Riemanniana dei *vettori di Fisher*.

Il vettore di Fisher (FV) è una rappresentazione di una immagine ottenuta raccogliendo osservazioni locali dall'immagine stessa. E' usato comunemente come un descrittore globale di immagini nella classificazione.

Sia $I = (x_1, \ldots, x_N)$ un insieme composto da N vettori di D osservazioni (*features vectors*) estratte dall'immagine. Siano poi $G = (\mu_k, \Sigma_k, \pi_k; k = 1, \ldots, K)$ i parametri di un Modello a Mistura di Gaussiane (GMM) opportuno per la distribuzione dei descrittori.

Il GMM associa ciascun vettore x_i a uno stato k:

$$q_{ik} = \frac{e^{-\frac{1}{2}(x_i-\mu_k)^T\Sigma_k^{-1}(x_i-\mu_k)}}{\sum_{t=1}^{K} e^{-\frac{1}{2}(x_i-\mu_t)^T\Sigma_k^{-1}(x_i-\mu_t)}}$$

Per ciascuna moda k consideriamo il vettore media ed il vettore di covarianza della deviazione:

$$u_{ik} = \frac{1}{N\sqrt{\pi_k}} \sum_{i=1}^{N} q_{ik} \frac{x_{ji} - \mu_{jk}}{\sigma_{jk}}$$

$$v_{jk} = \frac{1}{N\sqrt{2\pi_k}} \sum_{i=1}^{N} q_{ik} [\frac{x_{ji} - \mu_{jk}}{\sigma_{jk}}^2 - 1]$$

con $j = 1, \ldots, D$.

Il FV di un immagine I è la concatenazione dei vettori u_k e v_k per ciascuno dei K stati del GMM

$$FV(I) = \begin{bmatrix} u_1 \\ \vdots \\ u_K \\ v_1 \\ \vdots \\ v_K \end{bmatrix}$$

3 Algoritmo

Utilizzando le tecniche descritte nella sezione precedente è stato ideato l'algoritmo descritto in [1].

L'algoritmo può essere suddiviso in 4 parti fondamentali:

1. Creazione dei filtri di Gabor
2. Estrazione di features
3. Creazione di un modello a mistura di gaussiane

4. Calcolo dei *tensori di fisher*

5. Classificazione

Inizialmente viene creato il set di filtri di Gabor, impostando 4 diverse orientazioni e 3 diverse scale. Si ottengono così 12 filtri distinti che verranno utilizzati nella fase di estrazione delle *features*.

3.1 Estrazione delle *features*

In questa fase ciascuna scansione viene elaborata al fine di estrarre delle *features*.

Inizialmente, data un immagine viene calcolata la sua maschera, ovvero un'immagine binaria che evidenzia i pixels di *foreground*. In questo modo il calcolo delle *features* verrà effettuato solamente sui pixels rilevanti dell'immagine, escludendo lo sfondo. La maschera viene calcolata mediante un'operazione di sogliatura tramite il metodo di Otsu.

L'immagine viene scansionata a blocchi quadrati di dimensione 80 pixels, sovrapposti con un passo di 20 pixels.

Blocco Maschera

Figura 3: Esempio di blocco e maschera di elaborazione

A questo punto, considerando solo i pixel del blocco filtrati dalla maschera, per ciascun pixel viene estratto un vettore di 15 osservazioni così composto:

$$F_{x,y} = [I(x,y), x, y, |G_{0,0}(x,y)|, \ldots, |G_{0,v}(x,y)|, |G_{1,0}(x,y)|, \ldots, |G_{u,v}(x,y)|]$$

dove

- $I(x,y)$ è il livello di grigio del pixel alla posizione (x,y).
- x, y solo rispettivamente l'ascissa e l'ordinata del pixel considerato.
- u è il numero delle scale del filtro di Gabor (in questo caso 3).
- v è il numero delle orientazioni del filtro di Gabor (in questo caso 4).

A questo punto a ciascun pixel del blocco è associato un vettore di 15 osservazioni, dunque la matrice delle *features* avrà dimensione 15 X n^2, con n dimensione del blocco (in questo caso 80). Si procede dunque con il calcolo del *descrittore di covarianza* del blocco, ottenendo una matrice C di dimensione 15 X 15.

Gli elementi della diagonale di questa matrice rappresentano le varianze di ciascuna osservazione e gli elementi extra diagonali rappresentano le correlazioni.

In ultima istanza viene effettuato il *mapping* sullo spazio tangente per ricondurci alla geometria Riemanniana.

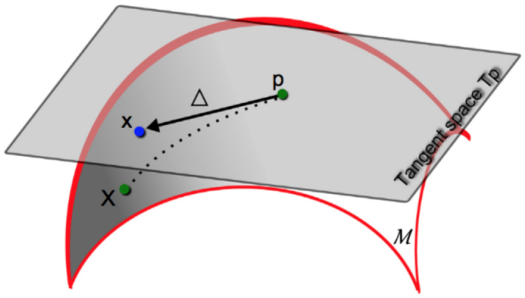


Figura 4: *Illustrazione dello spazio tangente T_P al punto P sulla varietà Riemanniana $\mathcal{M}$*

Il vettore tangente Δ può essere ottenuto mediante mappa logaritmica, $\Delta = \log(C)$.

3.2 Creazione di un modello a mistura di gaussiane

A partire dalle *features* estratte in precedenza si procede ora alla creazione di un modello a mistura di gaussiane. Viene fissato un valore di K, nel nostro caso 16, un valore di *tolleranza* e un un numero massimo di *iterazioni*.

Il modello così creato verrà utilizzato per il calcolo dei tensori di Fisher.

3.3 Calcolo dei *tensori di Fisher*

Per ciascuna immagine vengono ora calcolati i *tensori di Fisher*. Dati in ingresso:

- $\mathbf{Q} = \{Q_t\}_{t=1}^{p}$ descrittori di covarianza estratti da una immagine (p numero di blocchi).
- $\lambda_i = \{\omega_i, \mu_i, \Sigma_i\}$, componenti della mistura di gaussiane, con $i = 1, \ldots, K$

Sia $p(x|\lambda)$ una distribuzione di probabilità modellata da una GMM con K gaussiane

$$p(x|\lambda) = \sum_{i=1}^{K} \omega_i g(x|\mu_i, \sum_i)$$

Algoritmo 1 Tensore di Fisher di $\mathbf{Q}$

- Calcolo della rappresentazione logaritmica euclidea di $\mathbf{Q}$, $q_t = Vec(\log(\mathbf{Q}))$
- $\gamma_i(q_t) = \frac{\omega_i g(q_t|\mu_i, \sum_i)}{\sum_{j=1}^{K} \omega_j g(q_t|\mu_j, \sum_j)}$

for $i = 1, \ldots, K$ **do**

$\mathcal{G}_i = \frac{1}{p\sqrt{\omega_i}} \sum_{t=1}^{p} \gamma_i(q_t)\sigma_i^{-1}(q_t - \mu_i)$ con $\sum_i = \sigma_i^2$

end for

- $FT(\mathbf{Q}) = [\mathcal{G}_1^T, \ldots, \mathcal{G}_K^T]$

3.4 Classificazione

Infine si procede con la classificazione, affidata al classificatore SVM.

Viene creato il modello multiclasse con cross-validazione basato su kernel gaussiano, suddividendo il dataset in 8 parti uguali.

Il modello viene poi valutato andando a predire l'intero dataset.

4 Dataset

Per la valutazione dell'efficienza dell'algoritmo implementato è stato utilizzato una parte del dataset dell'*ICPR HEp-2 Cell Classification Contest*.

Il dataset contiene al suo interno 149 immagini che rappresentano le scansioni di vetrini di laboratorio. All'interno di queste immagini è possibile distinguere 9 diversi pattern: *punteggiato*, *nucleolare*, *citoplasmico*, *omogeneo*, *granulare*, *negativo* e *centromero*. Alcune immagini del dataset sono poi classificate come *altro*, rendendo difficile una loro classificazione, questi elementi sono stati dunque non considerati. A causa del basso numero di elementi classificati con *centromero* anche questi sono stati rimossi fino ad ottenere un totale di 137 immagini.

Le immagini sono state acquisite tramite un microscopio a fluorescenza (40 ingrandimenti) in combinazione con una lampada di mercurio vaporizzato a 50W e una fotocamera digitale SLIM con risoluzione 1920 x 1110.

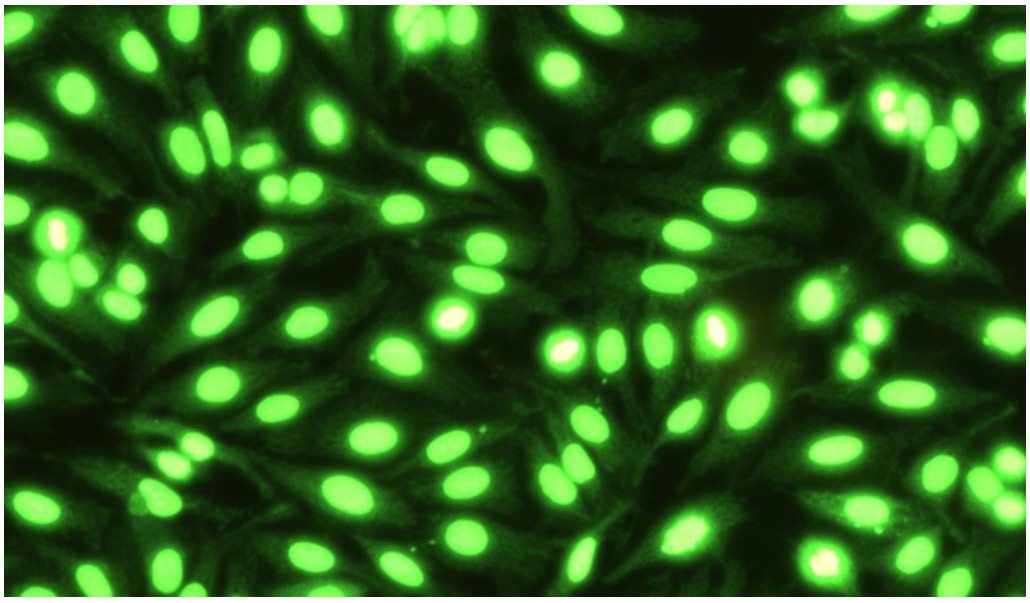
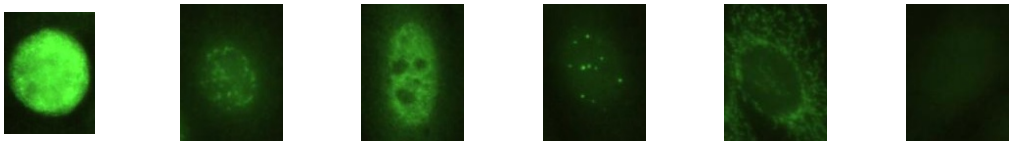


Figura 5: *Esempio di immagine del dataset*

Ciascuna immagine è stata poi annotata da un medico specialista ed associata ad una delle classi di pattern sopra esposte. Le annotazioni sono state inserite in una tabella così da poterne ricavare un *training set*.



Omogeneo Nucleolare Punteggiato Granulare Citoplasmico Negativo

Figura 6: Esempi di cellule classificate nel dataset

5 Risultati

In questa sezione vengono illustrati e documentati i risultati ottenuti mediante questo procedimento.

I test sono stati effettuati considerando l'intero dataset di immagini e utilizzando la tecnica della classificazione con *cross-validazione*. La cross-validazione (*cross-validation* in inglese) è una tecnica statistica che consiste nel suddividere in k parti equinumerose il dataset, *k-fold cross-validation*, e, ad ogni passo, la parte $\frac{1}{k}$-esima del dataset viene ad essere il validation dataset, mentre la restante parte costituisce il training dataset.

Così, per ognuna delle k parti (in questo caso $k = 8$) si allena il modello, evitando quindi problemi di *overfitting*, ma anche di campionamento asimmetrico del training dataset, tipico della suddivisione del dataset in due sole parti (ovvero training e validation dataset). In altre parole, si suddivide il campione osservato in gruppi di egual numerosità, si esclude iterativamente un gruppo alla volta e lo si cerca di predire con i gruppi non esclusi. Ciò al fine di verificare la bontà del modello di predizione utilizzato.

Una volta costruito il modello tramite questa tecnica, ne viene valutata l'efficacia: tutto il dataset viene valutato e si ottengono così i risultati di corretta o errata classificazione.

Il valore di K per la creazione del modello a mistura di gaussiane scelto è 16, valore con cui è stato ottenuto il punteggio più alto di accuratezza finale.

Nella Tabella 1 vengono raccolti i dati ottenuti dalla valutazione del dataset tramite il modello.

Classe	**Totale**	**Corretti**	**Percentuale**
Punteggiato	49	38	77.55
Nucleolare	11	1	9.09
Citoplasmico	15	10	66.67
Negativo	33	30	90.91
Omogeneo	20	12	60.00
Granulare	9	2	22.22

Tabella 1: Risultati

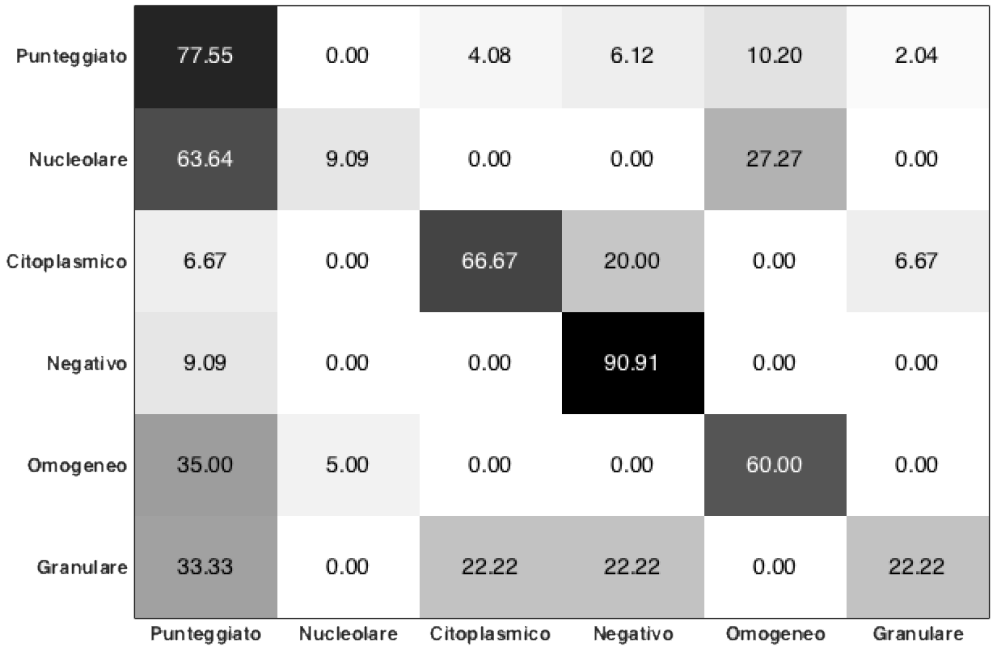


Figura 7: *Matrice di confusione*

Il miglior risultato ottenuto ha classificato correttamente 93 scansioni su 137, con un accuratezza totale del 67.88 %.

Da una prima analisi sui risultati ottenuti si evince che il pattern *punteggiato* è quello che ha migliore probabilità di essere correttamente classificato. Questo può dipendere anche dal fatto che è la classe con il maggior numero di esemplari e dunque il modello costruito è tendenzialmente più preciso.

Un ottimo risultato si ottiene anche per quanto riguarda il pattern *negativo* che corrisponde a quelle cellule dove il marcatore fluorescente non emette luce visibile.

I pattern *nucleolare* e *granulare* sono quelli che hanno ottenuto risultati peggiori. Il *nucleolare*, ad esempio, viene riconosciuto correttamente una volta soltanto, confondendolo in buona parte con il *punteggiato* o con l'*omogeneo*, mentre il pattern *granulare* viene confuso quasi in modo equo fra tutte le altre classi. Questo può in parte dipendere da un minor numero di campioni su cui creare il modello ma anche da una varietà di cellule all'interno della stessa classe.

6 Implementazione

L'implementazione dell'algoritmo qui presentato è stata sviluppata interamente in codice Matlab, compatibile con la versione 2014a o successive. Per una corretta esecuzione è necessario avere installato alcuni pacchetti aggiuntivi:

- *Image Acquisition Toolbox*: nesessario per l'elaborazione di immagini.
- *Image Processing Toolbox*: necessario per l'elaborazione di immagini.

- *Statistics and Machine Learning Toolbox*: necessario per la creazione del modello per la classificazione SVM e per la creazione di un modello a mistura di gaussiane.

Per la creazione del *train set* di partenza viene utilizzata una funzione Matlab per il *parsing* di valori da file di validazione. In caso di diverse necessità sarà sufficiente modificare questa funzione senza interferire con il resto dell'esecuzione.

6.1 Configurazioni iniziali

Le configurazioni sono tutte racchiuse all'interno di un'unica classe, così da poter essere facilmente adattabile all'esecuzione su macchine diverse.

Il file **configuration.m** contiene al suo interno le configurazioni di base del progetto: è necessario modificarle prima di proseguire con l'esecuzione del programma.

- **image_path**: cartella dove si trovano le immagini da analizzare.
- **image_prefix**: prefisso del nome dei file.
- **image_ext**: estensione dei file.
- **validation_format**: formato del file di validazione. Può essere *xls*, *xlsx*, *csv*. File nel formato *xls* potrebbero non essere letti correttamente dal programma nel caso di esecuzione su sistemi Unix.
- **validation_file_worksheet_name**: nome del foglio di lavoro utilizzato nel file di validazione.
- **validation_file**: file per la creazione del *training set*.
- **validation_file_image_ids_column**: indice di colonna della tabella in cui è presente l'id dell'immagine.
- **validation_file_image_label_column**: indice di colonna della tabella che contiene la *label* assegnata all'immagine.
- **patterns**: mappa le varie classi con gli identificativi del file di validazione.
- **Gabor_options**: impostazioni utilizzate per la creazione del banco di filtri di Gabor.
- **block_size**: dimensione della finestra di scansione dell'immagine (in pixels).
- **delta**: passo (in pixels) di traslazione della finestra.
- **resize**: *true/false*, se vero le immagini verranno ridimensionate. Permette un'esecuzione più rapida del programma, ma viene meno l'accuratezza finale.
- **resizeTo**: imposta la dimensione alla quale ridimensionare l'immagine (se impostato il *resize* a *true*).
- **K**: numero di gaussiane per la creazione del modello a mistura.
- **kFolds**: numero di suddivisioni per la cross-validazione.

6.2 Esecuzione

L'algoritmo, come descritto nella Sezione 3 è stato suddiviso in 5 fasi fondamentali, da eseguire in ordine.

1. **loadTrainingSet.m**: si occupa della creazione del *training set*. Legge il file di validazione e va a creare una struttura contenente:
   - ID dell'immagine.
   - *Label* assegnata.
   - Nome completo del file e cartella.
2. **extractImages.m**: esegue quanto descritto nella Sezione 3.1.
3. **runGMM.m**: esegue quanto descritto nella Sezione 3.2.
4. **saveSignatures.m**: calcola i tensori di Fisher nel modo descritto nella Sezione 3.3.
5. **runSVM.m**: esegue la classificazione. Viene stampata la tabella dei risultati ottenuti e mostrata una versione grafica della matrice di confusione (generata tramite la funzione **plotConfusionMatrix.m**).

7 Sviluppi successivi

Al fine di ottenere migliori risultati nella fase di classificazione, sono state apportate alcune modifiche all'elaborato e considerato un modo migliore di trattare il dataset a disposizione.

7.1 Analisi

L'analisi effettuata in precedenza evidenziava che, trattando le immagini nella loro interezza (seppure con la possibilità di una variazione di scala), alcuni dei *pattern* fossero difficilmente classificabili. Questo può dipendere principalmente da diversi fattori, primo fra tutti la grande varietà di cellule presenti all'interno di uno stesso vetrino: subbene infatti fossero etichettati come un'unica entità, non tutte le cellule presenti si mostravano allo stesso modo, evidenziando in alcuni punti pattern ben diversi da quello di riferimento (vedi Figura 8). Questo fatto porta a *confondere* il classificatore che vedrà come pattern possibili più di un unico risultato.

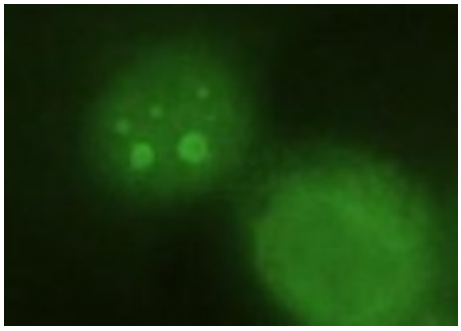


Figura 8: *Cellule con diverso pattern nella stessa immagine*

In aggiunta al problema sopraelencato i risultati evidenziano una particolare difficoltà nella classificazione di alcuni pattern: *nucleolare*, *granulare* e *citoplasmico* (Tabella 1). Questo può dipendere dalla loro effettiva somiglianza visiva, dalla bassa risoluzione con cui vengono trattate le immagini (in caso di scala) oppure da un non adatto modello di features usato per la loro descrizione.

7.2 Miglioramento del dataset

Date le problematiche evidenziate, una possibile soluzione era quella di utilizzare un dataset in cui ciascuna cellula, e non più l'intero vetrino, fosse etichettata singolarmente. In questo caso, nella fase di estrazione delle features, è possibile mantenendo accettabile il tempo di computazione necessario, provvedere a ingrandire le immagini così da consentire una miglior distinzione tra pattern singoli.

Il dataset è dunque ora composto da un totale di 1455 immagini di cellule singolarmente etichettate in 6 diversi pattern:

- Omogeneo
- Punteggiato
- Finemente punteggiato
- Centromero
- Nucleolare
- Citoplasmico

Come si nota, a differenza del precedente esperimento, in questo caso il pattern *punteggiato* è stato diviso nelle due sue forme più comuni ed è stato eliminato il pattern *negativo*, non rilevante ai fini della classificazione in quanto completamente (o quasi) nero.

7.2.1 Risultati

I risultati ottenuti con queste modifiche sono molto migliori dei precedenti, andando a migliorare di quasi dieci punti percentuali l'accuratezza ottenuta in precedenza.

I risultati sono ottenuti cross-validando il dataset in 10 sottoparti. La classificazione è fatta tramite SVM, utilizzando però la libreria open-source *libSVM*[6] alla versione 3.2. L'utilizzo di questa libreria ha consentito di poter meglio personalizzare i parametri del classificatore e presenta una migliorazione nei tempi di computazione rispetto alla versione implementata nativamente in Matlab.

I risultati ottenuti vengono presentati nella matrice di confusione, Figura 9 e nella Tabella 2.

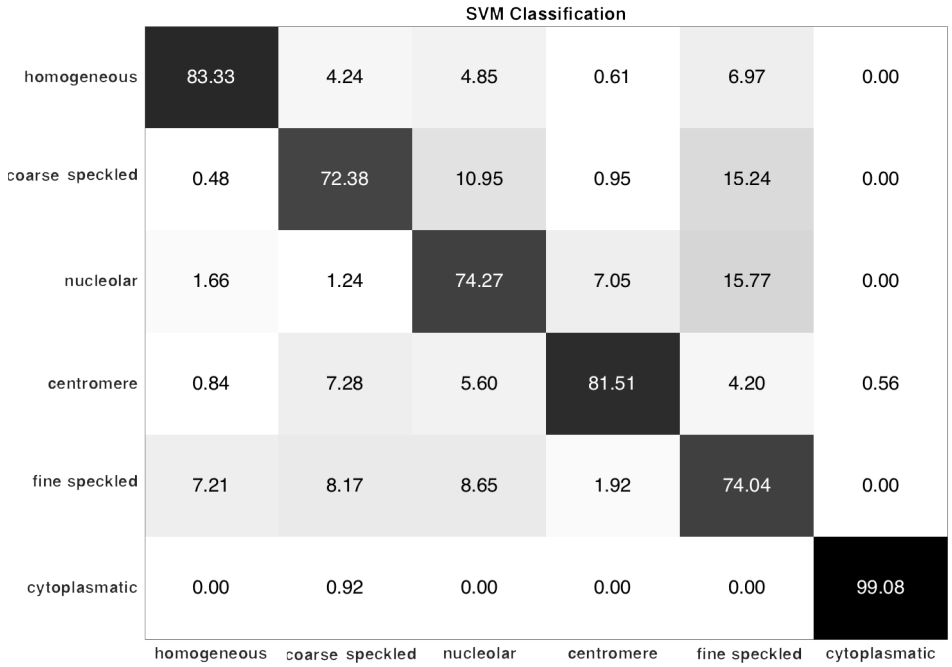


Figura 9: *Matrice di confusione*

Classe	**Totale**	**Corretti**	**Percentuale**
Omogeneo	330	275	83.33
Punteggiato	210	152	72.38
Nucleolare	241	179	74.27
Centromero	357	291	81.51
Fin. Punteggiato	208	154	74.04
Citoplasmico	109	108	99.08

Tabella 2: Risultati

Il miglior risultato ottenuto ha classificato correttamente 1159 cellule su 1455, con un accuratezza totale del 79.66 %.

Particolari miglioramenti si hanno per quanto riguarda la classificazione di cellule con pattern *citoplasmico* e *omogeneo*.

[6]LibSVM, *A Library for Support Vector Machines*, https://www.csie.ntu.edu.tw/ cjlin/libsvm/

8 Conclusioni

La tecnica utilizzata per la classificazione di cellule epiteliali HEp-2 ha dato risultati positivi sul dataset considerato.

Buona parte, circa il 70%, delle scansioni sono state correttamente classificate, in particolare la tecnica è molto buona per scartare i campioni negativi, riconosciuti con un'alta accuratezza.

Alcuni dei pattern sono però difficilmente classificabili: questo a causa anche del basso numero di campioni di training a disposizione che non consentono così la creazione di un modello adeguato.

Un possibile miglioramento potrebbe essere ottenibile aumentando il numero di filtri di Gabor utilizzati, andando a estrarre un numero maggiore di osservazioni dall'immagine e rendendo più accurata la descrizione delle singole classi, andando però ad aumentare la durata di computazione.

In conclusione il metodo presentato in [1] e qui implementato è una buona soluzione al problema di classificazione di cellule, migliorabile nella fase di estrazione delle *features* e nella scelta dei parametri.

Riferimenti bibliografici

[1] Masoud Faraki, Mehrtash T. Harandi, Arnold Wiliem, Brian C. Lovell, *Fisher tensors for classifying human epithelial cells.* Pattern Recognition, Volume 47, 2014, pp. 2348 - 2359.

[2] Oncel Tuzel, Fatih Porikli, Peter Meer, *Region Covariance: A Fast Descriptor for Detection and Classification.* Mitsubishi Electric Research Laboratories, Inc., 2006.

[3] Gabor, D.: *Theory of communication* In J. IEE, vol. 93, pp. 429-457, London, 1946.