Università degli Studi di Modena e Reggio Emilia

Dipartimento di Scienze Fisiche, Informatiche e Matematiche

Corso di Laurea Magistrale in Matematica

UNIMORE

UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

Relatrice:
Prof.ssa Giorgia Franchini

Laureanda:
Alice Bonaretti

Correlatori:
Dott. Matteo Lombardi
Dott.ssa Valeria Trojani

Anno Accademico 2024/2025

Abstract

La presente tesi si inserisce nel contesto della digital pathology e ha come obiettivo lo sviluppo e la valutazione di metodologie basate su deep learning per l'analisi di immagini istologiche. Il lavoro si focalizza principalmente sul problema della segmentazione dei nuclei cellulari, considerata un passaggio fondamentale per l'estrazione di informazioni morfologiche rilevanti e per l'analisi quantitativa del tessuto.

Dopo una revisione dei principali concetti di machine learning, deep learning e reti neurali convoluzionali, vengono analizzati i metodi di segmentazione di immagini, con particolare attenzione agli approcci basati su architetture deep learning. Viene quindi condotto un confronto tra modelli di stato dell'arte, tra cui StarDist e Cellpose, valutati su diversi dataset istologici, al fine di individuare la soluzione più adatta al contesto applicativo.

Nell'ambito di un tirocinio svolto presso l'AUSL di Reggio Emilia, il lavoro è stato esteso allo sviluppo di un modello di classificazione di vetrini istologici. A partire dalle segmentazioni ottenute dalle patch in cui è stato suddiviso ogni vetrino, sono state estratte feature morfologiche a livello cellulare, successivamente aggregate a livello di patch per rappresentare la variabilità strutturale del tessuto.

Per la classificazione è stato adottato un approccio di Multiple Instance Learning basato sul metodo Positive Instance Sampling (PINS), opportunamente adattato al problema in esame. Il modello integra informazioni visive e morfologiche e sfrutta strategie di campionamento adattivo delle istanze al fine di classificare i vetrini a partire dalle regioni più informative.

Le prestazioni del modello sono state valutate mediante validazione incrociata, considerando diverse metriche di classificazione e analizzando la variabilità dei risultati in relazione alle caratteristiche del dataset.

Indice

Capitolo 1

Deep learning e CNN

Negli ultimi decenni l'*intelligenza artificiale* (AI) è diventata molto popolare all'interno delle comunità scientifiche. Spesso il suo nome viene associato ai concetti di *machine learning* (ML) e di *deep learning* (DL). I tre termini sono altamente connessi ma non sono sinonimi.

Nel 1956 alcuni informatici teorizzarono che si potesse programmare il computer per "pensare", cioè che ogni aspetto dell'apprendimento e dell'intelligenza potesse essere descritto in maniera così precisa da permettere ad una macchina di simularlo. Questo principio viene appunto chiamato intelligenza artificiale.

L'AI include diversi approcci, tra questi ci sono il ML e il DL (Fig. 1.1). Questi ultimi però non esauriscono tutte le applicazioni possibili dell'AI, infatti esistono approcci non legati all'apprendimento come per esempio quello simbolico: in questo caso si descrive alla macchina esattamente come procedere in ogni scenario, appoggiandosi ad una conoscenza a priori del problema. Un esempio di quest'ultimo approccio è la regolazione della temperatura di una stanza: si conosce ad ogni ora la temperatura ottimale, va solo raggiunta in caso sia diversa da quella attuale.

Per tasks più complesse, come il riconoscere pattern ad alto livello o analizzare immagini, l'approccio simbolico risulta poco efficace: per questo vengono preferiti il ML e il DL.

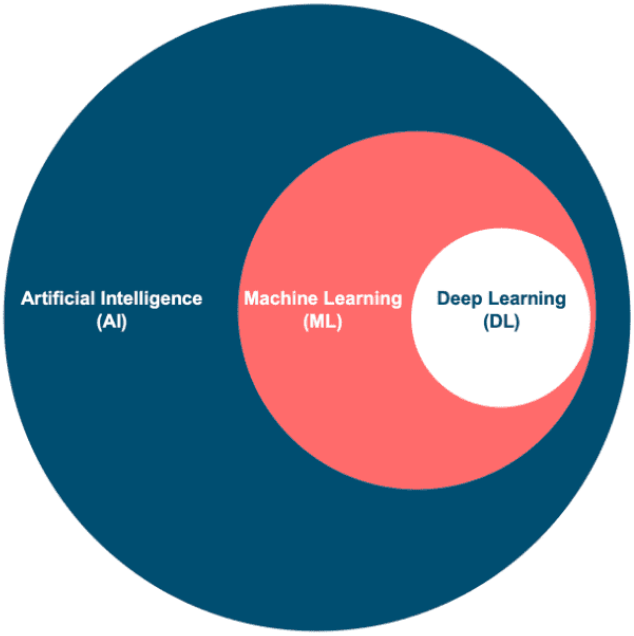


Figura 1.1: Relazione tra AI, ML e DL

1.1 Machine learning

Il Machine Learning è un approccio dell'AI focalizzato sull'apprendimento. A differenza della programmazione classica, in cui un algoritmo può essere codificato esplicitamente usando feature note, nel ML la struttura dell'algoritmo viene *imparata* a partire da un sottoinsieme dei dati a disposizione.
Il metodo con cui si crea la struttura dell'algoritmo dipende dalla tipologia del problema:

1. **Apprendimento supervisionato**: A partire da un dataset costituito da coppie feature-label, in cui le feature sono i dati da fornire all'algoritmo e le label sono ciò che deve prevedere, si creano, in genere, tre sottoinsiemi: training set, validation set e test set.

   Il training e validation set vengono usati per addestrare i parametri dell'algoritmo mentre il test set viene usato per valutare le performance; le metriche usate sono diverse ma l'obiettivo è sempre confrontare le labels del test set, predette tramite algoritmo e feature, con quelle originali del dataset.

   In ambito supervisionato si parla di classificazione quando le labels appartengono ad un insieme finito di valori mentre si parla di regressione quando le labels appartengono ad un intervallo continuo.

2. **Apprendimento non supervisionato**: In questo caso il dataset non contiene labels, l'obiettivo è quindi quello di riconoscere pattern all'interno dei dati. Uno dei modelli più usati è il *clustering* che mira a raggruppare le istanze in diversi gruppi basandosi su combinazioni specifiche delle feature.

3. **Apprendimento semisupervisionato**: Il dataset contiene dati con e senza labels, ponendosi quindi a metà strada tra i due precedenti. Spesso questo è il caso di dataset medici, in cui sarebbe proibitivo classificare tutte le immagini. Le immagini con label verranno usate per addestrare un modello, usato successivamente per prevedere le labels mancanti. Il dataset completo potrà poi essere usato per creare il modello definitivo, ottenendo comunque risultati migliori di quelli raggiungibili con un approccio non supervisionato.

4. **Apprendimento con rinforzo**: In quest'ultimo caso l'obiettivo non è un modello che classifichi correttamente una singola istanza ma il risultato complessivo. Come nell'apprendimento umano, il modello apprende tramite ricompense. Per esempio, nel gioco Mario Bros, l'obiettivo è che Mario dalla partenza arrivi alla bandierina finale, non conta la singola mossa ma la sequenza complessiva che porta a vincere o perdere. L'algoritmo è quindi libero di provare sequenze diverse e verrà premiato in caso di successo, in modo da imparare quali siano le *mosse* migliori da compiere in ogni situazione.

1.2 Reti neurali e Deep Learning

1.2.1 Reti neurali (ANN)

Una rete neurale (Artificial Neural Network - ANN) è un algoritmo di ML di ispirazione biologica, l'unità base è infatti ispirata ad un neurone che è connesso ad altri neuroni della rete. Le connessioni tra i nodi della rete sono pesate per ottenere l'output desiderato. Ci sono prove del fatto che i neuroni, lavorando insieme, siano in grado di imparare relazioni input-output complesse o non lineari tramite una procedura sequenziale di training.

Il percettrone è l'unità base della rete, è un algoritmo che simula la rete più semplice

che è quella costituita da un solo neurone. Le altre reti sono una generalizzazione di quest'ultima.
Considerando un percettrone (Fig. 1.2), siano $x_1, ...x_m$ l'informazione che il neurone riceve dall'esterno e $\mathbf{w} = (w_1, ..., w_m)$ il vettore dei pesi che simula le connessioni sinaptiche. I valori che il neurone riceve sono modificati dalle connessioni sinaptiche, quindi l'input netto che lo raggiunge è dato da

$$v = \sum_{j=1}^{m} w_j x_j$$

L'input netto è ciò che determina se un neurone sia attivato o no, l'output infatti si ottiene tramite una funzione di attivazione che è funzione di v:

$$y = \phi(v)$$

La più classica è la *funzione a soglia* per cui il neurone si attiva se l'input è maggiore di una costante fissata, in tal caso l'output sarà 1 altrimenti 0. Esistono poi funzioni di attivazione più complesse e non lineari.

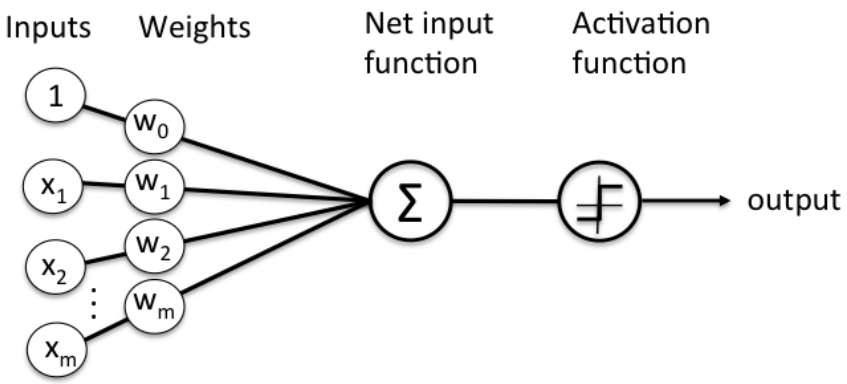


Figura 1.2: Schema del percettrone

Nel caso delle funzioni di attivazione non ci sono corrispondenze biologiche ma ciò permette alle ANN di avere molte applicazioni.

1.2.2 Deep Learning

Un insieme di neuroni che riceve simultaneamente lo stesso tipo di informazioni viene detto *layer*. Le ANN con un solo layer hanno una bassa capacità di processazione: il loro potere infatti dipende dalla complessità della struttura. Il modello sopra presentato si generalizza considerando che ogni neurone verrà attivato o meno in base ad input che possono provenire anche da altri neuroni della rete.

Si parla di Deep Learning (DL) quando si considerano reti composte da più strati nascosti (hidden layers) quelli cioè che si pongono tra gli strati di input e output. In questo caso si parla anche di Deep Neural Network (DNN). L'aggettivo *deep* non si riferisce alla profondità della conoscenza acquisita bensì al modo in cui la conoscenza viene acquisita, la *profondità* di una rete è infatti il numero di layers che la compongono (tolto quello di input). La *dimensione* della rete è invece data dal numero di neuroni che la compongono. Nel seguito si considereranno sempre reti di tipo *feedforward* che sono le reti in cui non sono presenti cicli, l'informazione cioè si muove sempre nella stessa direzione, senza connessioni di feedback.

In figura 1.3 è riportata una rete neurale DNN. Si può notare che, in generale, una rete è un grafo diretto in cui i neuroni sono i nodi e gli archi le connessioni.

Questa rete ha profondità 3 e dimensione $|V| = \left|\bigcup_{t=0}^{3} V_t\right| = |9 + 4 + 4 + 4| = 21$. Da notare che in ogni layer è stato aggiunto un nodo $(V_{0,9}, V_{1,4}, V_{2,4})$ che aggiunge 1 alle unità osservate e rappresenta l'intercetta o bias.

Generalizzando quanto fatto per il percettrone si può scrivere la forma analitica del modello:

$$V_{1j} = g_1\left(\sum_{i=1}^{9} w_{ji}^{(1)} x_i\right) \quad j = 1, ...3$$

$$V_{2k} = g_2\left(\sum_{j=1}^{4} w_{kj}^{(2)} V_{1j}\right) \quad k = 1, ...3$$

$$y_l = g_3\left(\sum_{l=1}^{4} w_{lk}^{(3)} V_{2k}\right) \quad l = 1, ...4$$

Dove la prima equazione rappresenta l'ouput del primo layer, la seconda del secondo e infine la terza l'output della rete. Il processo di apprendimento mira a stabilire i pesi $w_{ij}^{(n)}$ della rete in modo tale da fissare tutti i parametri del modello.

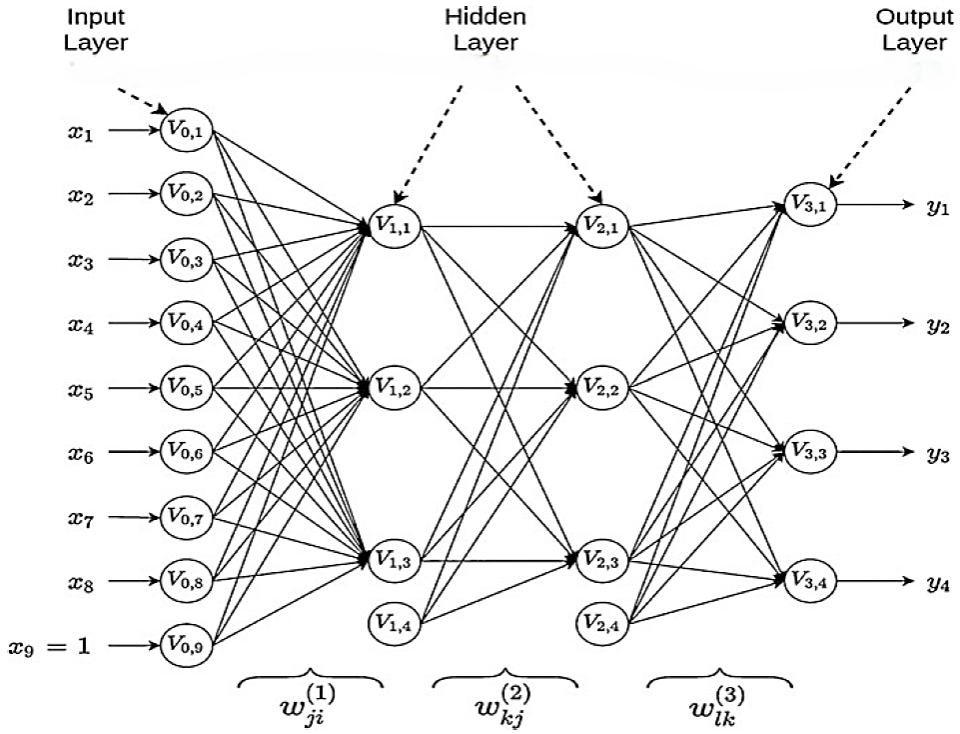


Figura 1.3: Rete Feedforward con otto variabili di input, quattro di output e due hidden layers con tre neuroni ciascuno

1.2.3 Funzioni di attivazione

g_1, g_2 e g_3 sono le funzioni di attivazione che decidono l'output dei neuroni di ogni strato e permettono di introdurre una non linearità nella rete.

Esistono diversi tipi di funzioni di attivazione, tra le più usate ci sono

1. **Lineare**: $g(z) = Wz$

   La proporzionalità è diretta (Fig. 1.4) quindi il segnale passa pressochè inalterato, a meno del coefficiente angolare della retta. Il problema con questo tipo di funzione è che non permette alla rete di apprendere funzioni non lineari.

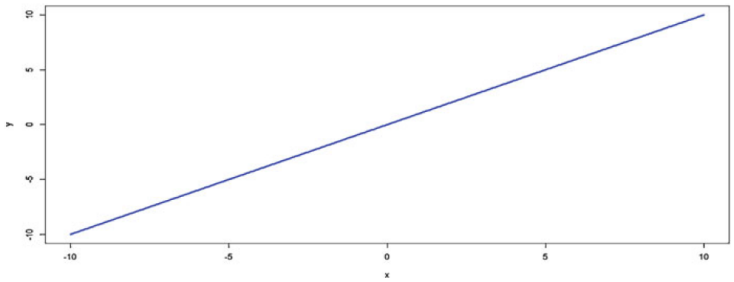


Figura 1.4: Funzione di attivazione lineare

2. **Sigmoide**: $g(z) = \frac{1}{1+e^{-z}}$

   La sigmoide converte la variabile indipendente in probabilità compresa tra 0 e 1 (Fig. 1.5). Come la trasformazione logistica, controlla gli outlier senza eliminarli. La sigmoide è la funzione più utilizzata per restituire una probabilità. Fornisce un buon bilanciamento tra comportamento lineare e non. Il rischio maggiore è quello di non progredire nell'addestramento quando gli output hanno valori estremi: i valori della sigmoide saranno sempre vicini a 0 o 1.

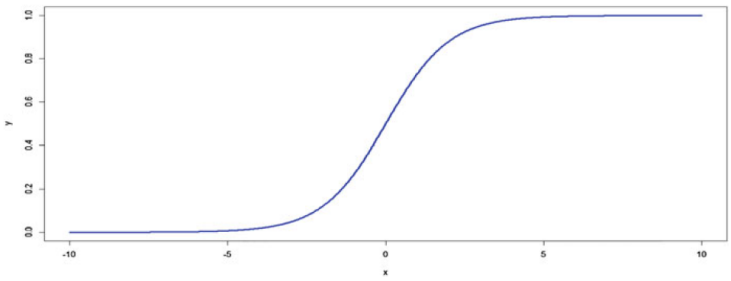


Figura 1.5: Funzione di attivazione sigmoidale

3. **ReLU** (Rectifier linear unit): $g(z) = max(\theta, z)$

   La ReLu è la funzione più usata, il segnale non passa sotto una certa soglia θ, che spesso è pari a 0, altrimenti risulta essere lineare (Fig. 1.6). Grazie alla soglia, ReLU permette alla rete di apprendere anche funzioni non lineari. Nonostante la sua semplicità si è dimostrata migliore, nella pratica, anche di funzioni di attivazioni sigmoidali. L'unico problema è dato dal gradiente che è zero o una costante e che quindi non è facile da controllare nel processo di addestramento. Inoltre la funzione non è differenziabile su tutto il dominio perchè presenta una discontinuità in θ.

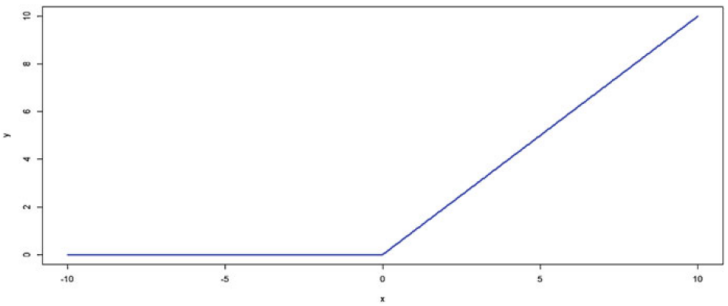


Figura 1.6: Funzione di attivazione ReLU

4. **Leaky ReLU**:

$$g(z) = \begin{cases} z & \text{se } z > 0 \\ \alpha z & \text{altrimenti} \end{cases}$$

L'obiettivo di questa ReLU modificata è quello di mitigare il problema del gradiente, inserendo una piccola pendenza negativa per i valori $z < 0$ (Fig. 1.7). La costante α usata è un valore compreso tra 0 e 1. In certi casi questa tecnica ha portato miglioramenti ma non sono sempre apprezzabili.

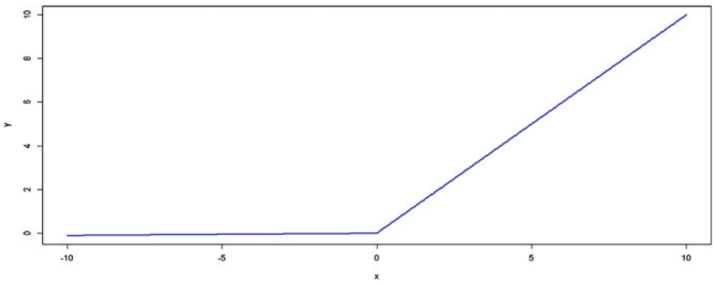


Figura 1.7: Funzione di attivazione Leaky ReLU

5. **Softmax**:

$$g(z_j) = \frac{e^{z_j}}{\sum_{c=1}^{C} e^{z_c}}, \quad j = 1, ...C$$

La softmax è una generalizzazione della sigmoide nel caso di etichette multinomiali: invece che un solo valore restituisce la distribuzione di probabilità su tutte le classi.

6. **Tanh**:

$$tanh(z) = \frac{sinh(z)}{cosh(z)} = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

Come la sigmoide ha un output a S con il vantaggio però di *bloccarsi* meno facilmente per valori estremi, dato che l'output è compreso tra -1 e 1 (Fig. 1.8). Per questa ragione è più utilizzata della sigmoide negli strati interni.

1.2.4 Funzioni loss

Una funzione loss (o funzione obiettivo) è una funzione che mappa un evento o un valore, di una o più variabili, in un numero reale che rappresenta il *costo* associato all'evento.

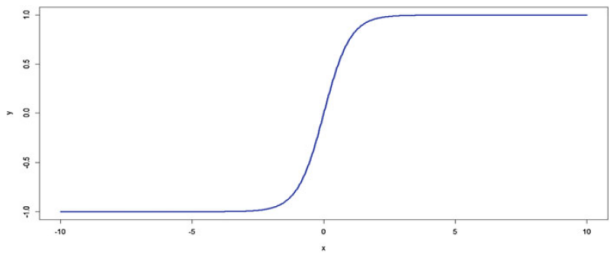


Figura 1.8: Funzione di attivazione tanh

Un problema di ottimizzazione, in generale, cerca di minimizzare una funzione loss (o la sua opposta in caso il problema iniziale sia di massimo).
Nel caso del machine learning, la funzione loss quantifica quanto un valore predetto sia distante dal valore reale. Con una metrica complessiva si ottiene un unico valore che quantifica quanto sia buono o meno il modello usato.
Addestrare una rete neurale con una funzione loss permette, quindi, di utilizzare metodi di ottimizzazione per stimare i parametri (pesi e bias) per il modello, minimizzando la loss si cerca infatti di ridurre l'errore sul training set (Fig. 1.9).
L'errore non verrà azzerato ma con un numero elevato di epoche di addestramento si cercherà di renderlo il minore possibile.
Tra le funzioni loss più comuni ci sono

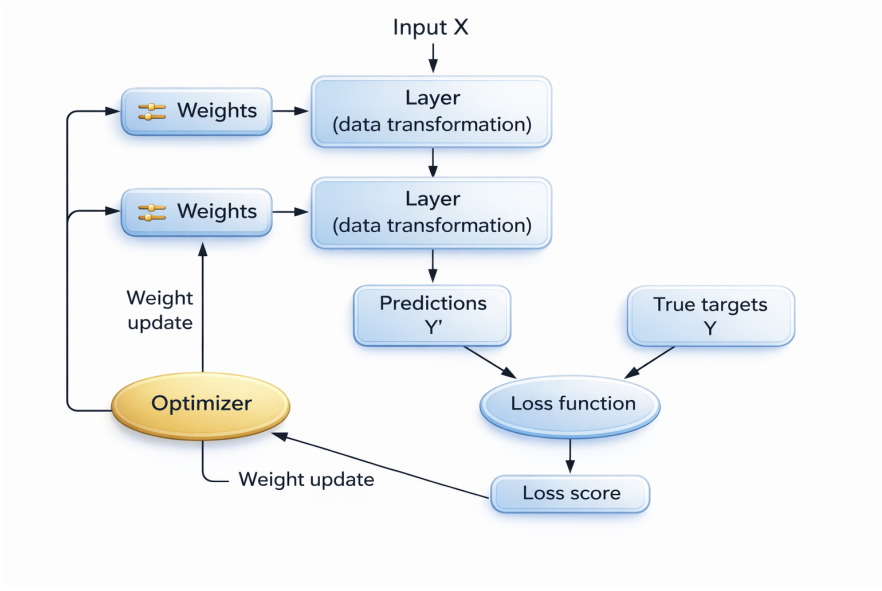


Figura 1.9: Schema di utilizzo delle funzioni loss durante il processo di training

1. **Errore quadratico medio (MSE)**:

$$L(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 = \frac{1}{n}||\hat{y} - y||_2^2$$

Viene fatta una media della somma dei quadrati delle differenze tra valore predetto ($\hat{y}_i$) e reale (y_i) su tutto il dataset (dimensione n). Questa loss è particolarmente sensibile agli outlier, per questo viene usata per output continui dove è difficile avere una corrispondenza perfetta.
Viene anche chiamata loss L_2 dato che la somma dei quadrati delle differenze non è altro che il quadrato della norma in L_2.

2. **Errore assoluto medio (MAE)**:

$$L(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| = \frac{1}{n}||\hat{y} - y||_1$$

Viene fatta una media della somma dei valori assoluti delle differenze tra valore predetto e reale. Viene anche detta loss L_1 o Lasso. A differenza della precedente non è differenziabile ma rende più semplice l'ottimizzazione ed è robusto agli outlier, per questo sono stati creati modelli che usano una versione ibrida delle due norme.

3. **Errore logaritmico quadratico medio (MSLE)**:

$$L(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}(log(\hat{y}_i) - log(y_i))^2$$

Sostituisce la differenza del MSE con una logaritmica; è molto utile quando gli output hanno intervalli di valori molto ampi e potrebbero portare ad errori esponenziali. Unico problema è che viene assegnata una penalità più alta alle previsioni basse rispetto che alle alte.

4. **Cross-entropy (CE)**: La Cross-entropy è una loss usata nel campo della classificazione. Nel caso binario, in cui si deve distinguere tra 0 e 1, la formula delle CE

è

$$L(\mathbf{w}) = -\frac{1}{n}\sum_{i=1}^{n}(y_i log(p_i) + (1-y_i) log(1-p_i))$$

dove p_i è la probabilità prevista da modello, è un valore compreso tra 0 e 1. Deriva dalla formula della divergenza di Kullback-Leibler (divergenza KL), che misura la *distanza* tra due distribuzioni di probabilità.

Esiste anche una Cross-entropy multiclasse data da

$$L(\mathbf{w}) = -\sum_{i=1}^{n}\sum_{c=1}^{C} y_{ic} log(p_{ic})$$

in cui C è il numero di classi, y_{ic} vale 1 se i appartiene alla classe c e 0 altrimenti e p_{ic} è la probabilità che i appartenga alla classe c.

Esistono tante altre funzioni loss create specificatamente per certi tipi di problemi, verranno presentate più avanti quelle specifiche nel caso della segmentazione.
Ottimizzare i pesi tramite le funzioni loss porta a grandi miglioramenti nelle prime epoche di addestramento (visioni dell'intero dataset), miglioramento che diventa sempre più lento fino al raggiungimento di un minimo locale. La minimizzazione della loss (rischio empirico) però non è l'unico aspetto da considerare. Infatti esiste un'altra componente, detta rischio strutturale, che *misura* il rischio che il modello cada nell'overfitting, cioè che diventi troppo performante sul training set e che perda di generalizzazione. Per questo vengono introdotti metodi come quello dell'*early stopping* che prevede di fermare l'addestramento prima della fine delle epoche, sfruttando l'errore ottenuto sul validation set (Fig. 1.10).

1.2.5 Retropropagazione del gradiente

Una volta che è stata calcolata la funzione loss per le previsione ottenute, come già anticipato, si modificano i pesi per cercare di minimizzarla. La tecnica principe di quest'ultimo passaggio è la *retropropagazione del gradiente* (Figura 1.11).
La funzione loss dipende dai valori reali dei dati e dalla previsione ottenuta in output dalla rete, che è frutto della funzione di attivazione dell'ultimo strato. Quest'ultima funzione a sua volta dipende dai valori dei neuroni al penultimo layer e dai pesi delle

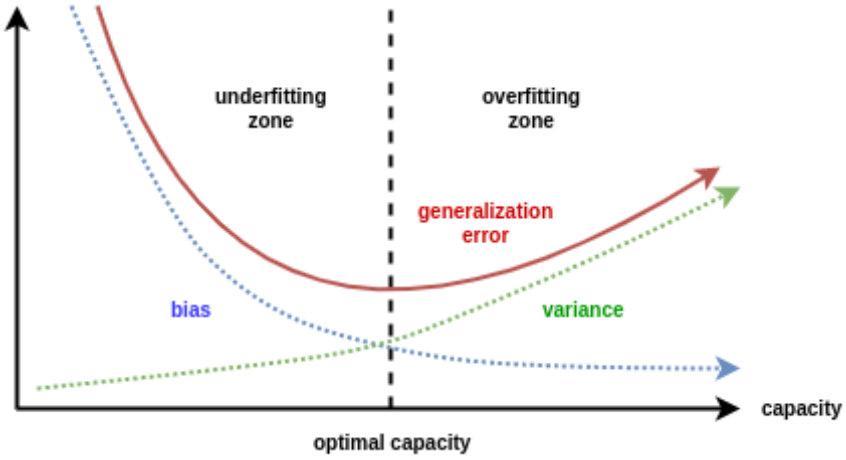


Figura 1.10: Metodo dell'early stopping per fermare l'addestramento nel punto di equilibrio tra underfitting (errore di bias) e overfitting (errore di varianza)

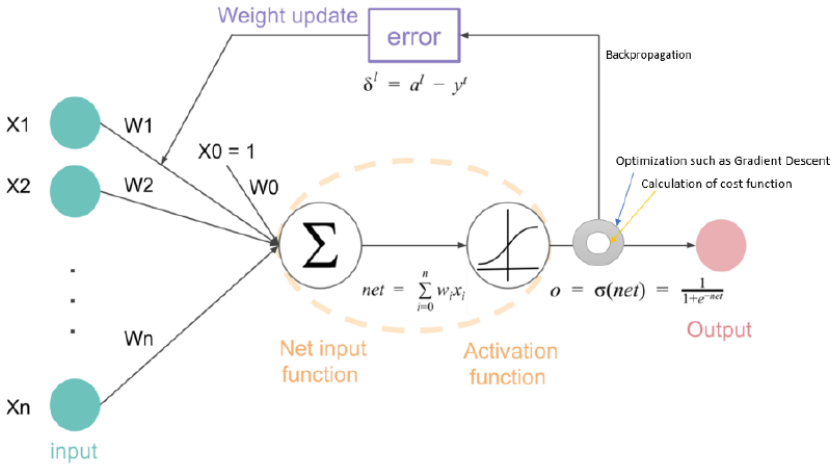


Figura 1.11: Schema di addestramento di una rete con retropropagazione del gradiente

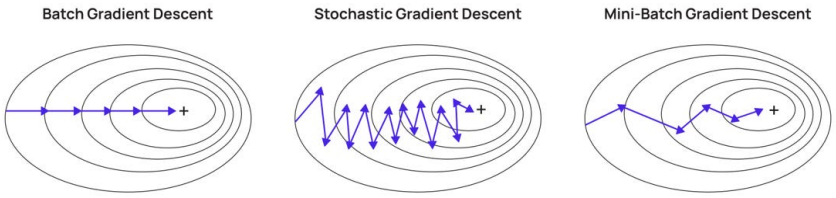


Figura 1.12: Tipologie di discesa del gradiente

connessioni. Ragionando ricorsivamente si giunge fino ai neuroni di input; la loss dipende dunque da tutti i parametri di pesi e bias che definiscono la rete.
Si può quindi calcolare il cosiddetto gradiente di perdita che è il gradiente della funzione loss rispetto a tutti i parametri di pesi e bias. Utilizzando i metodi di discesa del gradiente è ora possibile ottimizzare la funzione loss per determinare i valori migliori a partire dai dati usati in input: a ogni epoca si calcola il gradiente per ogni esempio del set e se ne fa una media (*gradiente batch*).
Calcolare tutti i gradienti con un dataset molto grande risulta però proibitivo, per questo sono stati cercati metodi alternativi come quello del gradiente *mini-batch* o del gradiente *stocastico* (Fig. 1.12). Nel primo caso si considera la media dei gradienti solo su un campione casuale ridotto di esempi del dataset mentre nel secondo si usa un solo gradiente alla volta, è meno preciso ma facendo tanti passi in un'epoca cerca una discesa in media.

1.3 CNN

Tra le tipologie di DNN più usate ci sono le reti convolutive (CNN) che hanno una vasta gamma di applicazioni tra cui il riconoscimento di oggetti e pattern all'interno di immagini, la trascrizione automatica di testi e la genomica.
Le CNN, tramite l'uso di filtri adatti, riescono a catturare dipendenze spaziali e temporali all'interno di immagini. Sono costituite da sequenze di strati convolutivi che permettono di catturare, prima caratteristiche a basso livello, come bordi e colori, per poi riconoscere sempre più dettagli negli strati seguenti. Tramite la concatenazione di questi strati si arriva ad avere una conoscenza completa dell'immagine.
L'operazione di convoluzione può ridurre le dimensioni dell'immagine oppure può

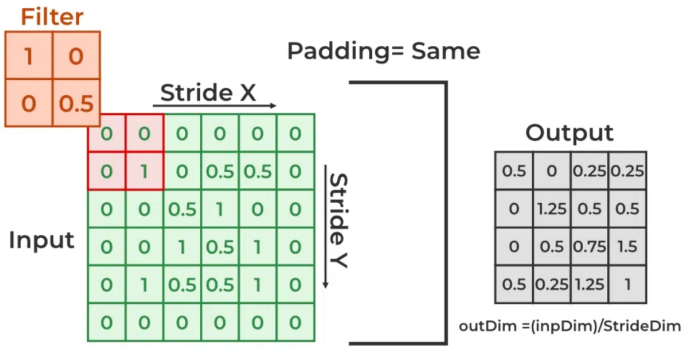


Figura 1.13: Schema di un procedimento di Padding

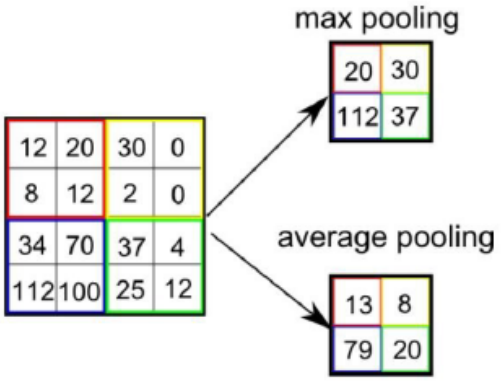


Figura 1.14: Tipologie di Pooling

conservarle: nel primo caso si parla di *Valid Padding* e nel secondo di *Same Padding*. L'operazione di Padding avviene scorrendo, con un passo predefinito (stride), un filtro sull'immagine (vista come matrice di pixel) e svolgendo tutti i prodotti tra i pixel dell'immagine e i valori del filtro. Il risultato sarà l'immagine convoluta. In figura 1.13 è riportato lo schema di un procedimento di Padding in cui, affinchè l'output avesse le stesse dimensioni della matrice di partenza, è stato aggiunto un bordo di zeri all'immagine: questa tecnica è detta di *Zero-Padding*.

Finiti i layer di convoluzione si ha un layer di *Pooling* che riduce ulteriormente le dimensioni delle caratteristiche convolute; ciò è utile per individuare le caratteristiche dominanti che sono invarianti per posizione e rotazione. Anche per il Pooling esistono diverse possibilità tra cui, ad esempio, *Max Pooling* e *Average Pooling*. Analogamente al Padding, per il Pooling si ha un filtro che scorre sull'immagine e restituisce il valore massimo (Max Pooling) o il valore medio (Average Pooling) (Figura 1.14).

L'output del Pooling viene "appiattito" in un vettore per poter essere fornito ad una

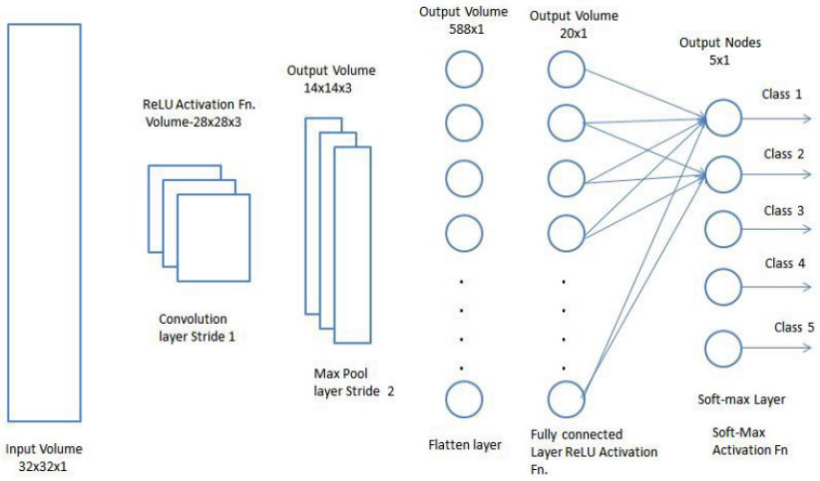


Figura 1.15: Struttura generica di una rete CNN

rete feedforward. L'ultimo strato di questa rete prima dell'output è un *Fully-connected layer* che permette alla rete di imparare combinazioni non lineari delle feature ad alto livello.

La funzione di attivazione dell'output è una funzione softmax, presentata nel paragrafo 1.2.3, che normalizza l'output restituendo una distribuzione di probabilità per le varie classi. In figura 1.15 è riportata la struttura di una rete CNN.

Capitolo 2

Modelli di segmentazione

All'interno di questo capitolo viene introdotta la classe di metodi di *computer vision* usati per la segmentazione di immagini. A seguito di una panoramica generale, il focus del capitolo si sposta su due metodi in particolare, StarDist e Cellpose, che sono stati oggetto di sperimentazione nel presente elaborato.

2.1 Segmentazione di immagini

La **segmentazione di immagini** è uno dei maggiori campi di ricerca della computer vision ed è alla base del riconoscimento di pattern e comprensione delle immagini.

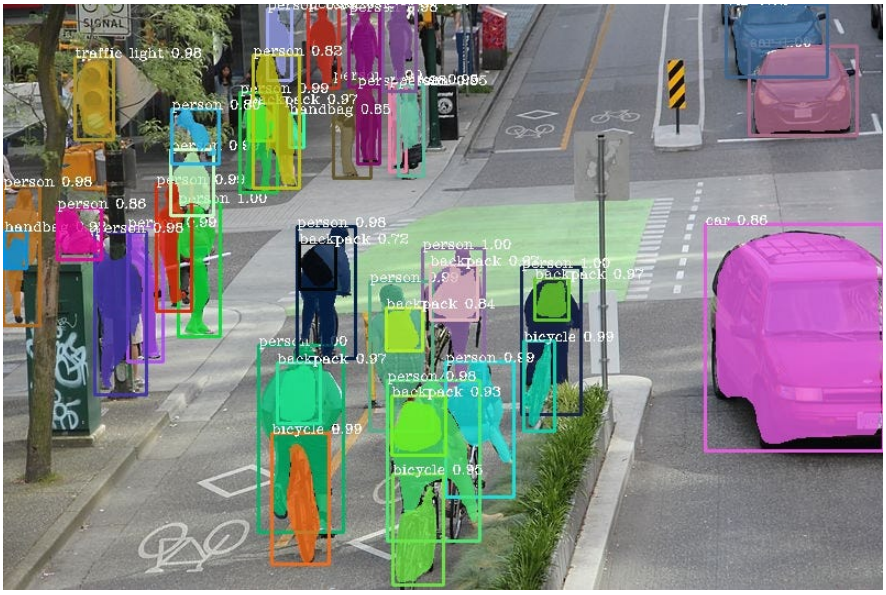


Figura 2.1: Esempio di riconoscimento di oggetti tramite segmentazione

La segmentazione riveste un'enorme importanza in svariati ambiti tra cui la guida autonoma, la realtà aumentata, strumenti di ricerca tramite immagini e tecnologie mediche intelligenti.

La segmentazione consiste nel dividere le immagini in regioni con caratteristiche differenti, per poi estrarre alcune regioni significative, dette *regioni di interesse (ROI)*. La difficoltà in questo campo risiede proprio nella definizione stessa di ROI, che varia da persona a persona, rendendo questo problema un problema mal posto.

Una seconda difficoltà è legata poi al come viene rappresentato l'oggetto a cui si è interessati all'interno dell'immagine. L'immagine è un insieme di pixels che possono essere raggruppati in insiemi in base al colore, alla texture o ad altre informazioni. Queste caratteristiche sono attributi locali delle immagini, da cui non è facile risalire ad attributi globali come forma e posizione.

La tipologia più generale di segmentazione è quella *semantica* (Fig. 2.2) in cui viene associata una classe semantica ad ogni pixel senza distinzione tra oggetti e elementi di sfondo. Specializzando queste tecniche si può ottenere la *segmentazione di istanza* in cui lo sfondo viene ignorato e le forme degli oggetti diventano più precise; in figura 2.2 si può notare come questa seconda segmentazione posti alla distinzione delle singole macchine.

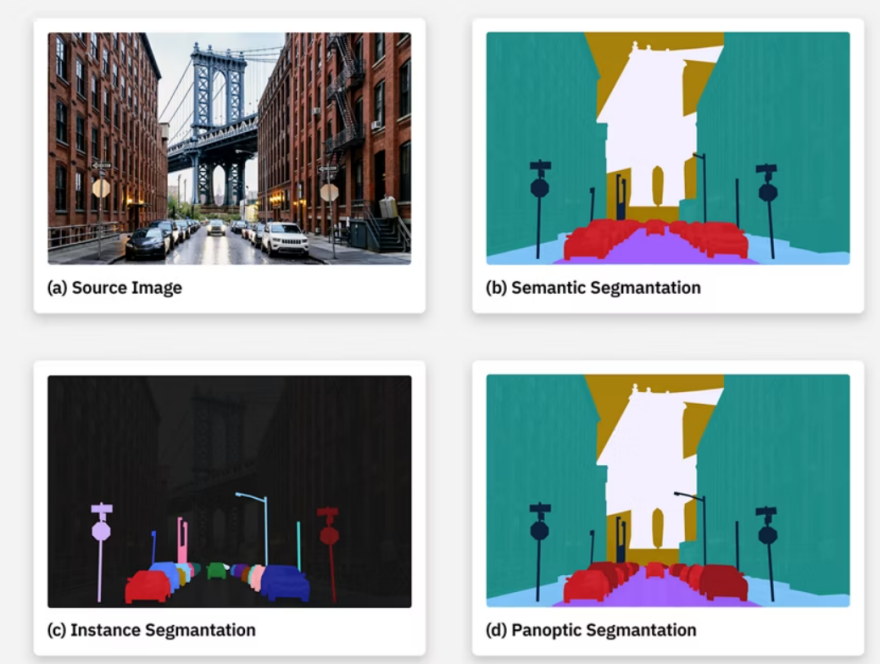


Figura 2.2: Tipologie di segmentazione

Infine si ha la *segmentazione panottica* che unisce le due precedenti, differenziando le istanze e segmentando anche lo sfondo.

I *metodi classici di segmentazione* si focalizzano principalmente sull'evidenziare e ottenere precise informazioni a partire da una singola immagine; ciò richiede spesso una conoscenza professionale e l'intervento umano.

Dal 1970 ad oggi la ricerca ha prodotto diverse tecniche in questo settore. La *Co-segmentazione*, per esempio, si differenzia da quanto detto in precedenza perchè mira ad identificare oggetti comuni presenti in più immagini. Anche in questo caso però la conoscenza a priori è fondamentale.

Grazie poi all'enorme quantità di dati e annotazioni disponibili, è stato introdotto, anche in questo settore, l'uso del deep learning.

Sono stati fatti molti passi avanti nella ricerca anche se le sfide nella segmentazione semantica sono ancora tante, a causa di alcuni limiti come la scarsità di annotazioni o i lunghi tempi di processazione. In figura 2.3 è riportato un elenco dei principali metodi di segmentazione. Il capitolo, dopo una panoramica sulle prime due classi, si concentrerà sulla segmentazione semantica che sfrutta il DL.

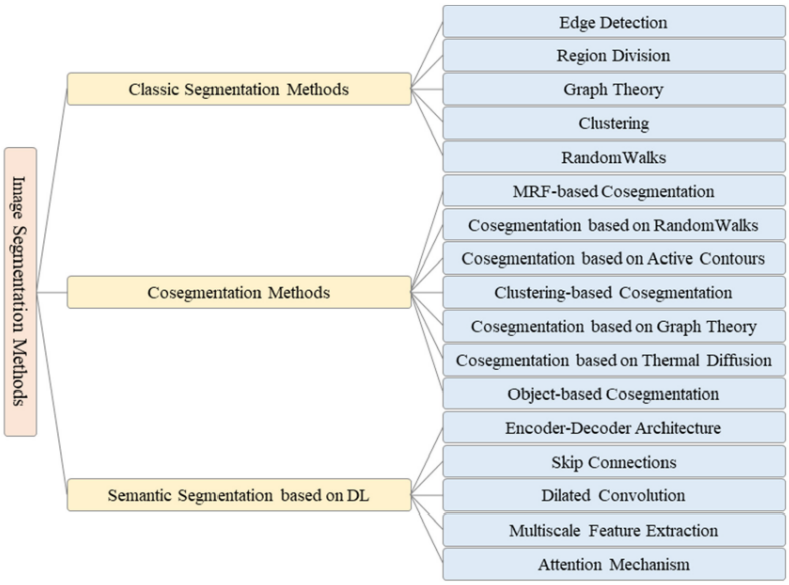


Figura 2.3: Classi di metodi per la segmentazione di immagini

2.1.1 Metodi classici di segmentazione

Inizialmente i metodi classici sono stati proposti per immagini in scala di grigi, considerando nella stessa regione i pixel con livello di grigio simile e dividendo le regioni in corrispondenza delle discontinuità. Nelle immagini a colori si usa sempre la similarità tra pixel per creare "superpixel" che poi andranno uniti per formare regioni più grandi. I principali metodi classici sono:

1. **Edge Detection**: si segmenta cercando prima i bordi delle regioni. Questi sono costituiti dai punti che separano pixel con livelli di grigio molto diversi.

2. **Region Division**: è un metodo bottom-up in cui, partendo da pixel singoli, questi vengono aggregati iterativamente in base alla similarità delle loro feature formando delle regioni. Questa operazione di *merging*, tra pixel prima e tra regioni poi, avviene se la misura di similarità supera una certa soglia ricavata dall'istogramma della scala di grigi dell'immagine. Quando non è più possibile procedere, si è arrivati alla suddivisione ottimale.

3. **Graph Teory**: i pixels dell'immagine diventano i vertici di un grafo mentre gli archi tra pixels vicini sono pesati con una misura di similarità. Analogamente a quanto si fa con il metodo precedente, i pixels/regioni vengono aggregati ricorsivamente, sfruttando tecniche di ottimizzazione specifiche per i grafi.

4. **Clustering**: è un metodo basato sull'algoritmo di Lloyd per dividere un insieme di punti in K cluster. L'algoritmo prevede di inizializzare K punti come centri dei cluster; vengono poi calcolate le distanze di ogni punto dell'immagine da ogni centro, in modo da assegnare ad ognuno il centro più vicino. Una volta completata l'assegnazione, viene calcolato il nuovo centroide di ogni cluster come media dei punti presenti al suo interno. Infine si ripetono questi ultimi due passaggi fino a convergenza dell'algoritmo.

5. **Random Walks**: è un particolare algoritmo, sempre basato sulla teoria dei grafi, che sfrutta la distanza di ogni pixel da ciò che viene inizializzato come primo piano e sfondo, per assegnare le labels.

Figura 2.4: Due esempi di risultati della co-segmentazione

2.1.2 Metodi di co-segmentazione

Come già anticipato, i metodi classici si focalizzano nell'estrarre feature da una sola immagine, il che rende difficile individuare informazioni ad alto livello. L'idea della segmentazione collaborativa, detta anche co-segmentazione, prevede l'estrazione di oggetti in primo piano comuni in un set di immagini, senza l'intervento umano (esempio in figura 2.4). Per raggiungere la co-segmentazione è necessario estrarre il primo piano di una o più immagini, con metodi classici, come conoscenza a priori, in modo da poter riconoscere lo stesso oggetto in immagini simili. In un metodo di co-segmentazione convivono quindi un algoritmo di segmentazione classica, per ottenere una base, e un metodo di apprendimento non supervisionato, per imparare a segmentare immagini simili alla base. Di conseguenza, l'aumento di efficienza di un metodo di questo tipo, si ottiene ottimizzando l'apprendimento non supervisionato o migliorando l'algoritmo classico usato per la base.
In base alla scelta fatta per i due algoritmi, si differenziano i principali metodi in

1. **MRF-Based Co-Segmentation**
2. **Co-Segmentation Based on Random Walks**
3. **Co-Segmentation Based on Active Contours**
4. **Clustering-Based Co-Segmentation**
5. **Co-Segmentation Based on Graph Theory**
6. **Co-Segmentation Based on Thermal Diffusion**
7. **Object-Based Co-Segmentation**

2.2 Segmentazione basata sul Deep Learning

Con il continuo sviluppo dell'attrezzatura per l'acquisizione delle immagini, si è verificata una crescita significativa nella complessità dei dettagli delle immagini e nella differenziazione degli oggetti. Non bastano più le feature a basso livello, come texture e colore, per ottenere buoni risultati. I metodi manuali o euristici, inoltre, non possono più competere con la crescente complessità delle immagini.
Inizialmente sono stati impiegati metodi di Machine Learning come il Random Forest per poi iniziare a sfruttare algoritmi di Deep Learning che hanno portato significativi miglioramenti.
Nell'approccio originale con il DL, le immagini venivano divise in piccole patch per addestrare il modello e classificare i pixels. La divisione in patch era dovuta all'uso, nella rete, di layers completamente connessi che richiedono immagini di dimensioni fissate. Nel 2015 poi è stata proposta una rete con la convoluzione al posto della connessione completa (fully convolutional networks - FCN) che rende possibile avere immagini di dimensioni diverse [1]. Nelle FCN, come mostrato in figura 2.5, le immagini vengono inizialmente ridotte di dimensione tramite convoluzione e pooling nella fase detta di *encoder*; successivamente, tramite l'*up-sampling*, si ottiene un output delle stesse dimensioni delle immagini di partenza.

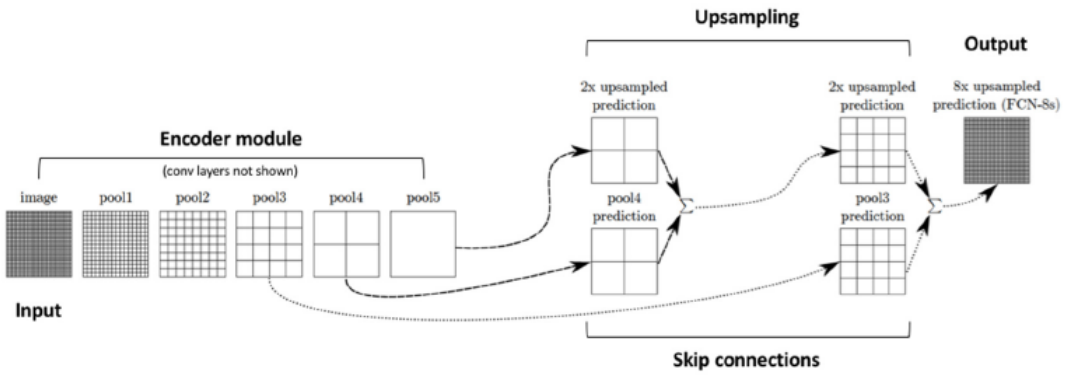


Figura 2.5: Architettura di una rete FCN

La struttura delle FCN è alla base delle reti più utilizzate oggi nell'ambito della segmentazione di cui di seguito sono riportate le principali caratteristiche.

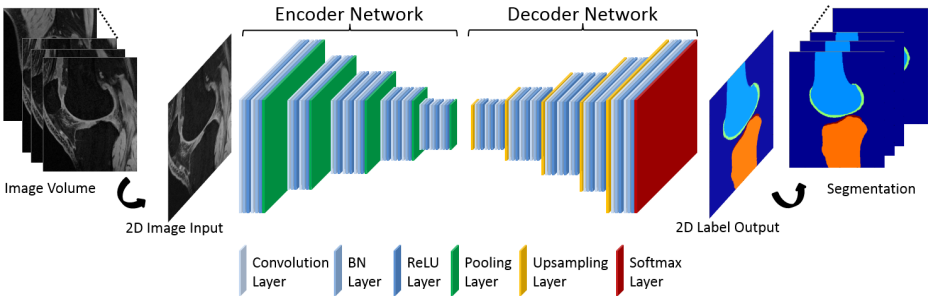


Figura 2.6: Architettura encoder-decoder

2.2.1 Architettura Encoder-Decoder

L'architettura *encoder-decoder* è basata sulle reti FCN. Prima delle reti FCN, le convolutive (CNN) avevano raggiunto ottimi risultati nella classificazione di immagini. L'output delle CNN però è dato dalla sola classe mentre la segmentazione semantica necessita di mappare le feature ad alto livello ottenute di nuovo sull'immagine. Da qui la necessità della fase di *decoder*.

Nella fase di encoder, la convoluzione e il pooling vengono usati per estrarre dall'immagine le feature ad alto livello che contengono le informazioni semantiche. In questa sezione le reti più usate sono le VGG, Inception [2] e ResNet [3].

Nella fase di decoder, invece, vengono svolte operazioni per generare una maschera segmentata dell'immagine a partire dal vettore delle feature estratte, questo processo è appunto l'up-sampling nominato per le FCN.

Un esempio di up-sampling sono i metodi di interpolazione per aumentare la dimensione delle immagini, tramite l'inserimento di nuovi pixels tra quelli esistenti. Nel caso delle reti SegNet [4], invece, si usa l'operazione inversa del max-pooling delle CNN, chiamato *unpooling*.

In figura 2.6 è riportato uno schema generale dell'architettura encoder-decoder.

2.2.2 Skip Connections

La tecnica delle *skip connections* è stata introdotta per risolvere il problema di degradazione della rete: più una rete è profonda più le performance nel training peggiorano. Con questa tecnica, uno strato non è necessariamente connesso solo a quello successivo ma può anche "saltare" alcuni layers.

Diverse skip connections sono state presentate nelle reti ResNet e DenseNet [5], solo con le U-Net [6] però si arrivano a sperimentare lunghe skip connections. Nelle U-Net infatti ogni layer di encoder è connesso al suo corrispettivo nei decoder, saltando così, in alcuni casi, molti layer intermedi (Fig. 2.7).

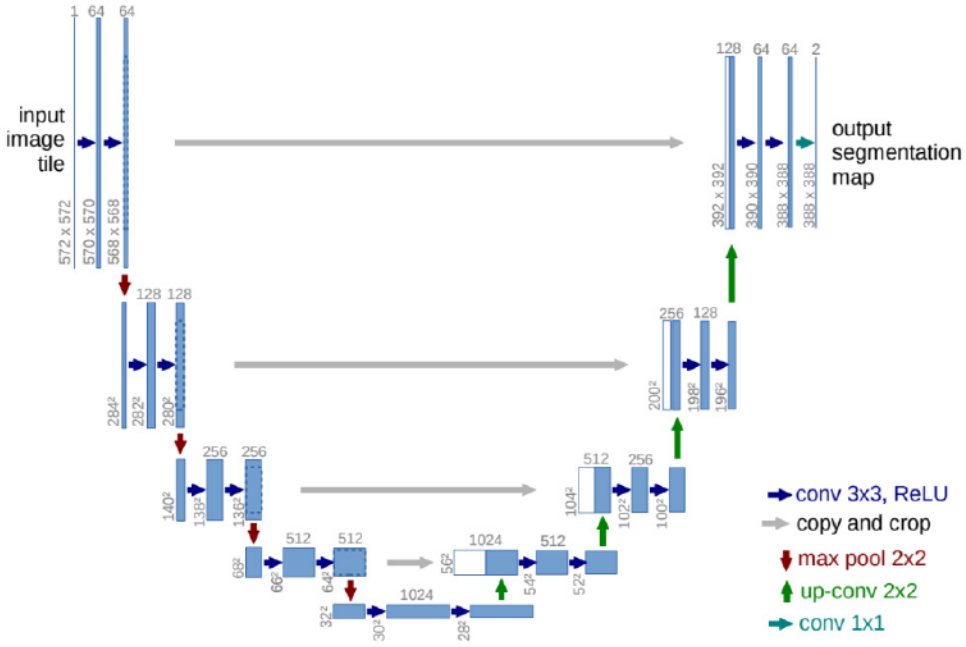


Figura 2.7: Architettura di una rete U-net

Le U-Net sono state proposte per risolvere il problema delle annotazioni nella segmentazione di immagini biologiche ottenute al microscopio; sono tuttora ampiamente utilizzate nella segmentazione di immagini mediche, infatti sono alla base dei modelli che verranno esposti nelle sezioni successive.

2.2.3 Convoluzioni Dilatate

Le convoluzioni dilatate, dette anche convoluzioni con "buchi", si costruiscono inserendo degli spazi vuoti all'interno del kernel usato nella convoluzione, nella fase di riduzione della dimensione. Questo tipo di convoluzioni permettono di usare kernel più grandi abbassando i costi computazionali. Questa tecnica è stata utilizzata in reti come la DeepLab [7] o la ResNet per evitare di perdere risoluzione dato che permette di evitare ulteriori riduzioni con layer di pooling. Infatti con un kernel di dimensioni maggiori si coprono aree più ampie pur preservando i dettagli.

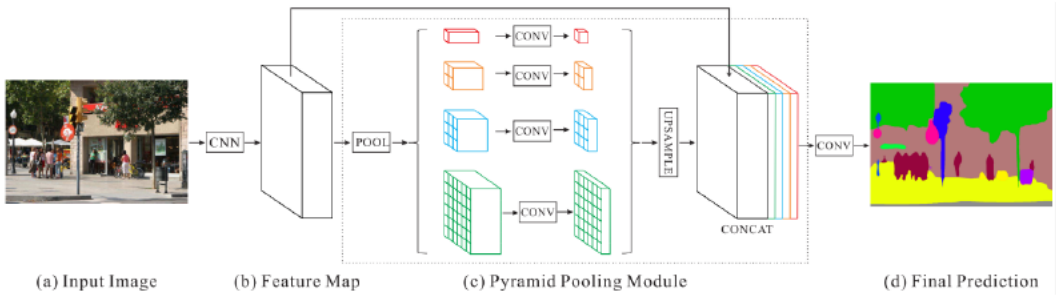


Figura 2.8: Schema di una rete PSP con modulo di pooling piramidale

Spesso, per evitare artefatti causati dall'uso del kernel a griglia, si usano tassi di dilatazione (quanti pixel vuoti inserire tra due pixel del kernel) diversi nei vari strati convolutivi.

2.2.4 Estrazione Feature Multiscala

Per risolvere il problema di dover fornire alla rete CNN input di dimensione fissata, è stato sviluppato il *pooling piramidale* che risulta efficiente nel caso della segmentazione semantica perchè permette di usare pooling diversi in zone differenti dell'immagine, per poi riscalare tutto alle dimensioni desiderate. Da questa tecnica nasce il modulo di pooling piramidale (PPM) che, a partire dalla mappa delle feature ottenuta con una CNN, serve ad estrarre e aggregare feature di sottoregioni diverse che hanno scale diverse. Riscalando e concatenando quanto ottenuto con la mappa iniziale delle feature, si ottiene una nuova mappa contenente informazioni sia di contesto globale che locale. In figura 2.8 è riportata la struttura di una rete PSP [8] che usa il PPM all'interno di una ResNet.
Il pooling piramidale può essere combinato con la convoluzione dilatata ottenendo così sia l'aumento del campo recettivo del kernel, sia la capacità di catturare feature multiscala.

2.2.5 Meccanismo di attenzione

I meccanismi di attenzione sono tecniche per far concentrare i modelli di DL sulle parti più importanti dell'input. Prestare maggiore *attenzione* ai dettagli più rilevanti permette di non perdere dettagli significativi e ottimizzare l'uso di spazio e memoria.

Un meccanismo di attenzione attribuisce i pesi che regolano l'importanza di ciascuna parte dell'input, attraverso un metodo di apprendimento supervisionato.
Per rappresentare la dipendenza tra regioni diverse dell'immagine, specialmente per regioni a lunga distanza, e ottenere la loro rilevanza semantica, alcuni metodi classici del campo della processazione del linguaggio naturale (NLP) sono stati applicati alla computer vision. Infatti i modelli di traduzione automatica con le reti ricorrenti (RNN) sono stati tra i primi che hanno implementato meccanismi di attenzione. Le reti RNN [9] processano i dati in serie, seguono quindi un ordine specifico e hanno difficoltà nel riconoscere dipendenze a grandi distanze; in questo caso i meccanismi di attenzione permettono di analizzare l'intera sequenza e poi decidere l'ordine da seguire, migliorando notevolmente la rete. Analogamente, anche le CNN perdono il contesto globale concentrandosi con le convoluzioni su sottoinsiemi sempre più piccoli, per questo, anche in questo caso, possono essere utili i meccanismi di attenzione.
Originariamente, quindi, i meccanismi di attenzione sono stati introdotti nelle RNN per risolvere il problema del gradiente che svanisce creando connessioni tra parti distanti di una sequenza di dati. Il processo viene quindi descritto, nella sua forma più generale, come nei Trasformer [10], con tre componenti: query, chiave e valori. La Query (Q) rappresenta ciò che il modello sta cercando, per esempio il soggetto in una frase. La Chiave (K) serve da identificatore per le informazioni contenute nell'input, infine il Valore (V) contiene l'informazione effettiva che verrà aggiornata con il meccanismo.
Complessivamente poi il peso degli elementi viene aggiornato, confrontando le Query con varie Chiavi ottenendo

$$Attenzione(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

dove d_k è il numero di dimensioni della matrice K. Ciò consente ai modelli di gestire dipendenze a lungo raggio, infatti QK^T è la matrice di similarità che confronta la Query con tutte le Chiavi. I valori ottenuti vengono poi riscalati e moltiplicati per V, ogni elemento diventa quindi "consapevole" dell'imporatnza che ha per gli altri. I vettori Q, K e V per ogni input vengono generati facendo passare l'input originale attraverso un livello lineare che precede il primo livello di attenzione. Questo strato lineare è suddiviso

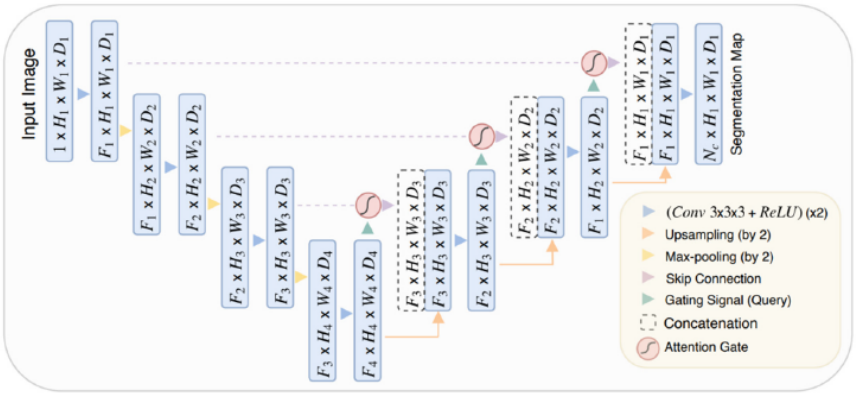


Figura 2.9: Architettura di una rete U-Net con Attention Gate

in tre matrici univoche di pesi del modello: W_Q, W_K e W_V. I valori di peso specifici in esso contenuti vengono appresi attraverso un pre-addestramento auto-supervisionato su un enorme set di dati di esempi di testo.

Anche nelle U-Net è stato introdotto un meccanismo di attenzione. Nelle reti U-Net classiche, le connessioni skip trasferiscono tutte le informazioni dall'encoder al decoder del livello corrispondente ma, in molte immagini mediche e di satelliti, gran parte dell'immagine è irrilevante e può confondere il modello. Si introduce così l'*Attention Gate (AG)* che filtra le connessioni skip riducendo il rumore (Fig. 2.9). Siano x la feature map dell'encoder corrispondente al decoder in considerazione e g la feature map del decoder al passo precedente, la formula usata nell'AG è

$$\alpha = \sigma\left(\psi^T \cdot \mathrm{ReLU}(W_x x + W_g g + b)\right) \tag{2.1}$$

da cui si ottiene il vettore da passare con la skip connection in cui le informazioni di x sono filtrate per rilevanza con

$$x' = \alpha \cdot x \tag{2.2}$$

Nell'equazione W_x e W_g sono le trasformazioni lineari, analoghe a quelle operate con Query e Chiavi, che portano i due vettori ad una dimensione comune. In questo caso però i risultati vengono sommati invece che moltiplicati come nel caso dei Trasformer. Vengono poi usate in sequenza la ReLU ($\max(0, z)$) per introdurre non linearità, la proiezione a singolo canale ψ^T e la sigmoide σ che serve a mappare i valori tra 0 e 1 in modo da ottenere i pesi di attenzione.

2.3 StarDist

La classificazione e segmentazione dei nuclei è una task molto importante nell'ambito della computational pathology. StarDist [11] [12] è un metodo di segmentazione, basato sul deep learning, inizialmente sviluppato per immagini ottenute da microscopi a fluorescenza, che è poi stato adattato anche a immagini istopatologiche.
In figura 2.10 è riportato un esempio di immagini utilizzate per l'addestramento di StarDist in ambito istopatologico, provenienti dalla Colon Nuclei Identification and Counting (CoNIC) del 2022 [13] vinta appunto dal metodo StarDist.
L'identificazione di nuclei in immagini microscopiche è un compito onnipresente nell'ambito delle scienze e può diventare particolarmente ostico nel caso in cui i nuclei siano raggruppati densamente tra loro. A differenza dei classici metodi bounding-box, che individuano gli oggetti racchiudendoli in rettangoli, l'idea di StarDist è quella di rappresentare gli oggetti come poligoni convessi a stella, forma che si adatta bene alla rotondità dei nuclei.
Nello specifico ciò che fa StarDist è, per ogni pixel, prevedere un poligono convesso

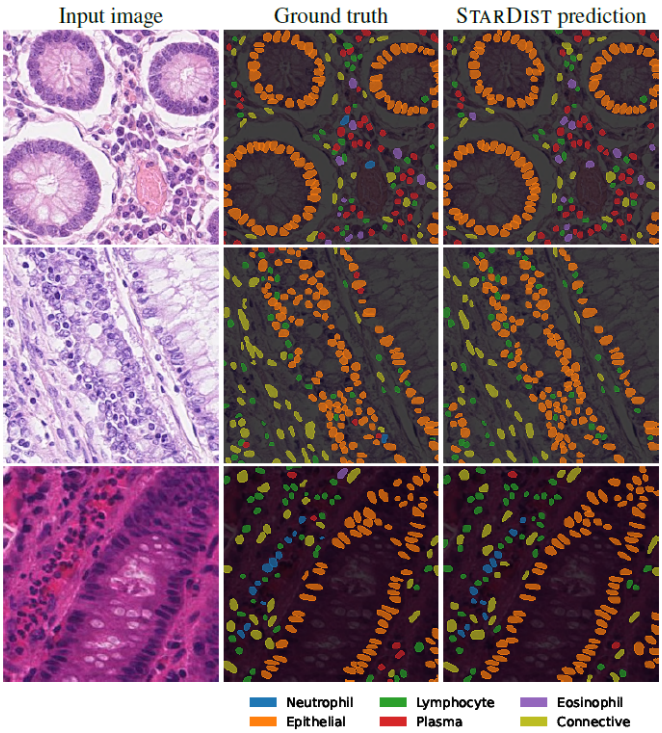


Figura 2.10: Esempi di immagini appartenenti alla CoNIC challenge con la maschera prodotta da StarDist

a stella, usando n raggi e il pixel di partenza come centro; il modello sfrutta una regressione sulle distanze, lungo i raggi, del pixel dal bordo del poligono predetto. Ovviamente ciò è ben posto per i pixel che non sono di background. StarDist infatti, separatamente dalla regressione sui raggi, opera una classificazione probabilistica sui pixel, così da accettare il poligono solo per i pixel che hanno una alta probabilità di appartenere ai nuclei (alta *object probability*).
I candidati poligoni rimasti con la loro object probability vengono scremati tramite una *non-maximum suppression (NMS)* per ottenere l'insieme finale di poligoni.
Più in dettaglio, mentre ogni pixel può essere classificato come appartenente ad un oggetto o allo sfondo, per quelli appartenenti agli oggetti, esiste anche l'object probability $d_{i,j}$ che è la distanza euclidea normalizzata dal pixel di sfondo più vicino. In questo modo l'NMS favorirà i poligoni associati a pixel vicino al centro delle cellule che, tendenzialmente, rappresentano gli oggetti in maniera più accurata. Per i pixel appartenenti agli oggetti, come anticipato, vengono calcolate anche le n distanze radiali $r^k_{i,j}$ che si ottengono, invece, seguendo le direzioni radiali fino ad incontrare un pixel con una classificazione diversa (altro oggetto o sfondo) e calcolandone la distanza euclidea.

2.3.1 Implementazione

Nonostante l'approccio generale del metodo non sia particolarmente legato ad uno specifico approccio di regressione o classificazione, è stata scelta la U-Net come base del modello. Dopo il layer finale della U-Net è stato aggiunto un altro layer di convoluzione 3x3 con 128 canali e funzione di attivazione Relu. Questo viene fatto per evitare che i due output layer, per object probability e distanze radiali, abbiano troppe feature da ridurre a pochi canali.
Nel training le loss da minimizzare sono una *binary cross-entropy* per quanto riguarda l'object probability e un *MAE* per le distanze. Per le distanze del poligono, inoltre, il MAE è pesato con l'object probability del ground truth per ogni pixel. Di conseguenza lo sfondo non contribuisce alla loss mentre i pixels di centro cellula sono pesati di più, il che è appropriato dato che verranno favoriti nella NMS.
Infine, la NMS viene eseguita con un approccio greedy, scegliendo in ogni zona i candidati con object probability sopra una certa soglia per poi restituire la loro intersezione.

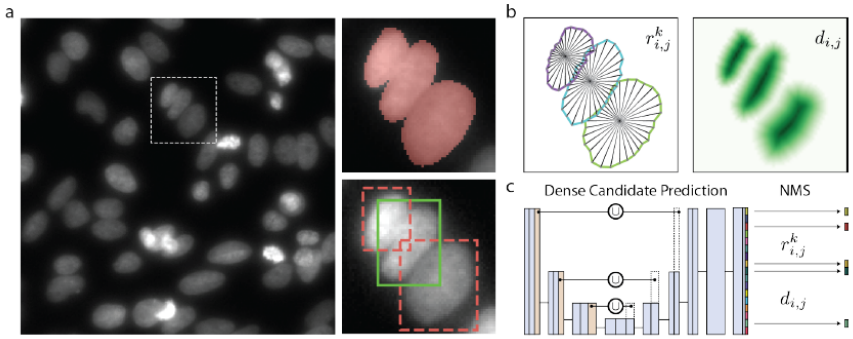


Figura 2.11: (a) Possibile errore di segmentazione per nuclei addensati: merge delle cellule o soppressione di istanze valide. (b) Previsione del metodo StarDist per $r^k_{i,j}$ e $d_{i,j}$. (c) Metodo di previsione dei due parametri tramite la U-Net e l'NMS.

In figura 2.11 è riportato un esempio del funzionamento del modello appena illustrato, in uno dei casi più difficili per la segmentazione: la sovrapposizione di oggetti.
Un vantaggio di questo modello è dato dai pochi parametri modificabili, tra i quali la soglia per l'NMS e per la scelta dei poligoni con object probability, cambiando i quali possono notevolmente migliorare le prestazioni del modello, come verrà mostrato in seguito.

2.4 Cellpose

Ogni metodo di segmentazione deve bilanciare l'automazione con la flessibilità. La gamma di metodi va dalla segmentazione completamente manuale a pipeline personalizzate dall'utente con molti parametri a metodi basati su reti neurali, completamente automatizzati. Nonostante questi ultimi riducano enormemente il tempo di lavoro umano e permettano di elaborare grandi dataset, sono generalmente allenati su dataset specializzati perdendo così potere di generalizzazione. Questo implica che, per ogni tipologia diversa di tessuto, il modello richieda nuove immagini segmentate a mano per raggiungere performance migliori.
Riconoscendo questo problema nella segmentazione dei nuclei, la challenge Data Science Bowl del 2018 [14] ha deciso di presentare un dataset ottenuto unendo dati provenienti da laboratori diversi. Si è notato come metodi allenati su questi dati abbiano più capacità di generalizzazione rispetto a quelli addestrati su dati provenienti da un unico laboratorio. Da questa challenge sono stati sviluppati altri metodi come StarDist e

nucleAlzar [15].

L'idea alla base di Cellpose [16] nasce proprio dalla ricerca di un metodo il più possibile flessibile e generalizzabile, nonostante la forma di cellule e nuclei sia molto variabile tra i tessuti.

In alcuni classici metodi di segmentazione, i valori dell'immagine in scala di grigi creano una mappa topologica in cui i bacini di attrazione rappresentano le regioni segmentate. Ciò funziona quando gli oggetti segmentati sono a forma di "blob", formano cioè bacini regolri. Molti tipi di cellule però non sono a forma di "blob", da qui l'idea di costruire una rappresentazione intermedia dell'immagine che permetta di ottenere sempre una mappa topologica regolare (Fig. 2.12 a,b).

Inizialmente le mappe topologiche vengono generate, a partire da maschere ground truth annotate manualmente, tramite un processo di diffusione simulata (Fig. 2.12 a). Una rete neurale è poi addestrata per predire il gradiente orizzontale della mappa topologica, il gradiente verticale e, infine, una mappa di probabilità che indica se un dato pixel faccia o meno parte di una cellula (Fig. 2.12 c,d).

Sulle immagini test, la rete neurale predice gradienti orizzontali e verticali che formano campi vettoriali o "paths". Seguendo questi campi vettoriali, ogni pixel appartenente ad una cellula deve essere condotto al suo centro. Raggruppando i pixel che sono attratti dallo stesso centro si ottengono le segmentazioni delle singole cellule (Fig. 2.12 e,f). La forma delle cellule è poi ulteriormente ridefinita rimuovendo i pixel che hanno la probabilità di appartenenza ad una cellula (cell probability) minore di 0.5.

2.4.1 Implementazione

La rete neurale che predice il campo vettoriale dei flussi (flow field) è basata su una generica U-Net (Fig. 2.12 d). Nella fase di upsampling è stato scelto di sfruttare una semplice somma delle feature per integrare il corrispondente strato di downsampling connesso tramite le connessioni skip. Questa scelta porta ad una riduzione dei parametri rispetto alla concatenazione delle feature. Inoltre vengono anche sostituiti i blocchi standard della U-Net con i blocchi residuali della ResNet, dato che migliorano le performance e permettono di raddoppiare la profondità della rete. Nel più piccolo strato di convoluzione viene usato un pooling con media globale così da ricavare il cosiddetto

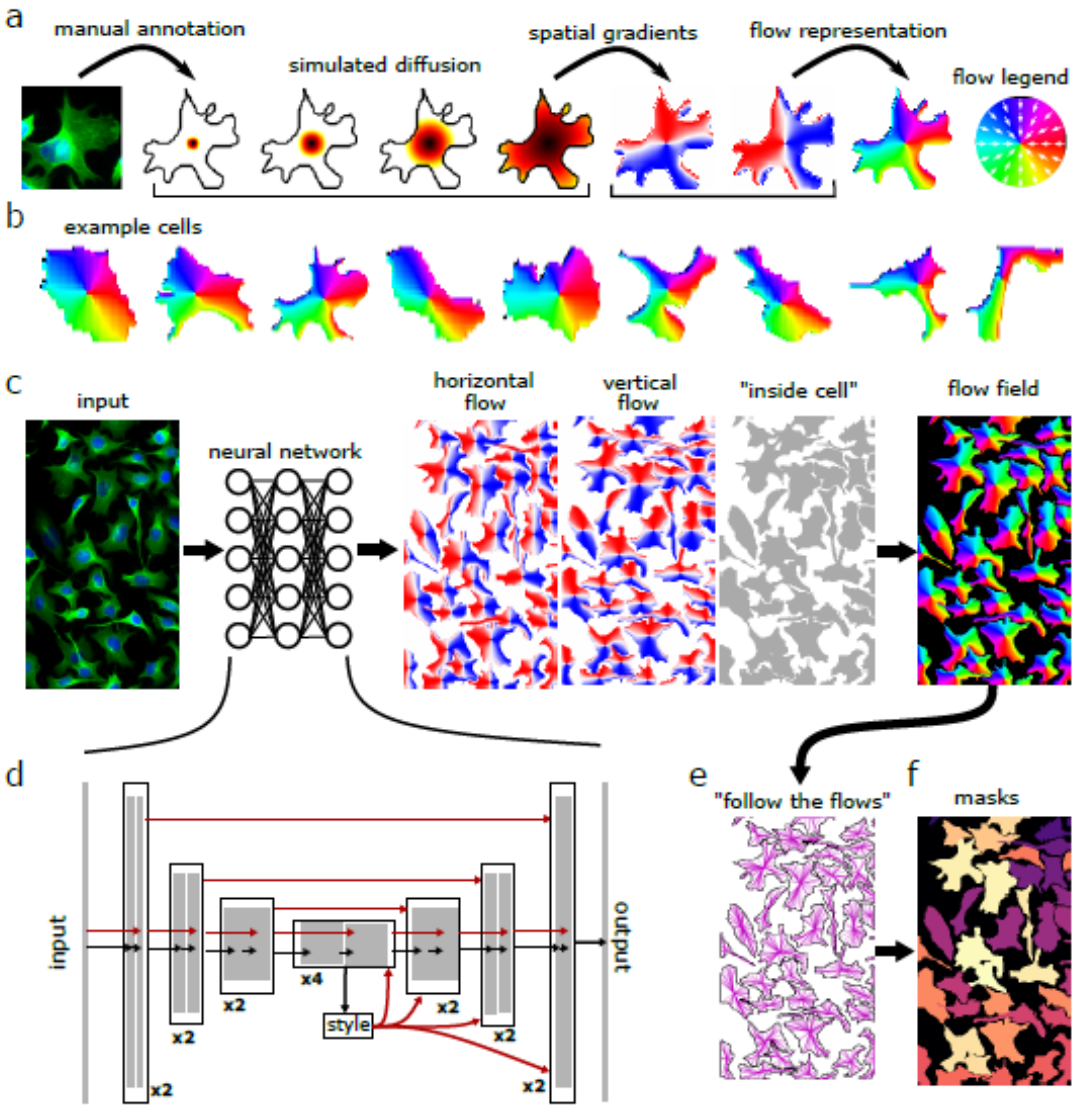


Figura 2.12: Architettura del modello Cellpose

"style" dell'immagine. Come premesso, immagini con stili diversi andrebbero processate in modo diverso. Connettere con i layer di upsampling (Fig. 2.12 d) il vettore di style permette potenzialmente di riaggiustare il resto della rete, adeguandolo al tipo di immagine.

Il modello è stato addestrato utilizzando un dataset vario, unendo tipologie diverse di immagini e utilizzando poi un metodo di clustering (t-SNE) sul vettore style per classificarle.

La versatilità di questo modello è il suo principale vantaggio, oltre alla possibilità, come con StarDist, di migliorare le prestazioni cambiando solo qualche parametro.

Capitolo 3

Segmentazione di immagini istologiche

Nel seguente capitolo e nel prossimo verranno esposti i risultati ottenuti applicando i due modelli descritti nel precedente a specifici dataset.
Nella prima parte del progetto è stato considerato il dataset pubblico "CAMEL" di Kaggle [17], che verrà presentato in questo capitolo nel paragrafo 3.3.1, mentre, la seconda parte di sperimentazione, è avvenuta in collaborazione con il dipartimento di Fisica Medica dell'AUSL di Reggio Emilia, sfruttando un dataset costituito da pazienti reali.
In entrambi i casi l'obiettivo è stato quello di segmentare i nuclei delle cellule di immagini istologiche con colorazione *ematossilina-eosina (H&E)*, che è lo standard per la diagnosi microscopica in ambito istopatologico. In nessun caso era presente un ground truth con annotazioni delle immagini del dataset; per questo si è scelto di procedere con una segmentazione che distinguesse soltanto i nuclei dal background, senza distinzione della tipologia di cellule presenti.
Inoltre per avere alcune maschere con cui confrontare i risultati della segmentazione è stato usato il programma QuPath [18] che permette di segmentare manualmente le immagini, aiutandosi anche con una prima segmentazione automatica che può essere modificata.

3.1 Metriche di valutazione

Per la valutazione dei metodi e dei parametri usati sono state usate le metriche di seguito esposte. In particolare, le prime due sono metriche *pixel-wise* che quindi confrontano pixel per pixel se le label sono le stesse. Poi è presentata una metrica a livello di oggetto che valuta se un oggetto sia stato o meno individuato correttamente e, infine, è presente una metrica combinata che sfrutta un prodotto tra una metrica pixel-wise e una a livello di oggetti.

1. **Intersection over Union (IoU)**: è chiamata anche indice di Jaccard e misura la sovrapposizione tra maschera binaria predetta e ground truth usando la formula

$$IoU = \frac{|P \cap G|}{|P \cup G|}$$

dove P è l'insieme dei pixel con la label di oggetto e G è l'insieme dei pixel oggetto nel ground truth. La IoU è compresa tra 0 e 1, dove 0 indica la totale assenza di sovrapposizione mentre 1 la perfetta coincidenza.

2. **Dice**: misura come l'IoU il livello di sovrapposizione dei due insiemi di pixel ma da più peso all'intersezione

$$Dice = \frac{2|P \cap G|}{|P| + |G|}$$

Il Dice risulta essere più stabile con oggetti piccoli e penalizza meno gli sbilanciamenti tra classi, per questo è spesso usato in ambito medico dove la classe del background è la prevalenza. Con cellule piccole e molto sfondo, infatti, l'unione può penalizzare troppo nell'IoU mentre il Dice compensa raddoppiando l'intersezione.

3. **F1 Score**: A differenza dei due precedenti, F1 score si concentra sui singoli oggetti, distinguendo tra oggetti rilevati correttamente, persi o falsi. Si etichettano gli oggetti con componenti connesse, per ogni oggetto della previsione viene poi calcolata la IoU con tutti quelli del ground truth; se una IoU supera 0,5 l'oggetto si considera correttamente rilevato (TP - True Positive) altrimenti viene classificato

come falso (FP - False Positive). Vengono classificati poi come falsi negativi (FN - False Negative) gli oggetti del ground truth non rilevati. Lo score F1 viene infine calcolato come

$$F1 = \frac{TP}{TP + FP + FN}$$

Anche in questo caso il valore 1 corrisponde alla rilevazione perfetta.

4. **Panoptic Quality (PQ)**: è una metrica complessiva che valuta contemporaneamente l'aspetto di segmentazione e di riconoscimento degli oggetti. La formula è data da

$$PQ = \frac{\sum_{(p,g)\in TP} IoU(p,g)}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|} = \frac{\sum_{(p,g)\in TP} IoU(p,g)}{|TP|} \frac{|TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}$$

che può essere vista come prodotto di una IoU fatta solo sui TP e di una versione leggermente modificata della F1.

3.2 Confronto StarDist - Cellpose

Inizialmente sono state scelte alcune immagini, provenienti da dataset diversi, per un veloce confronto tra StarDist e Cellpose e per capire quali parametri ottimizzare per migliorare le prestazioni di entrambi. Le valutazioni fatte sono state validate dai patologi dell'AUSL di Reggio Emilia a cui sono state presentate le segmentazioni ottenute.
Le immagini prese in considerazione sono:

1. Patch (senza GT) proveniente dal dataset CAMEL di Kaggle (descritto in dettaglio nel paragrafo 3.3.1) con tessuto tumorale (mesotelioma).
   Dimensioni: 224x224 (Fig. 3.1).

2. Patch con GT proveniente dal dataset TNBC [19] usato per l'addestramento del modello di Stardist specifico per i nuclei "2D_versatile_he"
   Dimensioni: 512x512 (Fig. 3.2)

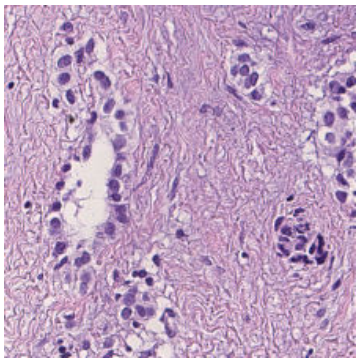


Figura 3.1: CAMEL - mesotelioma (senza GT)

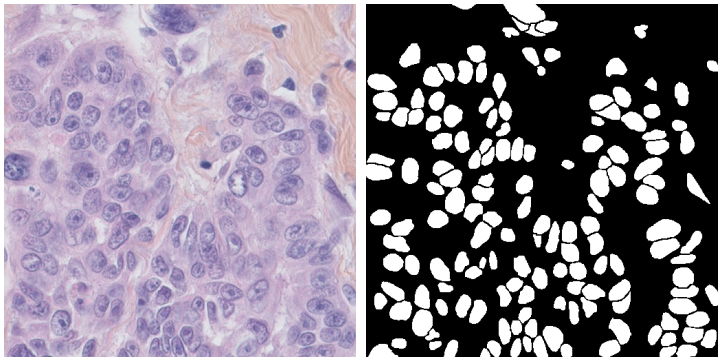


Figura 3.2: TNBC - HE102 (con GT)

3. Patch con GT proveninete da dataset Nulnseg [20] con tessuto polmonare, "human_lung_6".
   Dimensioni: 512x512 (Fig. 3.3)

4. Patch con GT proveniente dal dataset Nulnseg con tessuto polmonare, "human_lung_12".
   Dimensioni: 512x512 (Fig. 3.4)

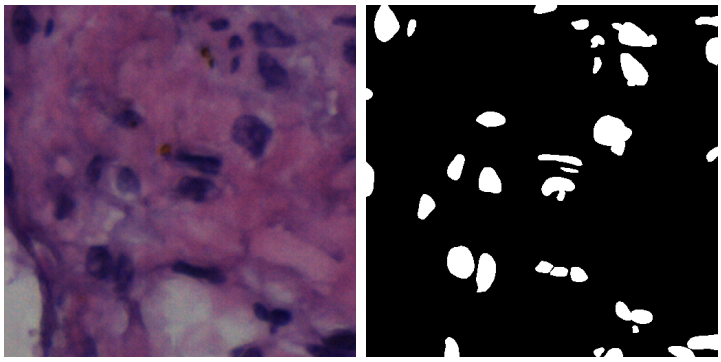


Figura 3.3: Nulnseg - lung 6 (con GT)

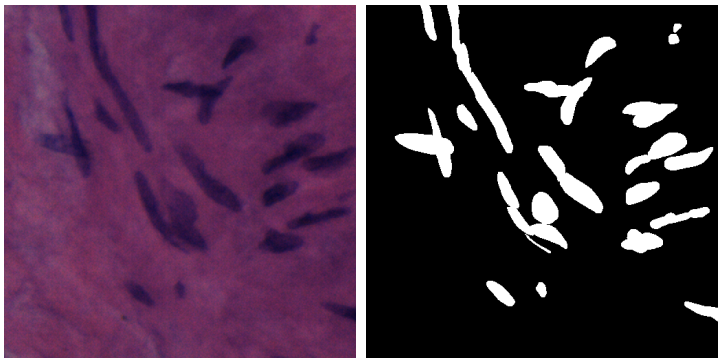


Figura 3.4: Nulnseg - lung 12 (con GT)

3.2.1 CAMEL - Mesotelioma

StarDist

Per primo è stato testato il modello StarDist preaddestrato "2D_versatile_he" [11], specifico per immagini istologiche, per il quale è stato necessario un resize alla dimensione 512x512 supportata dal modello.

In figura 3.5 è riportata la maschera ottenuta segmentando con i parametri standard del modello. Anche senza l'ausilio del ground truth, non presente per questa immagine, si può notare che mancano diverse cellule in questa segmentazione. In assenza di metriche per la valutazione, sono riportate in figura 3.6 le sovrapposizioni all'immagine originale (in scala di grigi), rispettivamente, dei poligoni usati per l'individuazione delle cellule e della maschera delle label.

Si è deciso, quindi, di variare i due parametri principali del modello che sono la

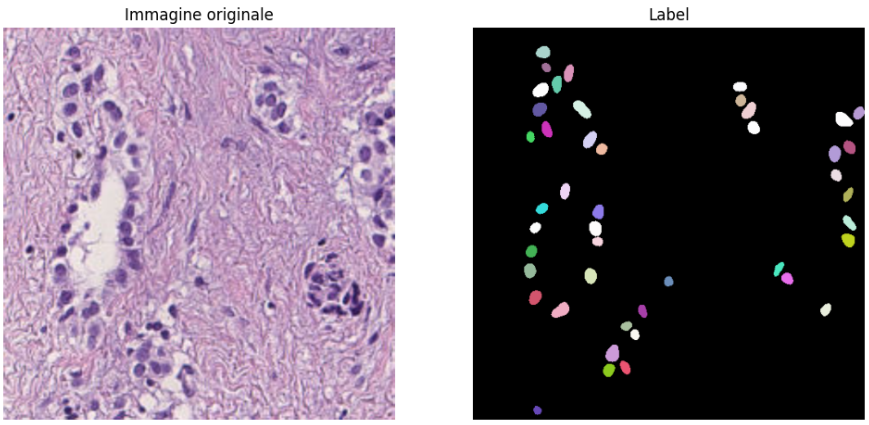


Figura 3.5: Mesotelioma resize 512x512 - Maschera ottenuta da StarDist con parametri standard

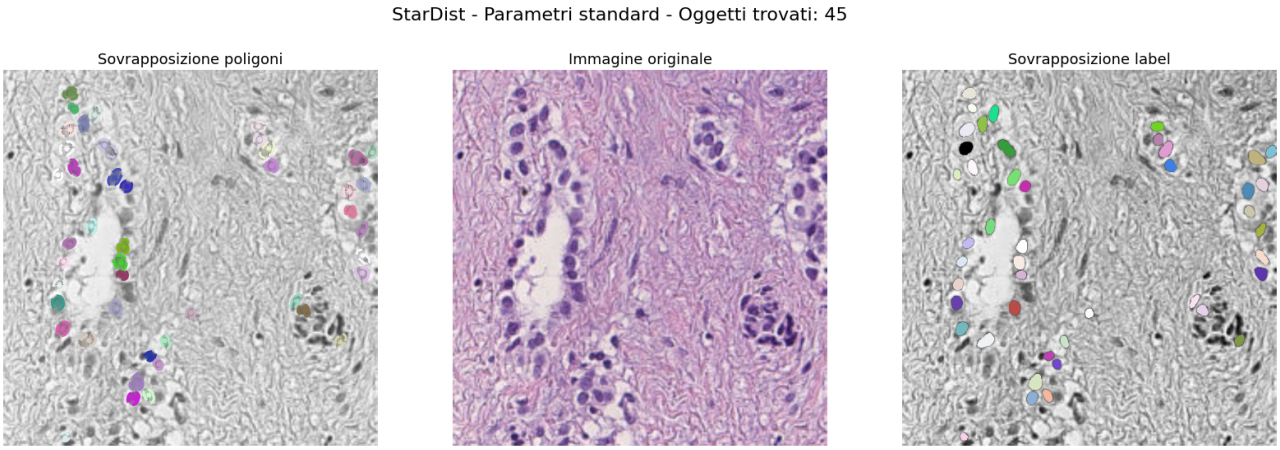


Figura 3.6: Mesotelioma resize 512x512 - Cellule individuate da StarDist con parametri standard

probability threshold e la NMS threshold che riguardano, rispettivamente, la soglia di object probability con cui vengono accettati i poligoni e la soglia per la NMS.
Per queste valutazioni qualitative iniziali, è stato deciso di calcolare la segmentazione per ogni combinazione di valori nei vettori:

```
prob_thresh_vett = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8] #0.692478
nms_thresh_vett = [0.3, 0.4, 0.5] # 0.3
```

I vettori sono stati scelti considerando punti equispaziati nell'intervallo di valori più usati. A fianco dei vettori è riportato il valore standard del modello "2D_versatile_he". In figura 3.7 sono riportati alcuni dei risultati ottenuti; in particolare, il caso con `prob_thresh = 0.1` e `nms_thresh = 0.4` è risultato quello che ha trovato più cellule: ne ha individuate 122 contro le 45 trovate con i parametri standard. Da solo questo dato non garantisce una migliore segmentazione perchè, con probability threshold molto basso, il rischio è di non scartare cellule individuate male. Comunque, in questo caso specifico, la segmentazione è buona perchè non sembra individuare cellule che non esistono, al massimo risulta essere non troppo preciso sui bordi. In figura 3.7 sono riportati anche un altro caso segmentato bene con `prob_thresh = 0.2` e `nms_thresh = 0.4` e il caso peggiore con `prob_thresh = 0.8` e `nms_thresh = 0.5`. Per quest'ultimo si può notare come aumentando il probability threshold il numero di cellule individuate tenda a calare dato che la soglia di accettazione diventa più alta. Al contrario della probability threshold, è più difficile notare cosa cambi al variare della NMS threshold. Visivamente,

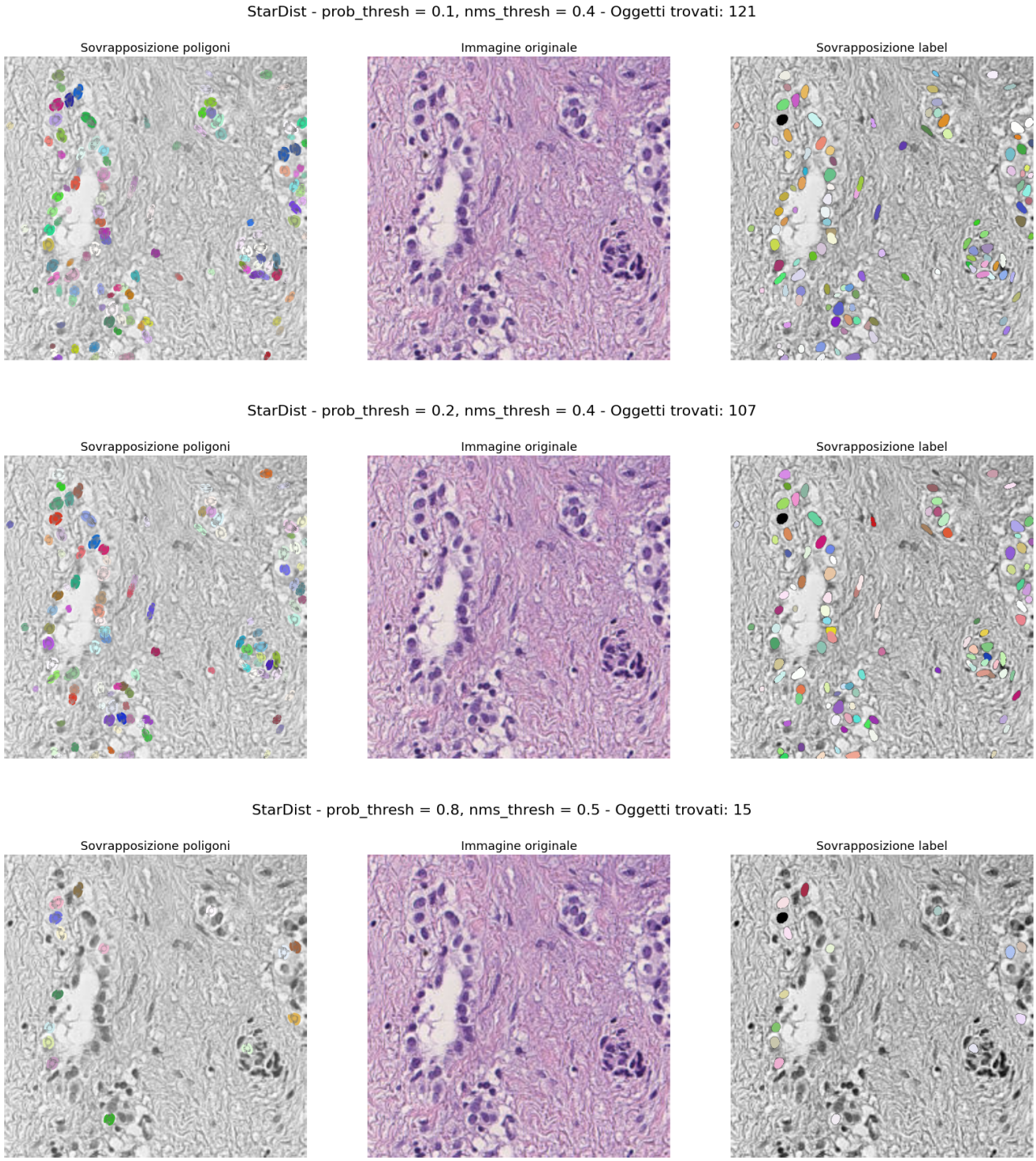


Figura 3.7: Mesotelioma resize 512x512 - Cellule individuate da StarDist con parametri personalizzati

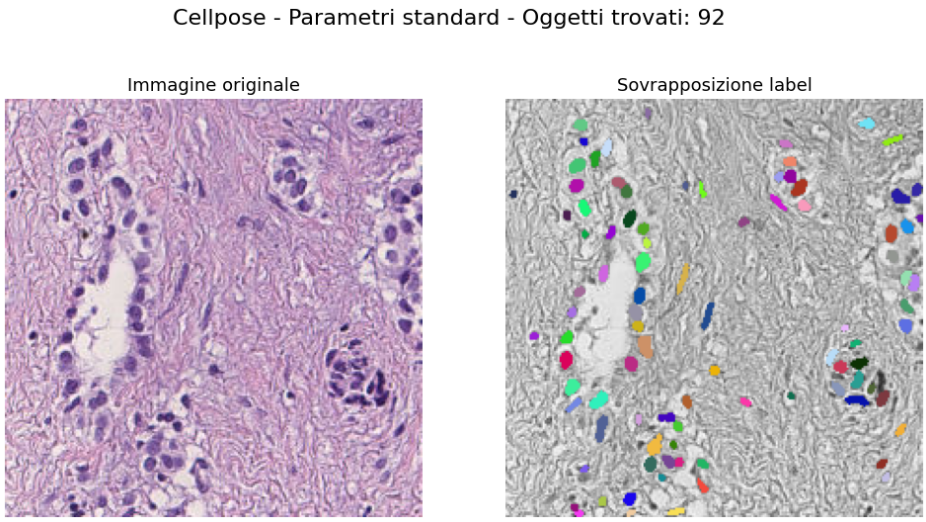


Figura 3.8: Mesotelioma - Cellule individuate da Cellpose con parametri standard

ciò che viene modificato da quest'ultima è il livello di aggregazione delle cellule: una NMS threshold molto bassa porta ad avere più cellule piccole raggruppate, mentre una NMS threshold alta porta ad avere cellule singole più grosse.
Ultima osservazione per StarDist, che diventerà più evidente in seguito, riguarda la sua difficoltà nel gestire le cellule di forma "allungata"; quando vengono identificate sono quasi sempre rappresentate come unione di più cellule piccole. Questo fatto è intrinseco del metodo che, per definizione, costruisce poligoni convessi a stella e che quindi è più portato a riconoscere forme vicine alla circonferenza.

Cellpose

Un ragionamento analogo è stato fatto anche per Cellpose. Inizialmente è stato usato il modello standard che può essere usato anche sull'immagine originale 224x224 senza dover cambiare le dimensioni. Gli unici parametri non di default fissati sono `Augment=True` che valuta l'immagine più volte con rotazioni e ribaltamenti e `min_size=5` per non perdere le cellule più piccole. In figura 3.8 è riportata la sovrapposizione delle label ottenute all'immagine originale. Si può notare che questa segmentazione risulta essere migliore di quella standard di StarDist ottenuta al passo precedente. Anche se le cellule trovate sono meno della migliore segmentazione di StarDist, Cellpose risulta essere più preciso, anche per quanto riguarda le cellule "allungate". Infatti dove StarDist usa più cellule per rappresentarne una allungata, Cellpose ne usa una sola.

Anche per Cellpose sono stati modificati alcuni dei parametri più importatanti. In questo caso la scelta è ricaduta sul diametro delle cellule che, di default, è calcolato usando una funzione interna al metodo e sulla flow threshold, solitamente di 0,4, che è legata all'accettazione dei campi vettoriali della mappa topologica creata. Una flow threshold bassa permetterà di individuare più oggetti che potrebbero avere però forme irrealistiche, al contrario una flow threshold alta garantisce più precisione ma meno oggetti.
Per una prima analisi qualitativa, i parametri sono stati scelti iterativamente a partire dai vettori

```
diam = [5,10,15,20]
flow_thresh = [0.2,0.4,0.6,0.8]
```

In questo caso la segmentazione migliore è risultata essere quella iniziale con il diametro calcolato dalla funzione di Cellpose. In figura 3.9 sono riportate la migliore e la peggiore delle segmentazioni ottenute, da notare che anche nel caso peggiore Cellpose individua comunque 40 oggetti dimostrandosi tendenzialmente più stabile di StarDist.

3.2.2 TNBC - HE102

Questa immagine è stata scelta tra quelle usate per addestrare il modello di StarDist "2D_versatile_he", con lo scopo di verificare se, anche in questo caso, le performance di StarDist con parametri standard risultano essere peggiori di quelle di Cellpose standard. Secondariamente, verrà effettuata, per entrambi i metodi, la variazione dei parametri come al caso precedente.
Per l'immagine HE102 è presente il ground truth (GT) ed è quindi possibile utilizzare le metriche presentate nel paragrafo 3.1 per un reale confronto sulle prestazioni.

StarDist

In figura 3.10 è riportata la segmentazione di Stardist con parametri standard. Si può notare come, nonostante l'immagine sia stata usata per il training, la segmentazione non sia estremamente precisa e perda diversi oggetti. Sfruttando metriche e GT si ottengono i seguenti risultati

Seguendo lo stesso procedimento dell'immagine di mesotelioma si procede con la personalizzazione dei parametri.
Nella seguente tabella sono riportate tutte le metriche ottenute dal confronto con il GT al variare dei parametri.

In grassetto sono stati evidenziati i risultati migliori ottenuti: per Dice e IoU il risultato ottimo è dato da `prob_thresh = 0.1` e `nms_thresh = 0.5` mentre per F1 ce ne sono tre diversi ma solo con `prob_thresh = 0.2` e `nms_thresh = 0.4` si ha anche l'ottimo per PQ. Da notare che in tutti questi casi i risultati sono migliori di quelli

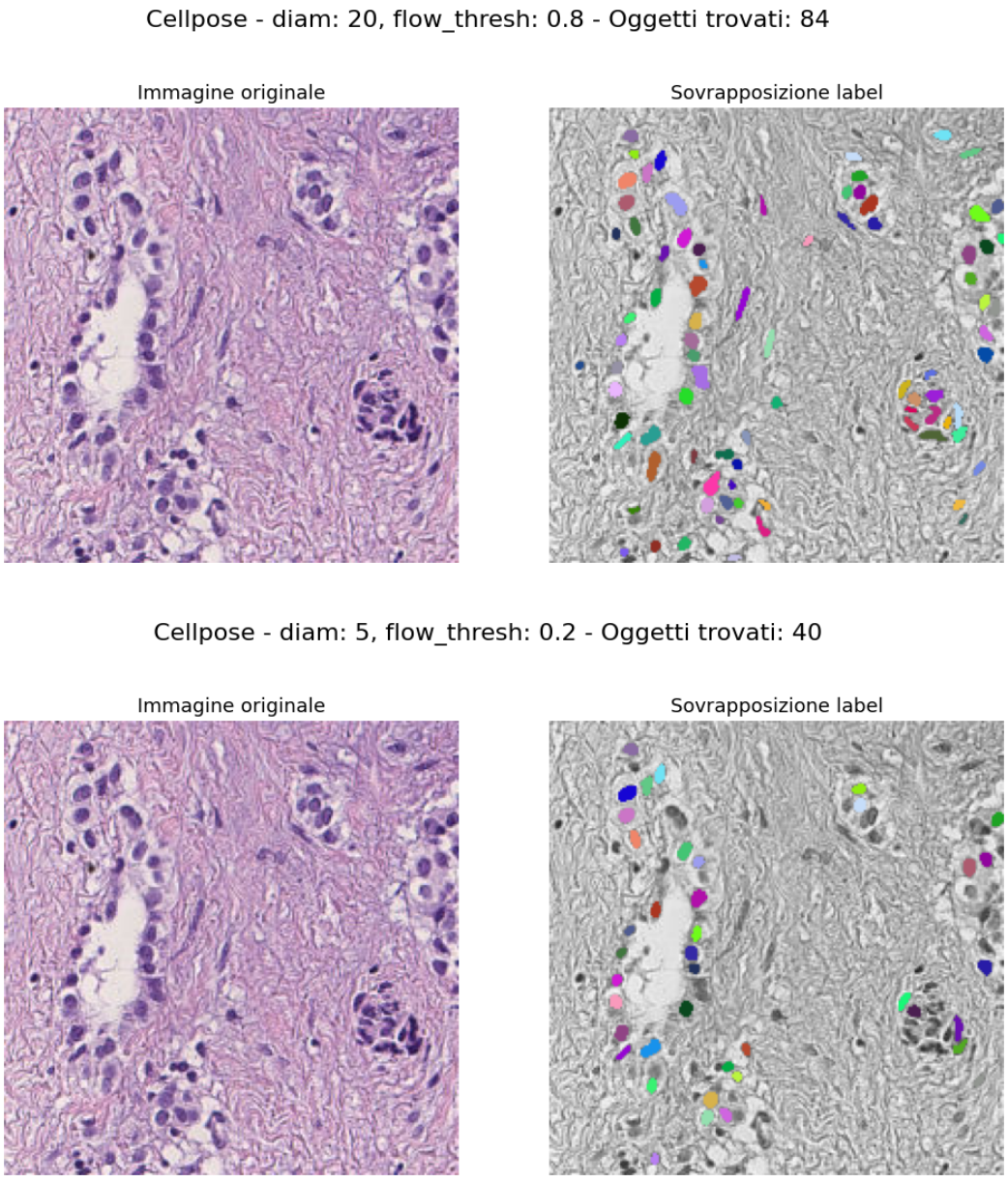


Figura 3.9: Mesotelioma - Cellule individuate da Cellpose con parametri personalizzati

Metriche	IoU	Dice	F1 [TP, FP, FN]	PQ
Standard	0.6239	0.7684	0.5590 [90, 6, 65]	0.3487

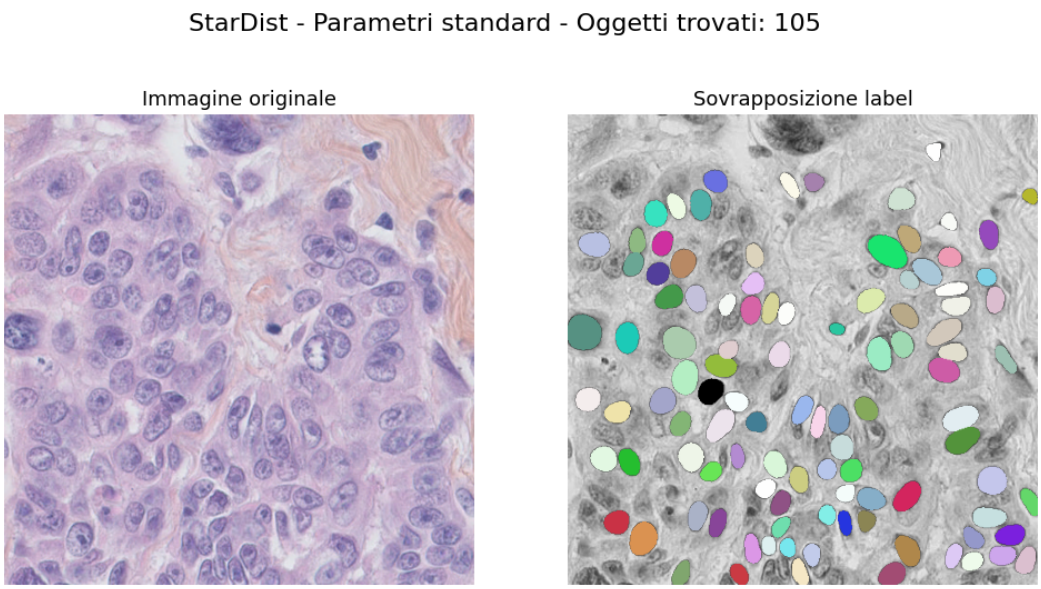


Figura 3.10: HE102 - Cellule individuate da StarDist con parametri standard

ottenuti con parametri standard.
In figura 3.11 sono riportati i confronti tra il GT e le maschere ottenute con i parametri standard e le diverse segmentazioni ottime trovate.

Cellpose

Analogamente a quanto appena fatto per Stardist, si riportano le metriche ottenute con Cellpose e in figura 3.12 la sovrapposizione all'immagine originale delle label.
Da questi risultati si deduce che non è molto precisa l'individuazione e la distinzione dei singoli oggetti, rappresentata da F1, ma a livello di sovrapposizione binaria con il GT la segmentazione è decisamente buona. Anche in questa immagine si conferma Cellpose come segmentazione migliore con le impostazioni di default.

Analogamente a Stardist al variare dei parametri si possono identificare le segmentazioni migliori per le singole metriche. Dalla tabella che segue si evince che i migliori per IoU e Dice siano `diam = 5` e `flow_thresh = 0.4` mentre per F1 e PQ `diam = 10` e `flow_thresh = 0.4`. In entrambi i casi i risultati ottenuti sono migliori di Cellpose standard, validando la ricerca di parametri ottimi anche per questo metodo.

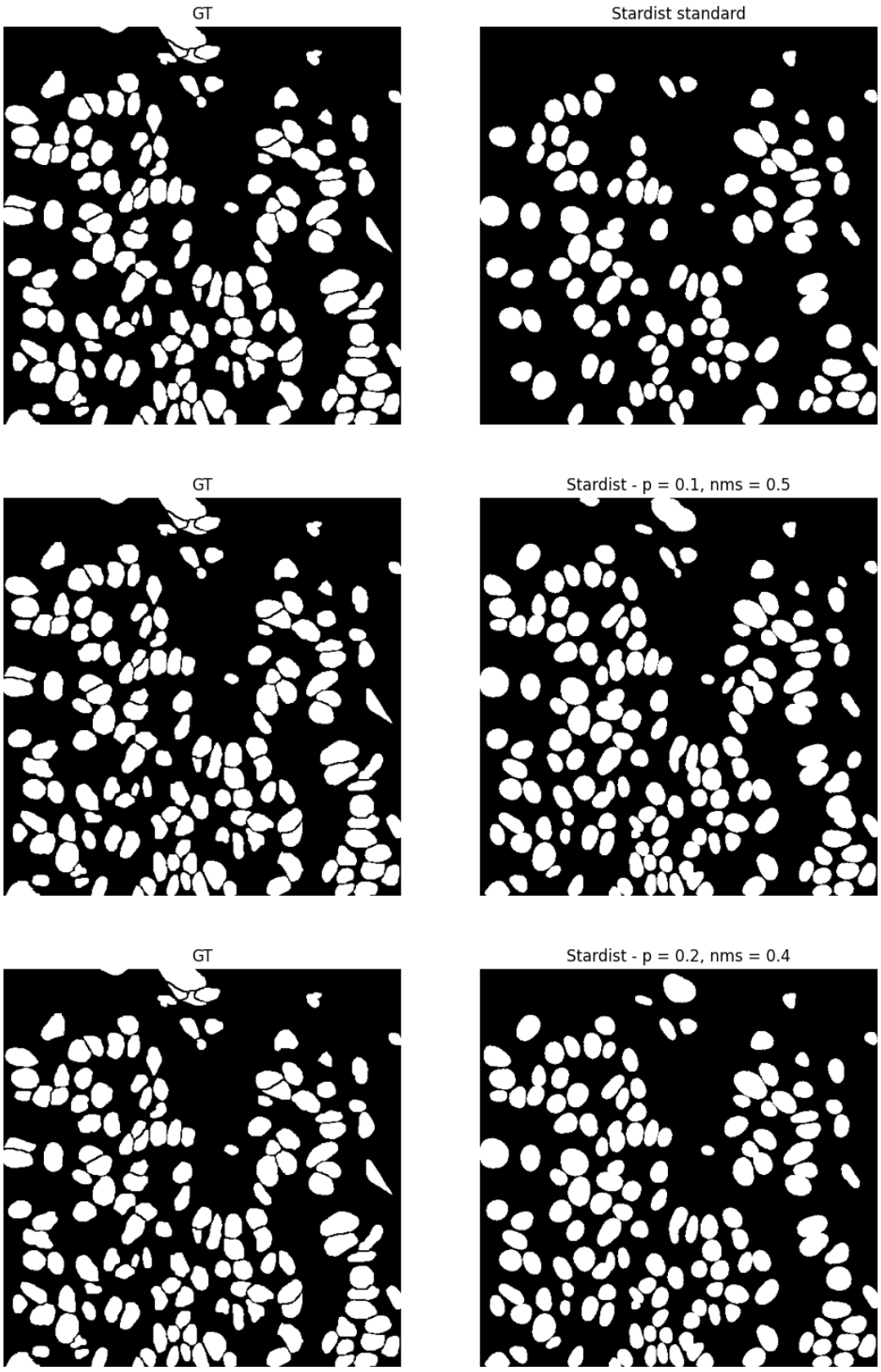


Figura 3.11: HE102 - Confronto GT e maschere predette con StarDist

p, NMS	IoU	Dice	[F1 TP, FP, FN]	PQ
0.1, 0.3	0.7633	0.8658	0.6437 [112,19,43]	0.4913
0.1, 0.4	0.7634	0.8658	0.6437 [112,19,43]	0.4914
0.1, 0.5	**0.7716**	**0.8711**	0.6437 [112,19,43]	0.4967
0.2, 0.3	0.7646	0.8666	**0.6627 [112,14,43]**	0.5067
0.2, 0.4	0.7647	0.8667	**0.6627 [112,14,43]**	**0.5068**
0.2, 0.5	0.7646	0.8666	**0.6627 [112,14,43]**	0.5067
0.3, 0.3	0.7494	0.8567	0.6471 [110,15,45]	0.4849
0.3, 0.4	0.7509	0.8577	0.6471 [110,15,45]	0.4859
0.3, 0.5	0.7508	0.8577	0.6471 [110,15,45]	0.4858
0.4, 0.3	0.7407	0.8510	0.6331 [107,14,48]	0.4690
0.4, 0.4	0.7422	0.8520	0.6331 [107,14,48]	0.4699
0.4, 0.5	0.7422	0.8520	0.6331 [107,14,48]	0.4699
0.5, 0.3	0.7210	0.8379	0.6095 [103,14,52]	0.4394
0.5, 0.4	0.7210	0.8379	0.6095 [103,14,52]	0.4394
0.5, 0.5	0.7225	0.8389	0.6095 [103,14,52]	0.4403
0.6, 0.3	0.6906	0.8170	0.5808 [97,12,58]	0.4011
0.6, 0.4	0.6906	0.8170	0.5808 [97,12,58]	0.4011
0.6, 0.5	0.6906	0.8170	0.5808 [97,12,58]	0.4011
0.7, 0.3	0.6239	0.7684	0.5590 [90,6,65]	0.3487
0.7, 0.4	0.6239	0.7684	0.5590 [90,6,65]	0.3487
0.7, 0.5	0.6239	0.7684	0.5590 [90,6,65]	0.3487
0.8, 0.3	0.3716	0.5418	0.3376 [53,2,102]	0.1254
0.8, 0.4	0.3716	0.5418	0.3376 [53,2,102]	0.1254
0.8, 0.5	0.3716	0.5418	0.3376 [53,2,102]	0.1254

Metriche	IoU	Dice	F1 [TP, FP, FN]	PQ
Standard	0.8220	0.9023	0.5465 [94, 17, 61]	0.4492

Rispetto a StarDist sono migliori gli indici di sovrapposizione (IoU e Dice) mentre lievemente peggiori quelli di riconoscimento dei singoli oggetti (F1).

In figura 3.13 sono presentati i confronti tra il GT e le maschere migliori.

3.2.3 Nulnseg - Lung

In questa sezione sono raccolte le ultime due immagini che appartengono allo stesso dataset. Sono state scelte perchè il tessuto polmonare presenta cellule di forma diversa e il dataset CAMEL, che verrà analizzato in seguito, tratta di una particolare forma tumorale propria del polmone, è quindi utile avere qualche ulteriore analisi.

Per entrambe le immagini si procederà direttamente ad un confronto tra i due metodi

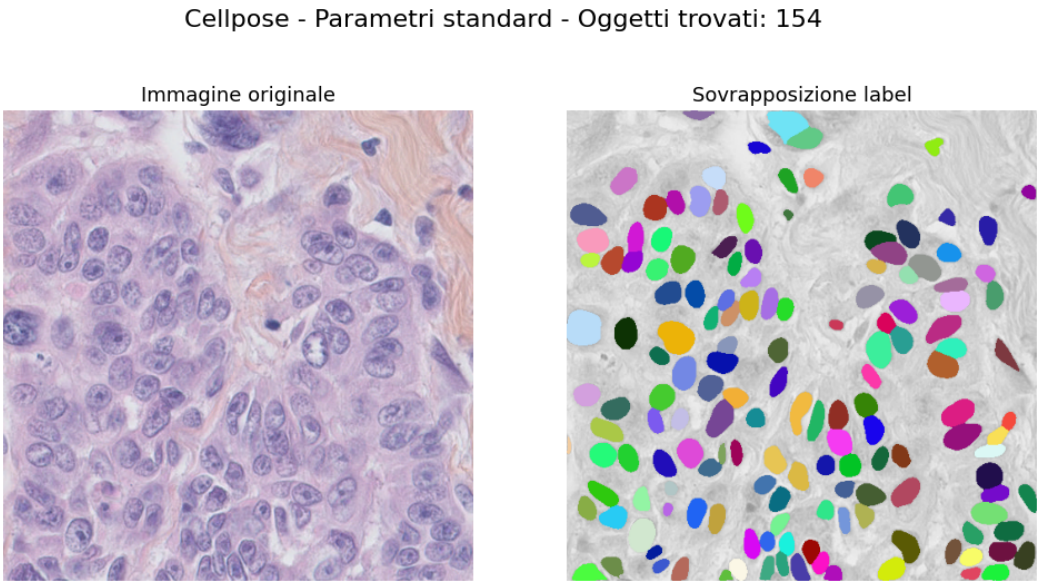


Figura 3.12: HE102 - Cellule individuate da StarDist con parametri standard

Diameter, Flow	IoU	Dice	F1 [TP, FP, FN]	PQ
5, 0.2	0.7383	0.8495	0.5833 [98,13,57]	0.4307
5, 0.4	0.8067	0.8930	**0.6271 [111,22,44]**	**0.5059**
5, 0.6	0.8129	0.8968	0.6201 [111,24,44]	0.5041
5, 0.8	0.8117	0.8961	0.6044 [110,27,45]	0.4906
10, 0.2	0.7947	0.8856	0.5988 [100,12,55]	0.4758
10, 0.4	**0.8277**	**0.9057**	0.5988 [103,17,52]	0.4957
10, 0.6	0.8243	0.9037	0.5852 [103,21,52]	0.4824
10, 0.8	0.8272	0.9054	0.5819 [103,22,52]	0.4813
15, 0.2	0.7924	0.8842	0.5799 [98,14,57]	0.4595
15, 0.4	0.8247	0.9039	0.5600 [98,20,57]	0.4618
15, 0.6	0.8262	0.9048	0.5568 [98,21,57]	0.4601
15, 0.8	0.8269	0.9052	0.5537 [98,22,57]	0.4578
20, 0.2	0.8004	0.8891	0.5680 [96,14,59]	0.4546
20, 0.4	0.8229	0.9028	0.5486 [96,20,59]	0.4514
20, 0.6	0.8253	0.9043	0.5393 [96,23,59]	0.4451
20, 0.8	0.8253	0.9043	0.5393 [96,23,59]	0.4451

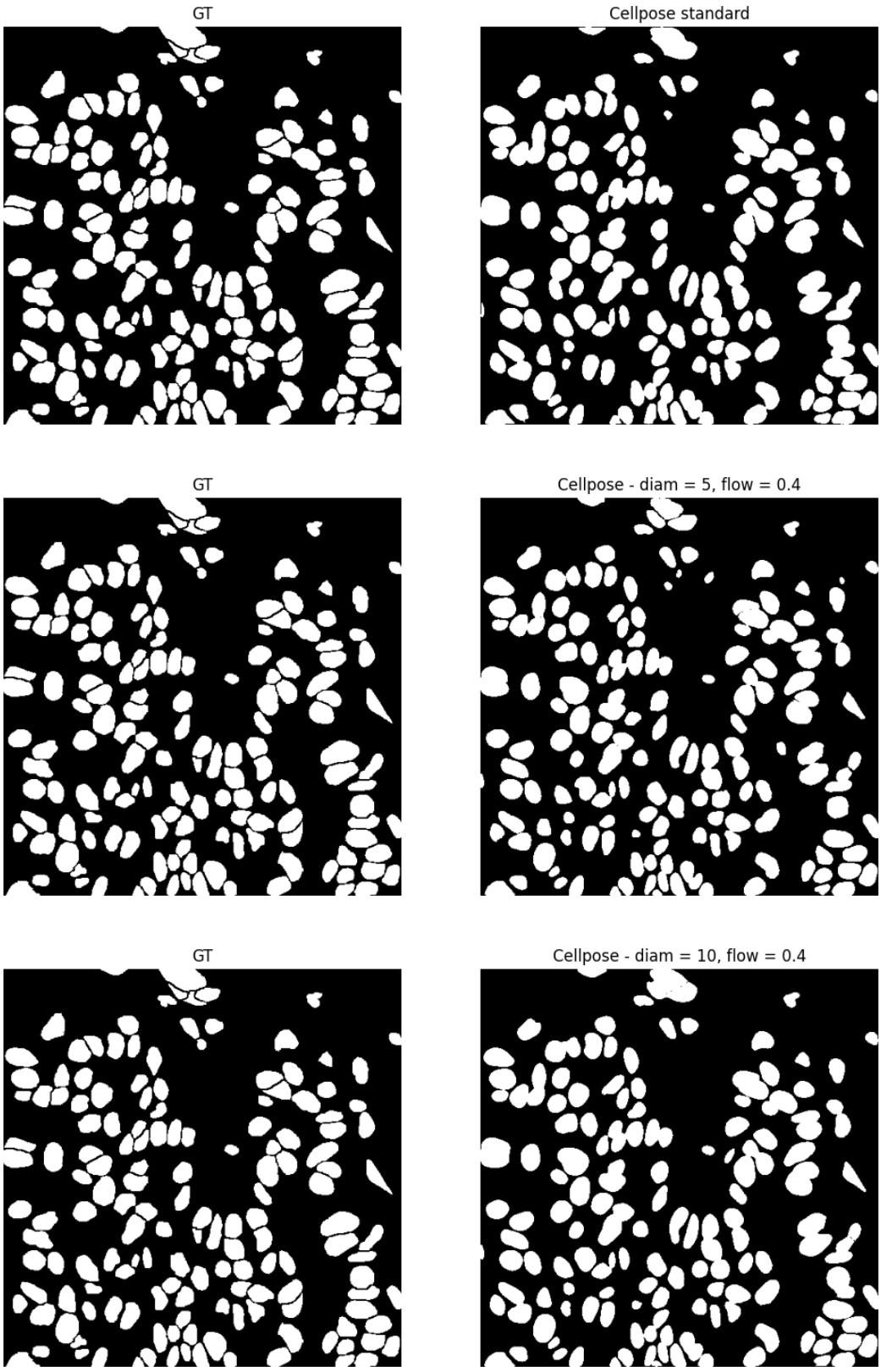


Figura 3.13: HE102 - Confronto GT e maschere predette con Cellpose

esattamente come fatto al paragrafo precedente. La prima, Lung_6, contiene cellule di varie dimensioni e forme, mentre la seconda, Lung_12, ha prevalentemente cellule allungate. L'obiettivo è quindi, per la prima immagine, osservare il comportamento dei metodi nell'adattarsi a riconoscere forme diverse nella stessa immagine, mentre per la seconda, specializzarsi nel riconoscere un certo tipo di forma difficile.

Lung_6

In figura 3.14 sono riportate entrambe le segmentazioni standard. Leggendo le metriche riportate di seguito si nota subito come Cellpose sia ancora una volta vincente. Anche

Metodo	IoU	Dice	F1 (TP, FP, FN)	PQ
Stardist	0.5081	0.6738	0.3684 [14,7,17]	0.1872
Cellpose	**0.7755**	**0.8736**	**0.7568 [28,6,3]**	**0.5869**

osservando le metriche ottenute dai modelli migliori al variare dei parametri il risultato non cambia. Questa notevole discrepanza tra le performance dei due metodi è dovuta alla buona abilità di Cellpose, a differenza di StarDist, di riuscire a riconoscere forme diverse anche senza dover modificare i parametri. Questo perchè alla base del modello non c'è la ricerca di una specifica forma ma di un campo vettoriale.

Metodo (parametri)	IoU	Dice	F1 (TP, FP, FN)	PQ
Stardist (0.4, 0.3)	0.6298	0.7729	0.5500 [22,9,9]	0.3464
Cellpose (15, 0.6)	**0.7938**	**0.8850**	**0.8000 [28,4,3]**	**0.6350**

Lung_12

Al contrario del caso precedente qui, pur restando Cellpose in netto vantaggio nelle metriche standard con la regolazione dei parametri, considerando sempre i casi migliori, la situazione diventa più equilibrata.

Metodo	IoU	Dice	F1 (TP, FP, FN)	PQ
Stardist	0.6208	0.7660	0.4815 [13,8,6]	0.2989
Cellpose	**0.7665**	**0.8678**	**0.5556 [15,8,4]**	**0.4259**

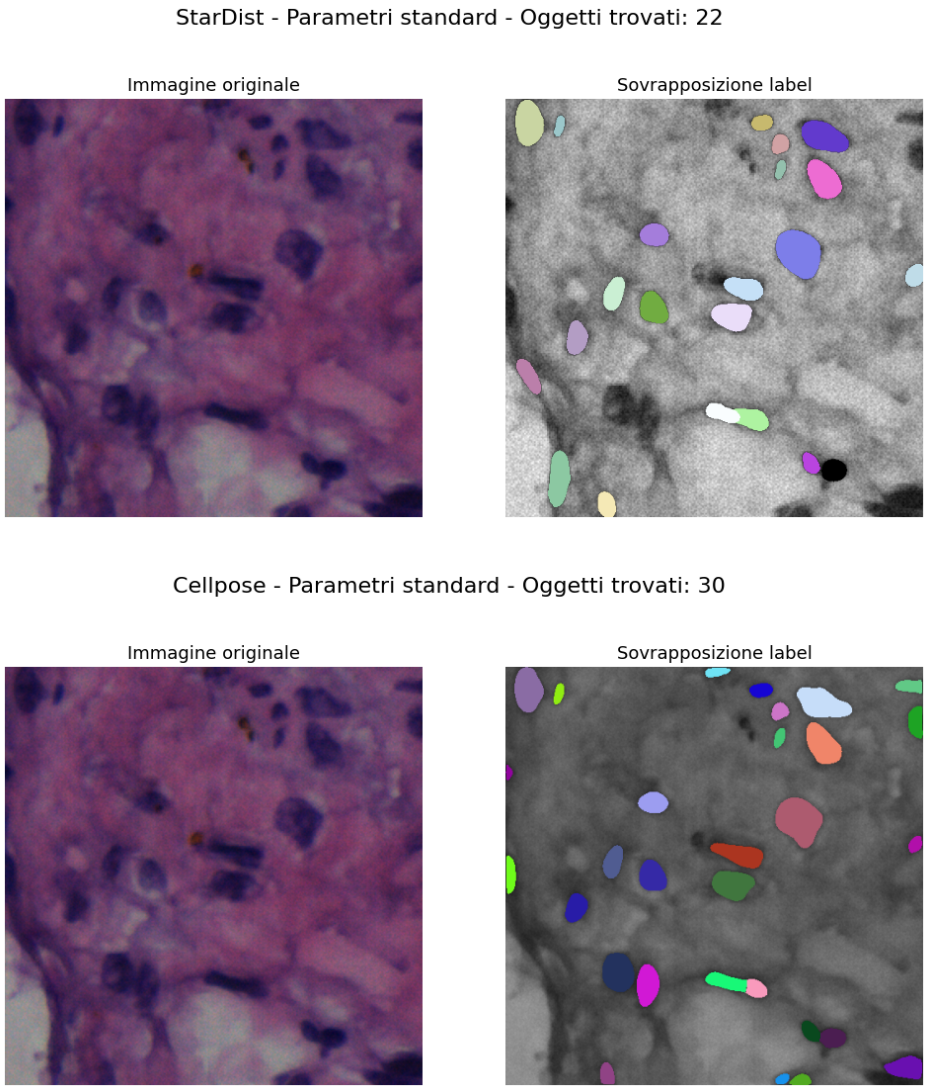


Figura 3.14: Lung_6 - Cellule individuate da StarDist e Cellpose con parametri standard

Ancora una volta vengono riportate le segmentazioni standard in figura 3.15. A queste immagini si aggiunge il confronto della maschera migliore di StarDist con il GT (3.16), da cui si può notare come, una volta ottimizzati i parametri, la maschera si avvicini bene ad approssimare anche le cellule più allungate. Abbassando molto la probability threshold vengono prese anche le forme allungate che sono meno regolari e alzando la NMS threshold vengono fuse con più facilità le cellule sovrapposte. Funziona bene anche perchè le cellule sono tutte della stessa tipologia: la combinazione di parametri per individuare queste potrebbe penalizzare altri tipi di cellule.

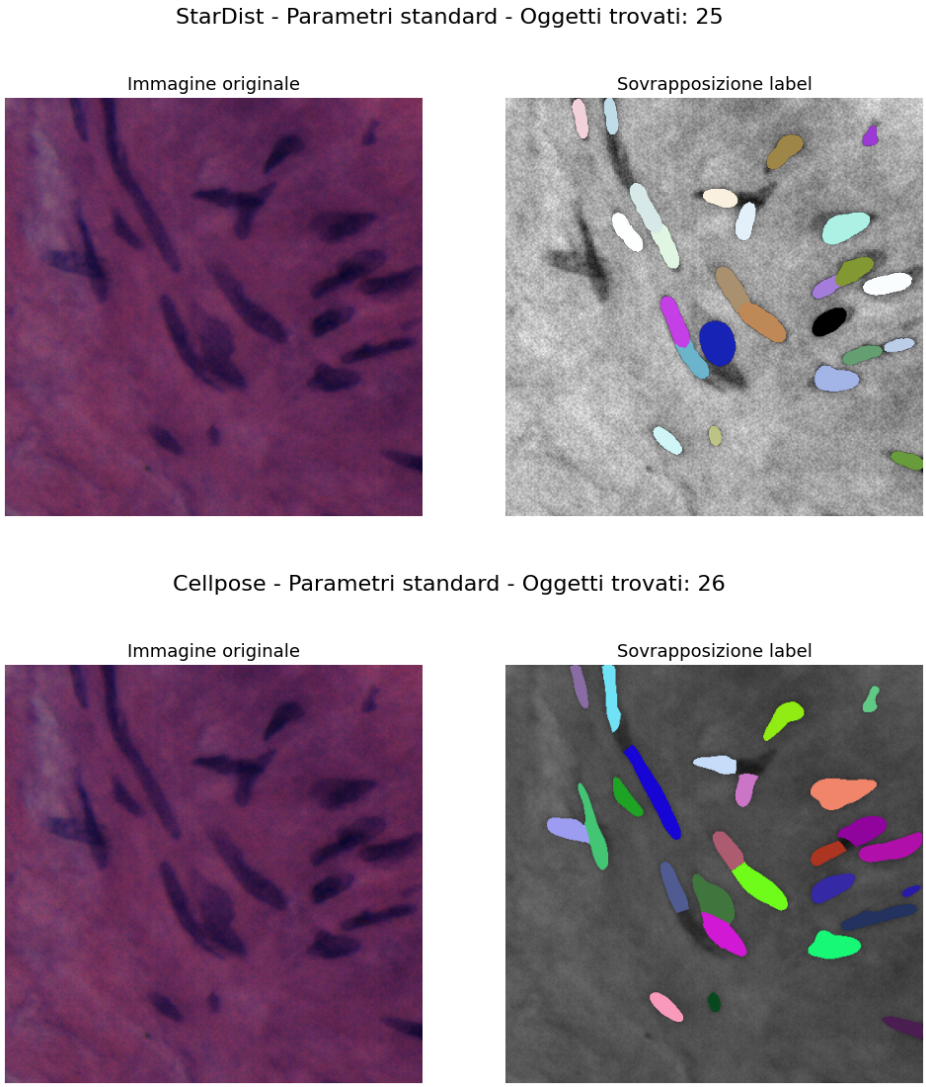


Figura 3.15: Lung_12 - Cellule individuate da StarDist e Cellpose con parametri standard

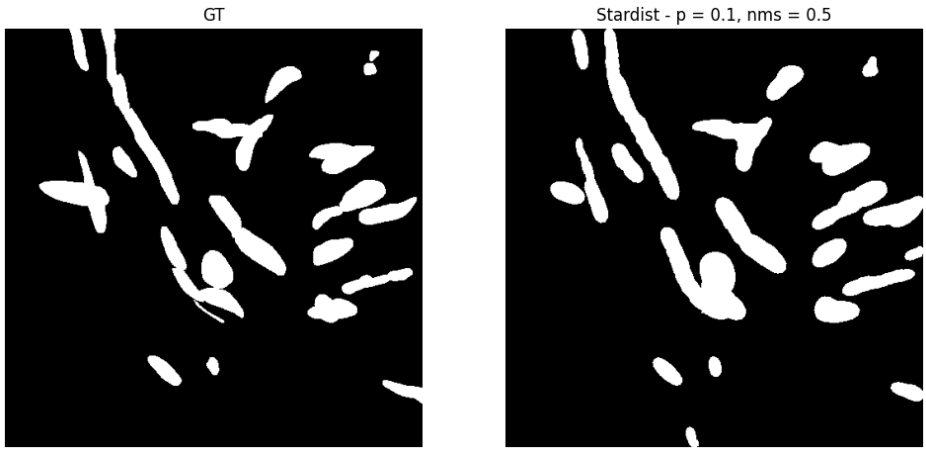


Figura 3.16: Lung_12 - Confronto GT e maschere predette con StarDist

Metodo (parametri)	IoU	Dice	F1 (TP, FP, FN)	PQ
Stardist (0.1, 0.5)	0.7352	0.8474	**0.7391 [17,4,2]**	0.5434
Cellpose (20, 0.6)	**0.7846**	**0.8793**	0.7083 [17,5,2]	**0.5558**

3.2.4 Valutazioni finali sui modelli

In conclusione è stato verificato che in tutti i casi il fine-tuning dei parametri principali dei due modelli permette un netto miglioramento delle prestazioni. Inoltre, grazie al parere dei patologi dell'AUSL di Reggio Emilia, è stato valutato che Cellpose possa essere il metodo migliore per segmentare grandi quantità di immagini istologiche anche con un fine-tuning fatto su poche patch con ground truth. Grazie alla sua abilità di generalizzazione Cellpose risulta infatti tendenzialmente più stabile su immagini nuove e mediamente garantisce anche performance migliori.

3.3 CAMEL dataset

3.3.1 Il dataset

Il dataset CAMEL è stato sviluppato con l'obiettivo di supportare la progettazione di pipeline di apprendimento automatico per la classificazione dei sottotipi istologici del mesotelioma pleurico maligno. In particolare, il dataset è stato concepito per consentire l'estrazione di dati di alta qualità a livello di patch a partire da immagini istopatologiche complete (Whole Slide Images, WSI), escludendo regioni contenenti artefatti, sfondo o informazioni ridondanti.

La sua realizzazione si inserisce nell'ambito del progetto del corso Machine Learning in Applications del Politecnico di Torino [21].

Il dataset è composto da 15 WSI, equamente distribuite tra i tre principali sottotipi di mesotelioma: epiteliode, sarcomatoide e bifasico (5 per ciascuna classe). Queste immagini sono state selezionate da un archivio più ampio di 117 WSI provenienti da pazienti reali (91 epiteliode, 21 bifasico, 5 sarcomatoide), sulla base della loro ricchezza in termini di patch informative e non ridondanti dopo il processo di filtraggio. La selezione è stata effettuata analizzando il numero di patch valide estratte da ciascuna slide. Un file di supporto (diagnosi.xls) fornisce la diagnosi istologica associata a ogni

WSI. Le WSI sono state acquisite mediante microscopio ottico e fornite dall'Ospedale San Luigi di Torino. La diagnosi e l'etichettatura delle immagini sono state effettuate da patologi esperti. Tutti i dati sono stati completamente anonimizzati, garantendo l'assenza di informazioni personali riconducibili ai pazienti.
Per garantire un'elevata qualità del segnale, ogni WSI è stata sottoposta a un processo di preprocessing che include l'estrazione di patch e il filtraggio del rumore mediante una funzione basata sulla saturazione, implementata in Python utilizzando le librerie OpenSlide [22] e OpenCV [23]. In particolare, le patch sono state estratte su griglia al livello 1 con dimensione pari a 224×224 pixel, mentre le regioni contenenti prevalentemente sfondo o aree bianche sono state rimosse tramite una soglia sulla saturazione media nello spazio colore HSV. Non sono state applicate tecniche di normalizzazione delle colorazioni; tuttavia, sono state introdotte diverse strategie di data augmentation (come ritagli casuali, rototraslazioni e rumore gaussiano) per aumentare la robustezza dei modelli al rumore.
Il dataset è destinato principalmente alla ricerca sulla classificazione dei sottotipi di mesotelioma in ambito di patologia digitale. Risulta particolarmente adatto per attività di estrazione di caratteristiche a livello di patch, apprendimento semi-supervisionato, modelli basati su meccanismi di attenzione ed esperimenti di explainability, come mappe di attenzione o approcci basati su prototipi.

3.3.2 Scopo

Il dataset è stato scelto per valutare le prestazioni dei modelli nel segmentare nuove patch a seguito di un fine-tuning dei parametri, fatto solo su un piccolo sottoinsieme del dataset. La necessità di questa valutazione è data dal fatto che, come in questo dataset, anche nei casi reali non è presente un ground truth; le informazioni ricavabili da esso, però, possono essere molto utili nella digital patology per integrare le immagini con features numeriche e ottenere risultati migliori con strumenti di deep learning come quelli di classificazione. Dato che da ogni WSI si possono estrarre migliaia di patch valide, la segmentazione a mano, e di conseguenza un training completo, non è una strada percorribile.
Come conseguenza di queste osservazioni si è pensato di ottimizzare solo alcuni parametri

dei modelli pre-addestrati su un campione di immagini rappresentative del dataset, in modo da migliorare complessivamente le prestazioni dei modelli sul resto del dataset.

3.3.3 Selezione immagini

Come esposto nell'introduzione al dataset, le immagini sono raggruppate in tre macro categorie che sono i diversi stadi tumorali: epitelioide, bifasico e sarcomatoide. Dai tessuti è possibile riconoscere lo stadio in base alla tipologia di cellule presenti, infatti lo stadio epitelioide ha nuclei di forma arrotondata mentre il sarcomatoide ha nuclei più allungati e meno regolari. Lo stadio intermedio, il bifasico, non ha caratteristiche proprie ma contiene in pari percentuale cellule degli stadi epitelioide e sarcomatoide. Per questa ragione il patologo coinvolto nel progetto di tesi ha consigliato 8 patch significative divise come segue

1. **Epitelioide:**
   M-59:
   - Patch_10080_11200.png
   - Patch_10080_11648.png

   M-68:
   - Patch_10304_9408.png
   - Patch_11648_10080.png

2. **Sarcomatoide:**
   M-101:
   - Patch_10528_17920.png
   - Patch_13440_10080.png

   M-114:
   - Patch_10080_21504.png
   - Patch_10080_21728.png

Per queste immagini è stata calcolato a mano il ground truth con il programma QuPath che permette di avere una prima segmentazione automatica sfruttando un'estensione con StarDist. Successivamente possono essere aggiunte e cancellate le segmentazioni delle singole cellule. Nel seguito va quindi tenuto in considerazione che il bordo della maggioranza di cellule è stato creato da StarDist, introducendo un bias nella valutazione del modello, e che il risultato potrebbe contenere qualche imprecisione. Come insieme test su cui validare i parametri ottenuti sono state scelte altre 8 immagini così suddivise

1. **Pazienti noti:**
   Epitelioide M-59:
   - Patch_10080_11424.png

   Epitelioide M-68:

Patch_10752_3360.png

Sarcomatoide M-101:

Patch_13440_10304.png

Sarcomatoide M-114:

Patch_10080_21280.png

2. **Pazienti nuovi:**

Bifasico M-30:

Patch_10080_10080.png

Sarcomatoide M-86:

Patch_10080_10080.png

Epitelioide M-85:

Patch_10080_11200.png

Bifasico M-73:

Patch_10080_17472.png

Analogamente a quanto fatto per le altre immagini anche per queste è stato ricostruito il ground truth.

3.3.4 Analisi Eseguite

Per quanto riguarda l'ottimizzazione dei parametri la scelta è ricaduta sulla libreria Optuna di Python [24]. Optuna è una libreria open-source per l'ottimizzazione automatica degli iperparametri nei modelli di machine learning. Utilizza tecniche efficienti come il pruning dinamico e l'ottimizzazione bayesiana per esplorare lo spazio dei parametri in modo intelligente e accelerare il training.
Data la velocità di Optuna nella ricerca dei parametri ottimali all'interno di intervalli continui, sia per StarDist che per Cellpose è stato considerato un parametro in più rispetto a quanto fatto in precedenza.
Per Cellpose, oltre a diametro e flow threshold, è stato aggiunta la *cellprobability threshold* che è una soglia applicata alla mappa di probabilità di cellula (cell probability map) prodotta dal modello. Non è una vera e propria probabilità perchè può essere anche negativa, infatti i pixel di sfondo tendono ad avere cellprobability negativa al contrario di quelli appartenenti a cellule.
Per quanto riguarda `min_size` invece non è stato creato un intervallo di valori ma è stato fissato a 2 in modo da considerare anche le cellule più piccole. Provare ad aumentarlo non ha apportato miglioramenti ma ha solo escluso alcune cellule utili. Altri parametri fissati sono `normalize=True` e `augment=True`, rispettivamente, per normalizzare i pixel dell'immagine e ruotarla durante l'inferenza, tutto ciò rende il modello più stabile anche

se più lento. Gli intervalli suggeriti ad Optuna sono stati scelti tra i più comuni per questo tipo di immagini

- diametro: $(3, 15)$
- flow threshold: $(0.3, 1)$
- cellprobability threshold: $(-1.5, 0.5)$

Per StarDist invece è stato aggiunto il parametro *scale factor* che, come il diametro per Cellpose, è legato alla dimensione dei pixel dell'immagine. Scale factor determina, infatti, un resize dell'immagine che è utile dato che StarDist lavora meglio se le cellule hanno la stessa dimensione di quelle su cui è stato addestrato. Per questa ragione nella valutazione della sezione precedente era stato fatto un resize dell'immagine 224x224.
Gli intervalli suggeriti ad Optuna per StarDist sono

- Scale factor: $(5, 1.5)$
- Probability threshold: $(0.1, 0.9)$
- NMS threshold: $(0.1, 0.9)$

Una volta scelti i parametri la loro ottimizzazione, per entrambi i metodi, sulle immagini di training, è stata articolata in quattro fasi:

1. Parametri ottimi per ciascuna immagine (8)
2. Parametri ottimi per ciascun paziente (4)
3. Parametri ottimi per ogni stadio (2)
4. Parametri ottimi globali (1)

Dato che i ground truth sono solo binari e non distinguono le singole label, la metrica scelta è il Dice.
L'obiettivo dell'analisi che segue è sia confrontare tra loro i modelli che le strategie per cercare di capire se qualche combinazione risulti migliore delle altre. Tutto questo sempre tenendo conto che le immagini a disposizione sono poche anche se questa condizione è uno specchio di ciò che spesso si ha davvero a disposizione in casi reali.

3.3.5 Risultati

Train

Nelle tabelle che seguono sono riportati i parametri migliori ottenuti per Stardist e Cellpose in ognuno dei 4 casi sopra descritti.

Tabella 3.1: Migliori parametri trovati per StarDist nelle diverse strategie di classificazione.

Immagini singole

Immagine	Scale factor	Prob. thresh	NMS thresh
Patch_10080_21728_s114	1.498	0.166	0.794
Patch_10080_21504_s114	1.403	0.114	0.559
Patch_10080_11200_e59	1.498	0.174	0.851
Patch_10528_17920_s101	1.443	0.160	0.320
Patch_13440_10080_s101	1.409	0.104	0.536
Patch_11648_10080_e68	1.411	0.133	0.794
Patch_10304_9408_e68	1.423	0.177	0.707
Patch_10080_11648_e59	1.423	0.160	0.541

Paziente

Paziente	Scale factor	Prob. thresh	NMS thresh
M-101	1.452	0.100	0.577
M-68	1.482	0.184	0.592
M-114	1.192	0.102	0.686
M-59	1.498	0.104	0.519

Stadio

Stadio	Scale factor	Prob. thresh	NMS thresh
Sarcomatoid	1.462	0.105	0.495
Epithelioid	1.496	0.115	0.649

Globale

Scale factor	Prob. thresh	NMS thresh
1.465	0.126	0.526

I nomi delle immagini sono stati leggermente modificati in modo da avere come ultimi caratteri un richiamo al codice del paziente e alla lettera iniziale dello stadio (per esempio _s101 si riferisce al paziente M-101 di stadio sarcomatoide). Si può notare come, soprattutto con StarDist, non ci sia una grande variazione nei parametri usando i diversi raggruppamenti delle immagini.

In Cellpose ciò che varia sono soprattutto il diametro e la Cellprobability threshold; si

Tabella 3.2: Migliori parametri trovati per Cellpose nelle diverse strategie di classificazione.

Immagini singole

Immagine	Diameter	Flow threshold	CellProb threshold
Patch_10080_21728_s114	14.837	1.000	-1.450
Patch_10080_21504_s114	14.405	0.878	-1.466
Patch_10080_11200_e59	14.734	0.681	-0.796
Patch_10528_17920_s101	13.170	0.801	-1.183
Patch_13440_10080_s101	14.103	0.757	-1.188
Patch_11648_10080_e68	11.128	0.852	-0.680
Patch_10304_9408_e68	13.657	0.972	-0.479
Patch_10080_11648_e59	13.292	0.994	-0.910

Paziente

Paziente	Diameter	Flow threshold	CellProb threshold
M-101	13.229	0.715	-1.194
M-68	14.770	0.734	-0.650
M-114	14.379	0.767	-1.375
M-59	12.416	0.913	-0.776

Stadio

Stadio	Diameter	Flow threshold	CellProb threshold
Sarcomatoid	14.269	0.875	-1.499
Epithelioid	13.302	0.649	-0.798

Globale

Diameter	Flow threshold	CellProb threshold
12.905	0.936	-1.218

può notare che questa variazione non è casuale nelle immagini ma dipende dallo stadio, infatti, stadi di tumore più avanzati (qui il sarcomatoide) sono caratterizzati da nuclei di forme più irregolari e allungate che hanno quindi diametri maggiori. In questi casi parte del nucleo fuoriesce quindi è anche più facile confonderlo con il background, da qui la Cellprobability minore.

In figura 3.17 sono riportate in ordine le immagini con più basso e più alto diametro: Patch_11648_10080_e68 e Patch_10080_21728_s114 che sono rispettivamente in uno stadio epitelioide e sarcomatoide e evidenziano bene questa differenza.

Si procede ora con la valutazione del Dice ottenuto sulle patch al variare del set di parametri utilizzato; di seguito le tabelle per StarDist e Cellpose:

Dalle tabelle si può notare come Cellpose sia tendenzialmente più costante nei valori

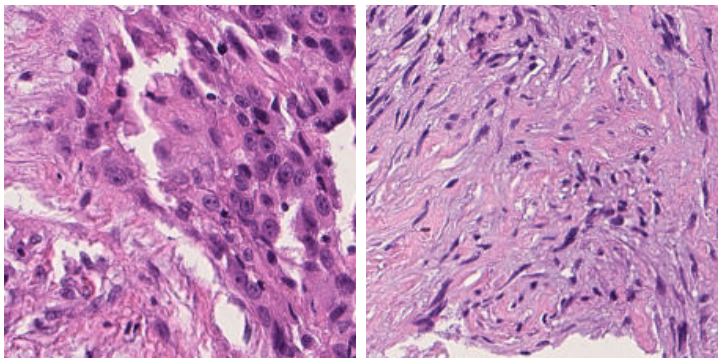


Figura 3.17: Confronto forma cellule in stadi diversi

Tabella 3.3: Risultati Dice per il modello StarDist

Immagine	**Immagini singole**	**Paziente**	**Stadio**	**Globale**
Patch_10080_11200_e59	0.800	0.803	**0.808**	0.786
Patch_10080_11648_e59	0.805	**0.829**	0.824	0.820
Patch_10080_21504_s114	0.585	0.417	**0.595**	0.590
Patch_10080_21728_s114	0.547	0.371	**0.560**	0.554
Patch_10304_9408_e68	**0.846**	0.841	0.832	0.836
Patch_10528_17920_s101	0.696	**0.700**	0.696	0.699
Patch_11648_10080_e68	0.825	**0.830**	0.828	0.817
Patch_13440_10080_s101	**0.700**	0.656	0.698	0.699

di Dice mentre StarDist abbia picchi di performance nelle immagini epitelioidi che sono tendenzialmente più semplici mentre cali poi nelle sarcomatoidi più complesse.
Non sempre il valore migliore per l'immagine (quello in grassetto) è quello ottenuto con l'ottimizzazione nell'immagine singola, quest'ultimo però non si discosta mai più di qualche millesimo. Questa leggera variazione può essere dovuta al fatto che Optuna ha una componente casuale, anche se pesata, che fa sì che non visiti mai gli stessi punti e questo può leggermente influenzare il Dice finale. Aumentando il numero di immagini usate in uno studio, inoltre, si ottengono tendenzialmente soluzioni più stabili.
Tranne in un caso, quasi mai il valore migliore è quello globale, ciò è coerente con quanto detto sulle differenze tra stadi diversi. Da notare però che si parla sempre di differenze minime rispetto agli altri parametri.
Per ulteriori valutazioni sono riportate le heatmap in figura 3.18 e 3.19 che confrontano tutti i valori ottenuti con StarDist con quelli di Cellpose, con l'aggiunta dei risultati ottenuti con i parametri di default dei due modelli.
Come già considerato i risultati peggiori si ottengono con StarDist di default che in

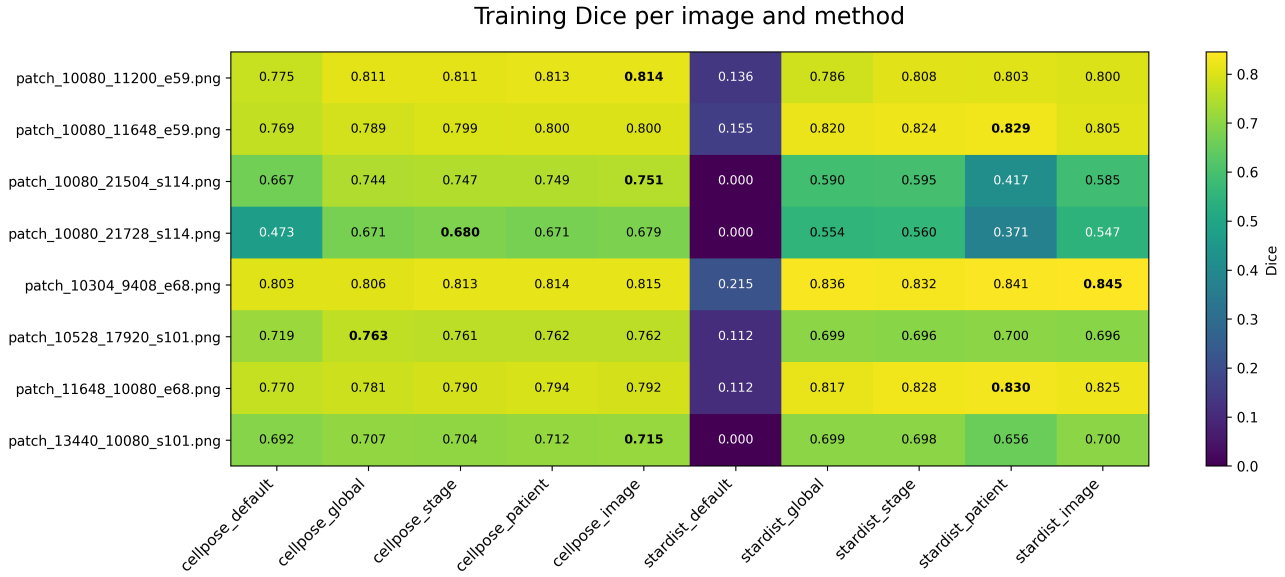


Figura 3.18: Heatmap di confronto sui Dice ottenuti durante il training

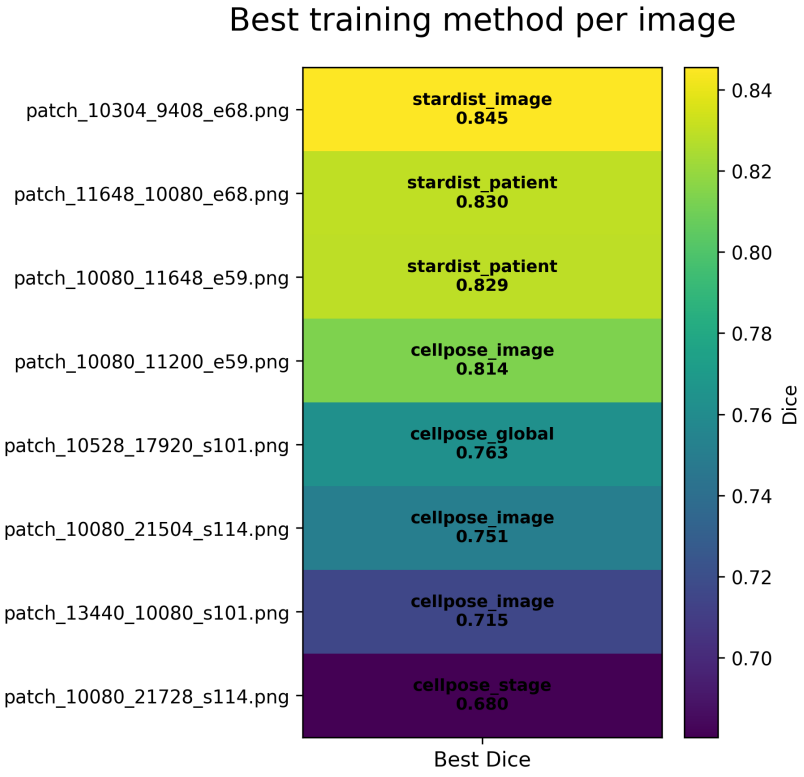


Figura 3.19: Heatmap dei Dice migliori per ogni immagine di training

Tabella 3.4: Risultati Dice per il modello Cellpose

Immagine	Immagine singola	Paziente	Stadio	Globale
Patch_10080_11200_e59	**0.814**	0.813	0.811	0.811
Patch_10080_11648_e59	**0.800**	**0.800**	0.799	0.789
Patch_10080_21504_s114	**0.751**	0.749	0.747	0.744
Patch_10080_21728_s114	0.679	0.671	**0.680**	0.671
Patch_10304_9408_e68	**0.815**	0.814	0.813	0.806
Patch_10528_17920_s101	0.762	0.762	0.761	**0.763**
Patch_11648_10080_e68	0.792	**0.794**	0.790	0.781
Patch_13440_10080_s101	**0.716**	0.712	0.704	0.707

alcuni casi raggiunge addirittura 0 di Dice. Su tre figure StarDist ottiene i risultati migliori, tutte e tre sono immagini di stadio epiteliode con le cellule regolari su cui quindi StarDist è molto performante. Nelle altre il migliore è Cellpose che, nonostante non raggiunga punteggi alti come StarDist, non perde troppo anche sulle immagini più difficili. Nonostante Cellpose di default performi comunque bene, in alcuni casi come nella Patch_10080_21728_s114.png, l'ottimizzazione dei parametri fa la differenza.
In figura 3.20 e 3.21 sono riportate le segmentazioni effettuate da Stardist e Cellpose migliori nei due casi estremi: Patch_10080_21728_s114.png e Patch_10304_9408_e68.png.

Nelle immagini riportate sono presenti la patch originale, il GT originale, la maschera predetta e l'heatmap che evidenzia in verde le cellule predette bene sovrapponendo le due maschere, in arancione quelle mancanti e in magenta quelle predette sbagliate.
Si può notare che nella Patch_10080_21728_s114.png l'errore di StarDist è principalmente nel non identificare le cellule piuttosto che sbagliarne la forma.

Test

Si può ora discutere dei risultati ottenuti nella fase di test. L'insieme test è stato costruito per avere un'immagine per ogni paziente già visto in modo da valutare se ci può essere un vantaggio nell'usare parametri ad hoc per paziente piuttosto che quelli di stadio o globale. Gli altri quattro pazienti sono divisi nei tre stadi: un epitelioide, due bifasici e un sarcomatoide. Per i bifasici, chiaramente, l'unico set di parametri applicabile sarà quello globale e l'obiettivo sarà valutare quanto il non aver visto pazienti bifasici in fase di training possa incidere nella segmentazione.

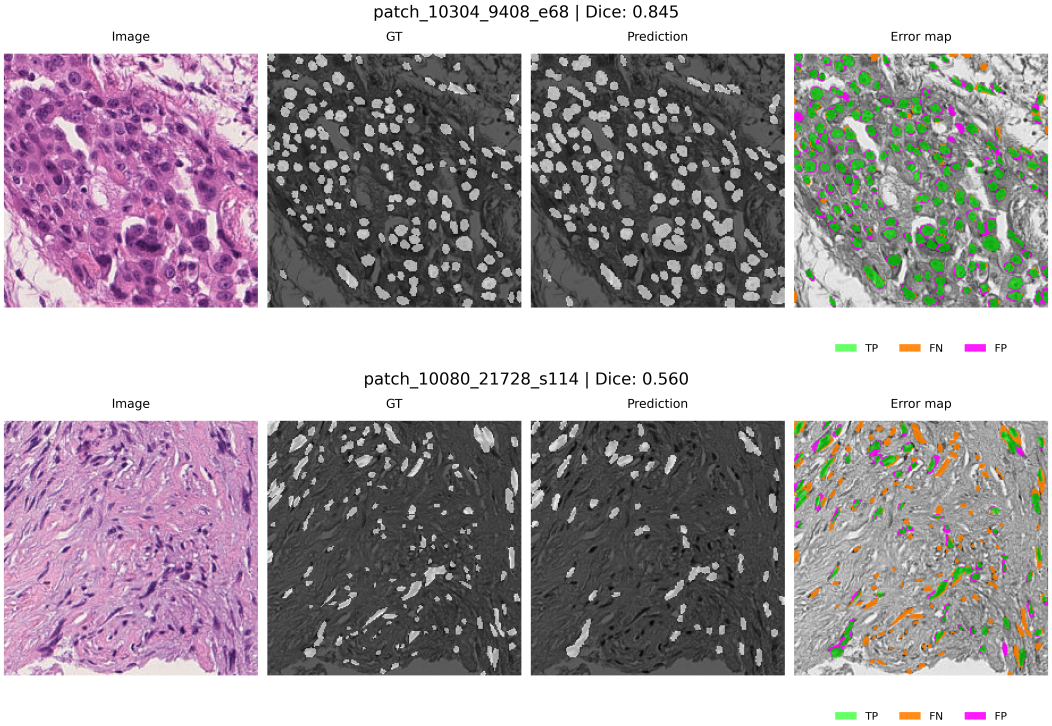


Figura 3.20: Segmentazione di StarDist per la Patch_10080_21728_s114.png con i parametri dello stadio e Patch_10304_9408_e68.png con i parametri dell'immagine singola

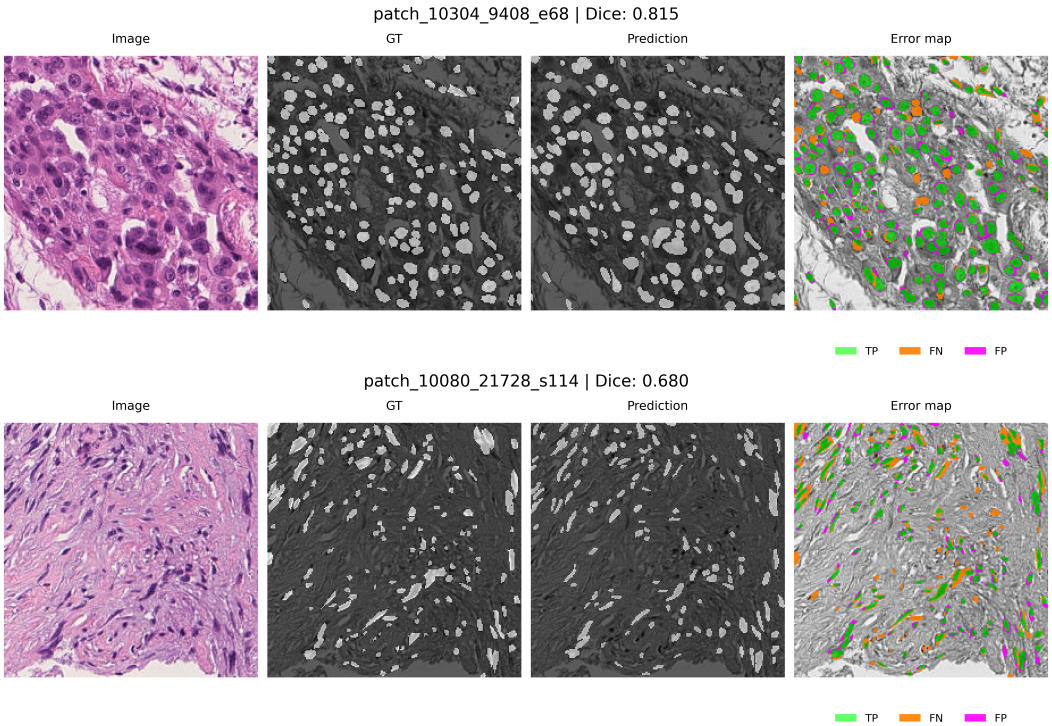


Figura 3.21: Segmentazione di Cellpose per la Patch_10080_21728_s114.png con i parametri dello stadio e Patch_10304_9408_e68.png con i parametri dell'immagine singola

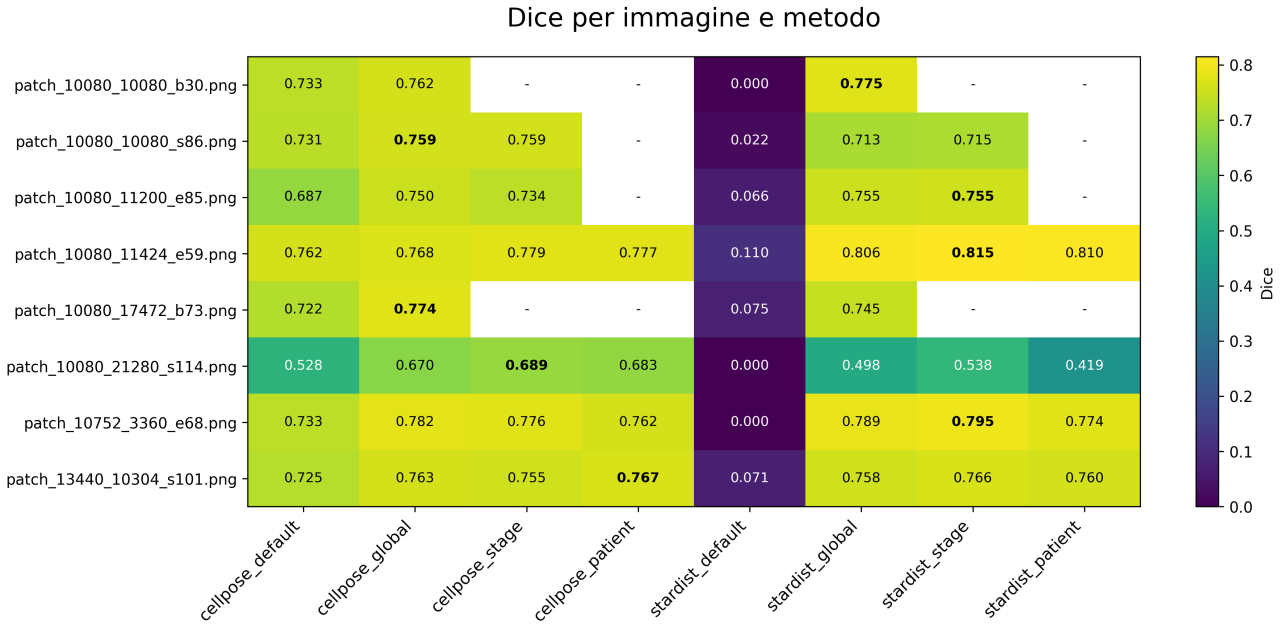


Figura 3.22: Heatmap di confronto sui Dice ottenuti durante il training

In figura 3.22, come fatto per il training sono riportati, per ogni immagine, i valori ottenuti al variare del metodo e del set di parametri utilizzato. Mentre in figura 3.23 i migliori set di parametri per immagine.

La situazione che si presenta è analoga a quanto visto nel training: Cellpose vince sui pazienti di stadio sarcomatoide e su un bifasico mentre StarDist sugli altri, ottenendo i migliori score di Dice.

Da notare che i risultati dei parametri globali su tutte le immagini sono buoni, soprattutto nei bifasici che non sono mai stati visti nel training. Questo dato va a conferma del fatto che l'importante è avere una buona panoramica delle tipologie di cellule da segmentare e che per questo basta usare i due casi estremi senza bisogno dello stadio intermedio o di rappresentare tutti i pazienti. Infatti, si può notare che solo in un caso i parametri ottimali sono quelli per paziente, più spesso, dove si possono utilizzare tutti i set, ciò che vince sono i parametri di stadio.

Altra osservazione è data dal fatto che i risultati più generalizzabili sono quelli in cui si avevano più immagini a disposizione per l'ottimizzazione dei parametri, quindi, avendo poche immagini è meglio privilegiare meno categorie ma più ampie piuttosto che molto specifiche.

Anche in questo caso sono stati aggiunti i risultati con i modelli standard, in particolare,

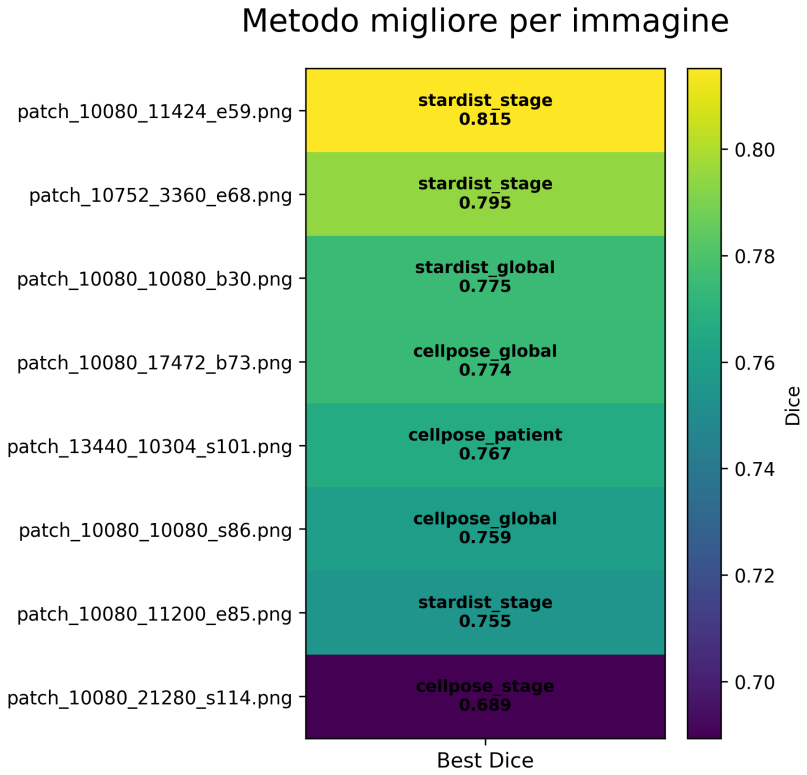


Figura 3.23: Heatmap dei Dice migliori per ogni immagine di training

in figura 3.24 sono state calcolate le differenze tra i modelli personalizzati e quelli standard per avere una panoramica più precisa dei miglioramenti ottenuti.
Come già osservato, i guadagni maggiori si hanno su StarDist mentre Cellpose risente di questa variazione principalmente sulle immagini più diffcili come quella scelta dal paziente M-114.

Conclusioni

In conclusione, quindi, si può affermare che entrambi i metodi con un fine-tuning dei parametri raggiungono buone prestazioni. La scelta migliore per questo dataset sarebbe quella di differenziare in base allo stadio del tumore, aggiungendo qualche immagine per il bifasico, oppure usare parametri globali a partire dalle immagini già disponibili. Chiaramente, nel caso in cui la segmentazione dovesse poi essere usata per addestrare un modello di classificazione, come verrà fatto nel capitolo successivo, per non introdurre bias sul test ci si dovrebbe affidare a parametri globali.
Come modello, il più stabile risulta essere Cellpose che permette di ottenere buoni

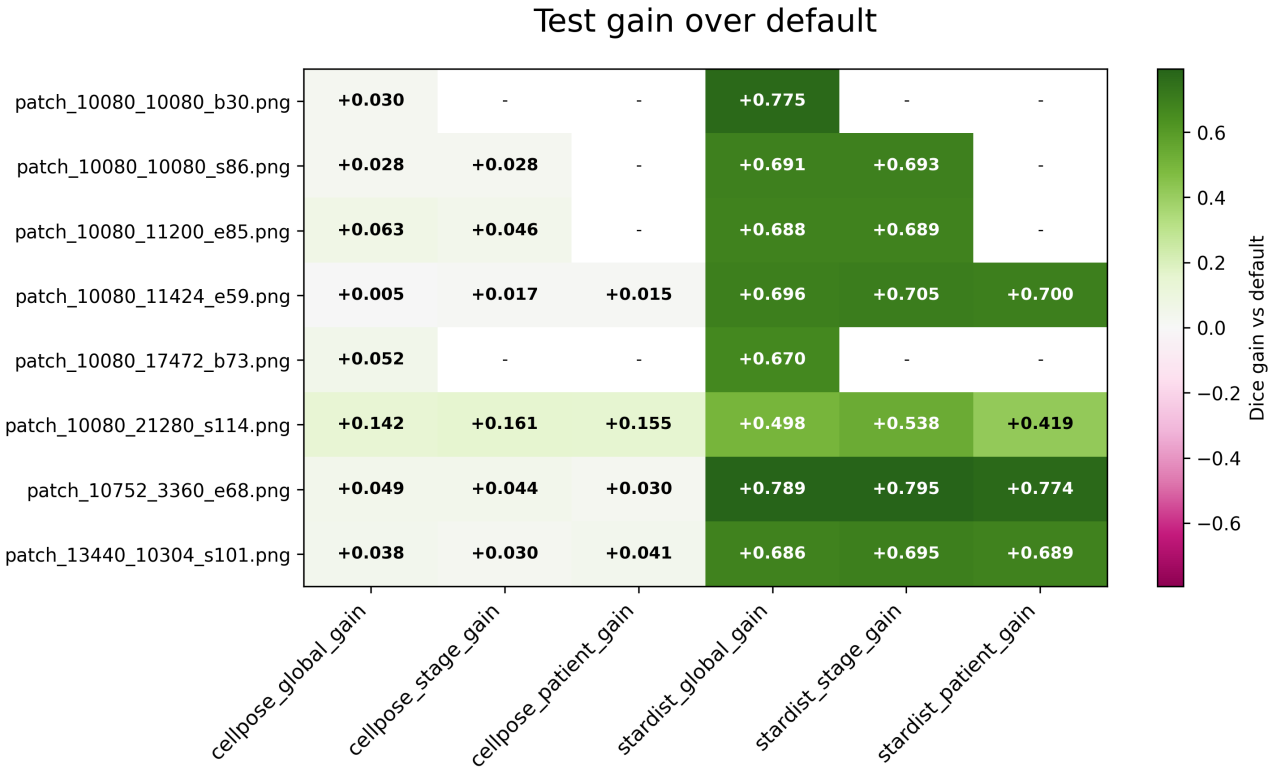


Figura 3.24: Guadagno in Dice tra i valori ottenuti personalizzando i parametri e quelli di default

risultati anche nei casi più difficili. Un altro aspetto da considerare, non ancora nominato, è dato dai tempi di calcolo. Infatti Cellpose sfrutta la GPU e richiede tempi più lunghi di StarDist, per questo, senza GPU o con un dataset molto grande, StarDist diventa la scelta migliore.

Capitolo 4

Classificazione di vetrini istologici

In quest'ultimo capitolo viene descritto il progetto realizzato nell'ambito del tirocinio presso il reparto di Fisica Medica dell'AUSL di Reggio Emilia.
Il lavoro di tesi si inserisce nel progetto sviluppato dal reparto di Anatomia Patologica in collaborazione con Fisica Medica, dal titolo: *Federated Learning as a step forward in digital pathology: a multi-centric pipeline and a quality assurance (QA) platform to support diagnosis of prostate cancer, in WSI of core needle biopsies, to evaluate artificial intelligence systems(CAD and QC tools).* [RF-2021-12374001]
I dati disponibili sono un insieme di slides di vetrini, con colorazione ematossilina & eosina (H&E), appartenenti a 40 pazienti consecutivi. Per ogni paziente sono presenti 12 campioni bioptici della prostata, corredati di diagnosi tumorale (indice da 0 a 5) e gleason score, in caso di positività al tumore. Quest'ultimo è un sistema utilizzato per valutare l'aggressività del tumore alla prostata (adenocarcinoma); è costituito da uno score primario e uno secondario che possono valere 3,4 o 5 dove 5 rappresenta la massima aggressività.
L'idea è stata quella di addestrare un modello di classificazione binaria, adattando il metodo PINS [25] con i campioni disponibili. Il metodo PINS è stato sviluppato per sfruttare, oltre che le immagini istologiche, i vettori di features morfologiche estraibili dalle maschere che si ottengono tramite segmentazione. Quanto detto fin'ora sulla segmentazione risulta pertanto essere alla base di questo modello.

4.1 La costruzione del dataset

Per l'addestramento del modello sono stati selezionati 500 WSI. Ogni campione bioptico refertato corrisponde a 2 o 3 WSI (slides) pertanto è stata fatta una prima suddivisione in modo da abbinare ad ogni set di WSI il suo referto corrispondente, per uno totale di 249 diagnosi. La ragione di usare i referti e non i pazienti come suddivisione dei WSI è data dal fatto che referti appartenenti allo stesso paziente possono avere diagnosi diverse perchè il tumore, se presente, non deve essere necessariamente rilevato in tutti i campioni bioptici refertati. Questa mancanza di correlazione garantisce di poter trattare i referti come individui distinti aumentando così notevolmente il campione disponibile per l'addestramento del modello.
La seconda operazione fatta è stata quella, tramite python, di caricare e dividere il WSI in patch formato png di dimensione 512x512. Per riuscire a estrarre le patch dal livello 0 del WSI che è quello a più alta risoluzione, le slides sono state caricate al livello desiderato selezionando ricorsivamente l'intervallo di coordinate che corrispondono alla patch cercata, senza dover quindi caricare l'intera immagine. Tra le patch estratte sono state eliminate tutte quelle contenenti più del 70% di colore bianco, che corrisponde al background del WSI. In totale sono state selezionate 704348 patch divise nelle 249 cartelle corrispondenti ai referti.
La diagnosi tumorale di ogni cartella, che prende valori da 0 a 5, è così articolata:

- 0: assenza di tumore
- 1: ASAP/ATYP che richiede approfondimento
- 2: PIN di alto grado
- 3: AK intraduttale
- 4: positivo per adenocarcinoma acinare
- 5: positvo per adenocarcinoma duttale

Ai fini di ottenere una classificazione binaria, distinguendo tra presenza o assenza di una forma tumorale, sono state considerate le diagnosi 0 e 2 come negative al tumore e 1,3,4,5 come positive all'adenocarcinoma. Il dataset così creato risulta essere bilanciato (121 negative e 128 positive).

4.2 Segmentazione e estrazione caratteristiche morfologiche

Modello e ottimizzazione parametri

Data l'enorme mole di dati la scelta del metodo è ricaduta su StarDist per ottenere una maggiore velocità di processazione. Come nel capitolo precedente, in assenza di un GT delle patch, si è scelto di procedere segmentando manualmente un campione di 6 patch, rappresentative del dataset, da utilizzare per un fine-tuning.
La scelta delle patch è stata fatta selezionando referti corrispondenti a diverse combinazioni possibili di diagnosi e score gleason. Individuati i referti adeguati, tramite le annotazioni sui WSI delle zone designate come tumorali, si è potuto risalire alle patch corrispondenti alle suddette zone per mezzo delle coordinate di estrazione delle patch. L'insieme delle patch scelte è quindi così composto:

- Patch1_C1_positiva_3:
  Diagnosi: Positiva | Gleason: 3

- Patch2_J2_positiva_4:
  Diagnosi: Positiva | Gleason: 4

- Patch3_G2_positiva_5:
  Diagnosi: Positiva | Gleason: 5

- Patch4_L1_negativa:
  Diagnosi: Negativa

- Patch5_L2_negativa:
  Diagnosi: Negativa

- Patch6_L2_negativa:
  Diagnosi: Negativa

In figura 4.2 sono riportate le immagini appena elencate. Tramite l'ottimizzazione con Optuna i parametri globali ottenuti sono riportati in tabella 4.1
Con questi parametri i risultati sull'insieme di training sono riportati in tabella 4.2.

Probability_threshold	**NMS_threshold**	**Scale**
0.255	0.767	1.110

Tabella 4.1: Parametri ottimi trovati

Immagine	**Diagnosi**	**Dice**
Patch1_C1_positiva_3	Positiva	0.883
Patch2_J2_positiva_4	Positiva	0.876
Patch3_G2_positiva_5	Positiva	0.896
Patch4_L1_negativa	Negativa	0.913
Patch5_L2_negativa	Negativa	0.890
Patch6_L2_negativa	Negativa	0.910
Media		**0.895**

Tabella 4.2: Dice per ogni immagine usando i parametri migliori trovati e media totale

Si nota che con una migliore qualità delle immagini e una maggiore dimensione della patch, che richiede a StarDist solo un minimo resize (scale factor vale 1.11), l'accuratezza nella segmentazione aumenta. In figura 4.2 sono riportate le sovrapposizioni delle maschere ottenute alle patch corrispondenti.

In figura 4.1, invece, sono mostrate le heatmap, analoghe a quelle create per il dataset CAMEL, per il migliore e il peggior Dice ottenuti, pur essendo entrambi ottimi risultati. Osservando la tabella 4.2, in cui sono presenti anche le diagnosi dei referti associati alle patch, si può affermare che non c'è una differenza netta nelle performance tra patch con e senza tumore, anche se il miglior Dice corrisponde ad una patch negativa e il peggiore ad una positiva. Si ricorda che, in caso di diagnosi positiva, la patch è stata scelta in modo tale da contenere effettivamente cellule tumorali.

4.2.1 Segmentazione e estrazione features morfologiche

Con i parametri ottimizzati è possibile, quindi, procedere alla segmentazione integrale del dataset. Dalle maschere così create sono state estratte, per ciascun oggetto segmentato, una serie di caratteristiche morfologiche e di intensità basate sulle proprietà regionali, usando *regionprops* di Python.
Tali feature consentono di descrivere forma, dimensione e distribuzione dell'intensità delle singole cellule.

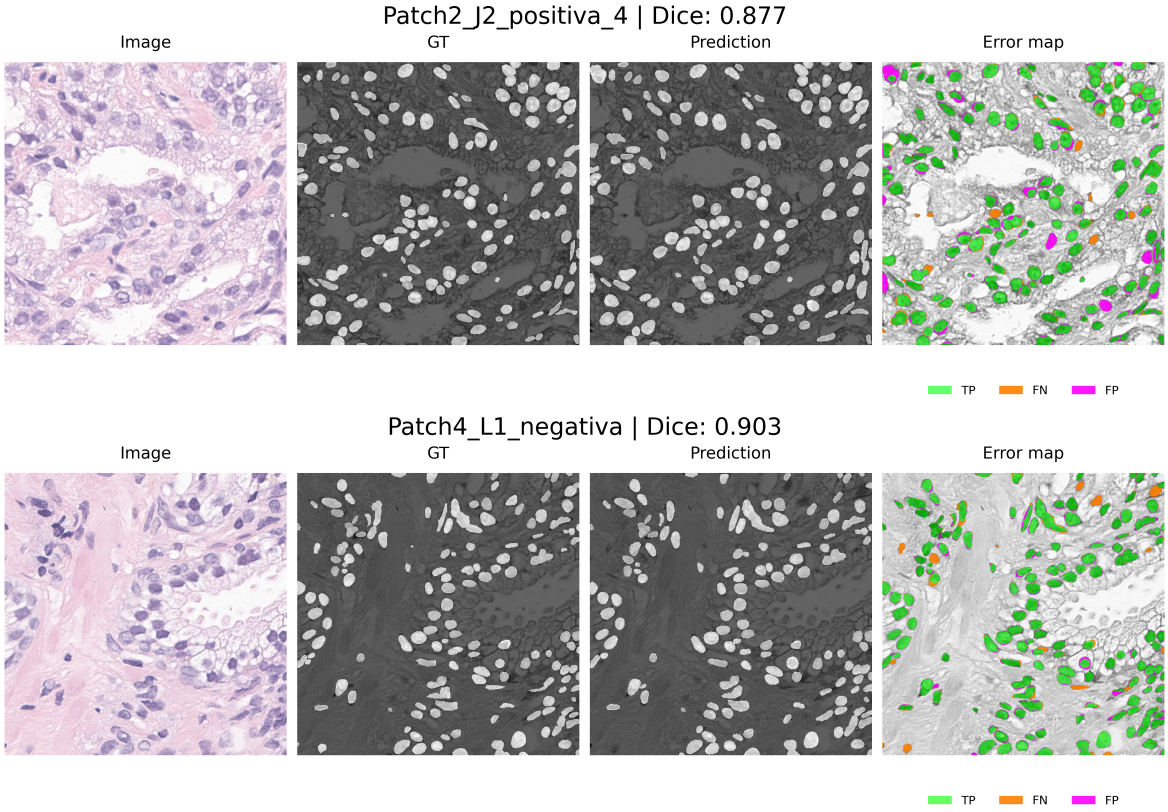


Figura 4.1: Heatmap delle patch segmentate meglio e peggio

Le principali caratteristiche estratte sono:

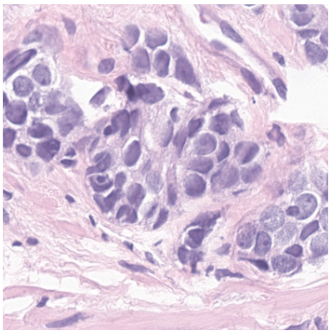
- **Area** (A): numero di pixel appartenenti alla cellula.
- **Perimetro** (P): lunghezza del contorno della cellula.
- **Eccentricità**: misura della deviazione dalla forma circolare, definita come l'eccentricità dell'ellisse equivalente (valori in $[0, 1]$).
- **Area convessa** (A_c): area del minimo inviluppo convesso della cellula.
- **Area bounding box** (A_b): area del rettangolo minimo che contiene la cellula.
- **Solidity**: rapporto tra area e area convessa che indica quanto la forma sia priva di concavità:

$$\text{solidity} = \frac{A}{A_c}$$

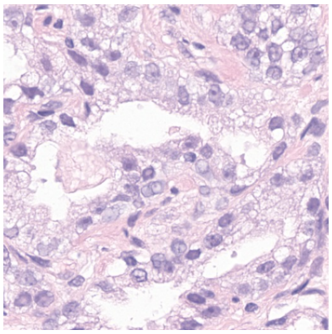
- **Extent**: rapporto tra area e area del bounding box:

$$\text{extent} = \frac{A}{A_b}$$

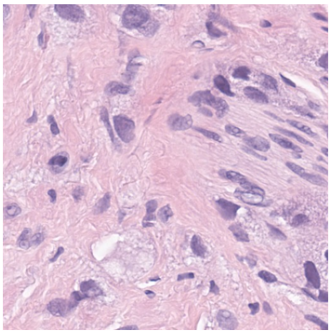
- **Lunghezza asse minore** (b): lunghezza dell'asse minore dell'ellisse equivalente.



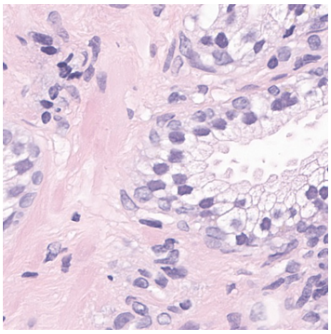
Patch1_C1_positiva_3



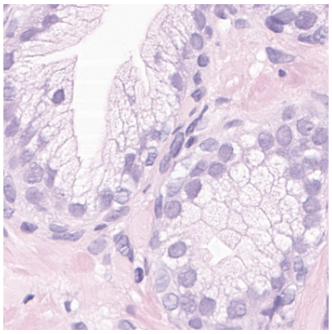
Patch2_J2_positiva_4



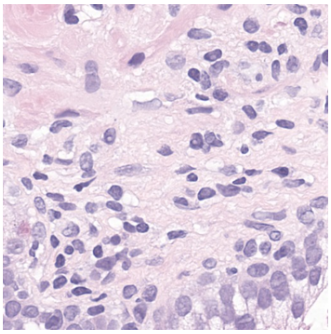
Patch3_G2_positiva_5



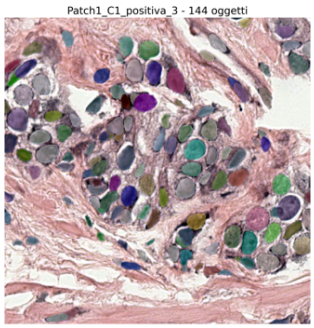
Patch4_L1_negativa



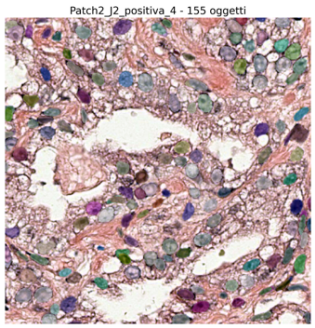
Patch5_L2_negativa



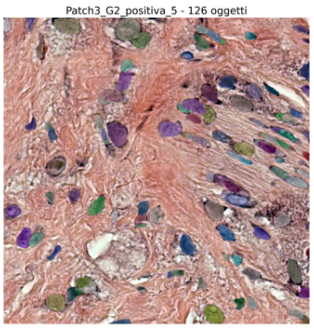
Patch6_L2_negativa



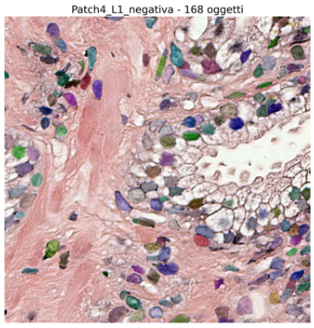
Patch1_C1_positiva_3_overlay



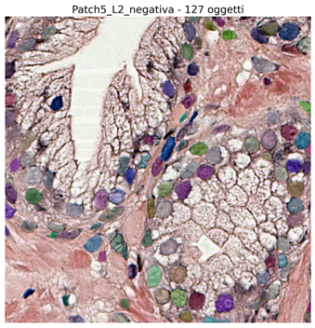
Patch2_J2_positiva_4_overlay



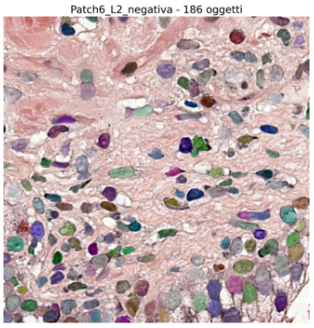
Patch3_G2_positiva_5_overlay



Patch4_L1_negativa_overlay



Patch5_L2_negativa_overlay



Patch6_L2_negativa_overlay

Figura 4.2: Immagini scelte per il fine-tuning e overlay con le maschere ottenute.

- **Lunghezza asse maggiore** (a): lunghezza dell'asse maggiore dell'ellisse equivalente.
- **Aspect Ratio**: rapporto tra asse maggiore e minore:

$$\text{aspect ratio} = \frac{a}{b}$$

- **Circularity**: misura di quanto la forma sia simile a un cerchio:

$$\text{circularity} = \frac{4\pi A}{P^2}$$

- **Roundness**: misura alternativa della rotondità basata sull'asse maggiore:

$$\text{roundness} = \frac{4A}{\pi a^2}$$

- **Compactness**: inverso della circularity:

$$\text{compactness} = \frac{P^2}{4\pi A}$$

- **Fill Ratio**: rapporto tra area e area del bounding box:

$$\text{fill ratio} = \frac{A}{A_b}$$

- **Intensità**: misura calcolata sui valori dei pixel appartenenti alla cellula. A livello cellulare vengono quindi usate:
  - **Intensità media**: media dei valori di intensità.
  - **Intensità minima**: valore minimo di intensità.
  - **Intensità massima**: valore massimo di intensità.
  - **Intensità Std**: deviazione standard dell'intensità.

Le feature estratte a livello di singola cellula sono state successivamente aggregate a livello di patch per ottenere una rappresentazione compatta della regione analizzata: per ogni feature vengono calcolate media, mediana e deviazione standard. Inoltre, per ciascuna patch viene registrato il numero totale di cellule segmentate.
Questa procedura consente di trasformare un insieme variabile di cellule per patch in

un vettore di feature di dimensione fissa, utilizzabile per successive analisi statistiche o modelli di apprendimento automatico.
Analizzando il csv con i vettori delle feature tramite Pandas è stato riscontrato che circa l'1% dei vettori conteneva valori di tipo NaN. I vettori tra questi con NaN in corrispondenza solo delle deviazioni standard delle feature cellulari (3282) sono stati conservati imputando 0 al posto del valore mancante. Questo perchè, in presenza di una sola cellula rilevata, la deviazione standard campionaria restituisce errore avendo 0 al denominatore. Negli altri casi (7391) i valori NaN erano diffusi in tutti i campi feature. Da un'osservazione delle patch incriminate si è notato che queste corrispondevano alle notazioni dei medici sulla slide, pertanto non erano patch bianche ma erano comunque prive di cellule. La decisione è stata quindi quella di eliminare tutte le righe corrispondenti a queste patch.

4.3 Classificazione

4.3.1 Introduzione al modello PINS

Per affrontare il problema della classificazione a livello di referti in presenza di etichette disponibili unicamente a livello di gruppo di slide, è stato adottato un approccio di Multiple Instance Learning (MIL) basato su campionamento adattivo delle istanze, ispirato al metodo Positive Instance Sampling (PINS) proposto in [25].
Nel contesto MIL, ogni insieme di slide, corrispondente ad un referto, è rappresentata come un insieme (chiamato bag) di istanze, corrispondenti alle patch che la compongono, mentre l'etichetta è disponibile solo a livello globale, cioè a livello di bag. Il modello apprende quindi una funzione che, a partire da rappresentazioni a livello di patch produce una predizione coerente a livello di bag.
Una selezione di feature morfologiche cellulari, viene aggregata ad ogni patch e integrata nel modello come informazione complementare alla patch, consentendo di incorporare conoscenza a priori sulla rilevanza delle caratteristiche cellulari nel tessuto.
A differenza degli approcci MIL tradizionali basati su selezione delle sole istanze più rilevanti (ad esempio max-pooling o top-N), il metodo PINS introduce una strategia di campionamento probabilistico che consente di sfruttare un insieme più ampio di istanze

informative durante l'addestramento. In particolare, ad ogni epoca viene definita, per ciascun bag, una distribuzione di probabilità sulle patch:

$$P_B(i) = \frac{z_i^\alpha + c}{\sum_j (z_j^\alpha + c)}$$

dove z_i rappresenta lo score predetto per la patch x_i, α controlla il grado di enfasi sulle istanze più rilevanti, e c è una costante che garantisce una probabilità non nulla per tutte le istanze.

Le patch vengono quindi campionate secondo tale distribuzione, privilegiando quelle con score più elevato ma mantenendo una componente esplorativa. Questo approccio consente di evitare sia l'utilizzo esclusivo di poche patch (come nei metodi basati su max pooling), sia l'assunzione che tutte le istanze condividano l'etichetta della bag.

Durante l'addestramento, il modello viene aggiornato ad ogni epoca utilizzando le istanze campionate e, successivamente, gli score vengono ricalcolati per aggiornare le distribuzioni di probabilità. Questo processo iterativo consente al modello di focalizzarsi progressivamente sulle regioni più informative, migliorando la robustezza rispetto all'inizializzazione e riducendo il rischio di overfitting.

4.3.2 Adattamento PINS

Nel presente lavoro, l'architettura PINS è stata adottata mantenendo l'integrazione tra informazione visiva e feature morfologiche a livello di patch, con l'obiettivo di ottenere una rappresentazione più espressiva della variabilità intra-tessuto e migliorare la capacità discriminativa del modello.

Nel modello originale le patch vengono processate di volta in volta da una rete ResNet che viene allenata all'interno del modello principale. In questo caso, invece, per rendere il modello più veloce ed evitare il caricamento dati ad ogni epoca, si è pensato di utilizzare una rete già addestrata per estrarre il vettore di feature visive da ogni patch, prima di iniziare il training della rete di classificazione. La rete scelta per l'estrazione è PathDINO [26], un modello basato su architettura Vision Transformer pre-addestrato mediante apprendimento auto-supervisionato secondo il framework DINO. PathDINO è stato addestrato su immagini istopatologiche su larga scala, consentendo di apprendere

rappresentazioni robuste e semanticamente rilevanti senza l'utilizzo di annotazioni esplicite, risultando particolarmente efficace in contesti di weak supervision come il Multiple Instance Learning.
I feature vector estratti hanno lunghezza 384, a questi vengono poi aggiunte le 58 feature morfologiche aggregate per ogni patch.

4.3.3 Training

Per la valutazione è stata usata una cross validation a 4 fold, con 200 epoche ciascuno e learning rate di $2e^{-5}$. Per ogni fold sono state considerate il 90% delle bag per il training e la validation e il restante 10% per il test. All'interno dell'addestramento di ciascun fold si è usato il 15% delle bag per la validation e le restanti per il training. Complessivamente quindi il 76,5% delle bag (190 bag) sono state usate nel training, il 13,5% (34 bag) nella validation e le restanti 10% (25 bag) nel test.
Per il controllo dell'addestramento è stato utilizzato un criterio di *early stopping*, che consente di interrompere l'ottimizzazione quando le prestazioni sul set di validazione non migliorano ulteriormente, evitando fenomeni di overfitting.
Inizialmente è stata utilizzata l'AUC come metrica di riferimento. L'Area Under the ROC Curve (AUC) misura la capacità del modello di discriminare tra classi positive e negative indipendentemente dalla soglia decisionale ed è interpretabile come la probabilità che un esempio positivo riceva uno score maggiore di uno negativo:

$$\text{AUC} = \frac{1}{|P||N|} \sum_{i \in P} \sum_{j \in N} \mathbf{1}(s_i > s_j)$$

dove P e N indicano rispettivamente gli insiemi degli esempi positivi e negativi, s_i è lo score predetto dal modello, e $\mathbf{1}(\cdot)$ è la funzione indicatrice.
Tuttavia, l'utilizzo della sola AUC si è rivelato instabile nel caso di un numero ridotto di bag nel set di validazione. In particolare, è stato osservato che il modello tendeva a produrre score elevati per la maggior parte delle bag, classificandole implicitamente come positive, pur mantenendo un corretto ordinamento relativo tra esempi positivi e negativi. In tali condizioni, l'AUC può assumere valori elevati anche in presenza di prestazioni di classificazione scarse.

Per questo motivo è stata considerata anche l'Accuracy, definita come:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

dove TP, TN, FP e FN sono rispettivamente il numero dei risultati positivi e negativi correttamente classificati e dei risultati positivi e negativi mal classificati. A differenza dell'AUC, l'Accuracy dipende da una soglia decisionale e penalizza esplicitamente predizioni sbilanciate, risultando più sensibile a comportamenti degeneri del modello. È stata quindi adottata una metrica combinata per l'early stopping:

$$-0.5 \cdot \text{AUC} + \text{Accuracy}$$

in modo da bilanciare la capacità discriminativa globale con la qualità della classificazione effettiva.

Infine, per evitare un arresto prematuro dovuto a fluttuazioni iniziali delle metriche, è stata introdotta una *pazienza* di 20 epoche. Questo accorgimento è particolarmente rilevante nel contesto del Multiple Instance Learning, dove l'elevata variabilità tra epoche, dovuta al campionamento delle istanze e all'aggregazione a livello di bag, può portare a miglioramenti apparenti ma non stabili. È stato inoltre imposto un intervallo minimo di 15 epoche all'inizio dell'addestramento, al fine di garantire una fase iniziale di stabilizzazione prima della valutazione del criterio di arresto.

4.3.4 Evaluation

Durante la fase di inferenza, la predizione a livello di bag viene ottenuta aggregando gli score delle istanze. In analogia al lavoro originale, è stata considerata un'aggregazione basata sulle istanze più rilevanti; tuttavia, invece di utilizzare un numero fisso di istanze (ad esempio le top-5), è stata adottata una strategia adattiva selezionando il top $k\%$ delle patch con score più elevato.

Questa scelta è motivata dalla necessità di evitare un bias verso la classe positiva. Infatti, l'utilizzo di un numero fisso e ridotto di istanze può risultare eccessivamente sensibile alla presenza di poche patch con score elevato, portando a sovrastimare la

probabilità della classe positiva anche in bag prevalentemente negative. Tale comportamento è particolarmente critico nel contesto del Multiple Instance Learning, dove anche un numero limitato di istanze "fortemente positive" può dominare il processo di aggregazione. L'adozione di una soglia percentuale consente, invece, di adattare il numero di istanze considerate alla dimensione di ciascun bag, rendendo l'aggregazione più rappresentativa del contenuto globale.
Per la determinazione della soglia percentuale è stata condotta un'analisi empirica valutando per ogni fold diverse metriche di performance al variare del threshold nell'insieme `thresholds = [0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5]`.
In particolare, per ciascun valore di soglia, sono state calcolate

- **Accuracy** che misura la proporzione di predizioni corrette ed è già stata definita nell'ambito dell'early stopping.

- **Recall** (o sensibilità) che quantifica la capacità del modello di identificare correttamente gli esempi positivi:

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **Precision** che misura la frazione di predizioni positive corrette:

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Loss (MSE)**:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$$

I risultati per fold sono stati aggregati con media e deviazione standard per avere una valutazione globale del modello.
Dal grafico risultante in figura 4.3, si osserva come la variazione della soglia influenzi in maniera significativa il comportamento del modello. In particolare, soglie più elevate tendono a ridurre il numero di predizioni positive, aumentando la precisione ma penalizzando la recall, mentre soglie più basse producono l'effetto opposto.
Nel presente lavoro è stato selezionato un valore di soglia pari a 0.25, in quanto

corrisponde al miglior compromesso tra le metriche considerate. Tale valore massimizza l'Accuracy sul set di validazione e, allo stesso tempo, minimizza il MSE, indicando una maggiore coerenza tra predizioni e ground truth.

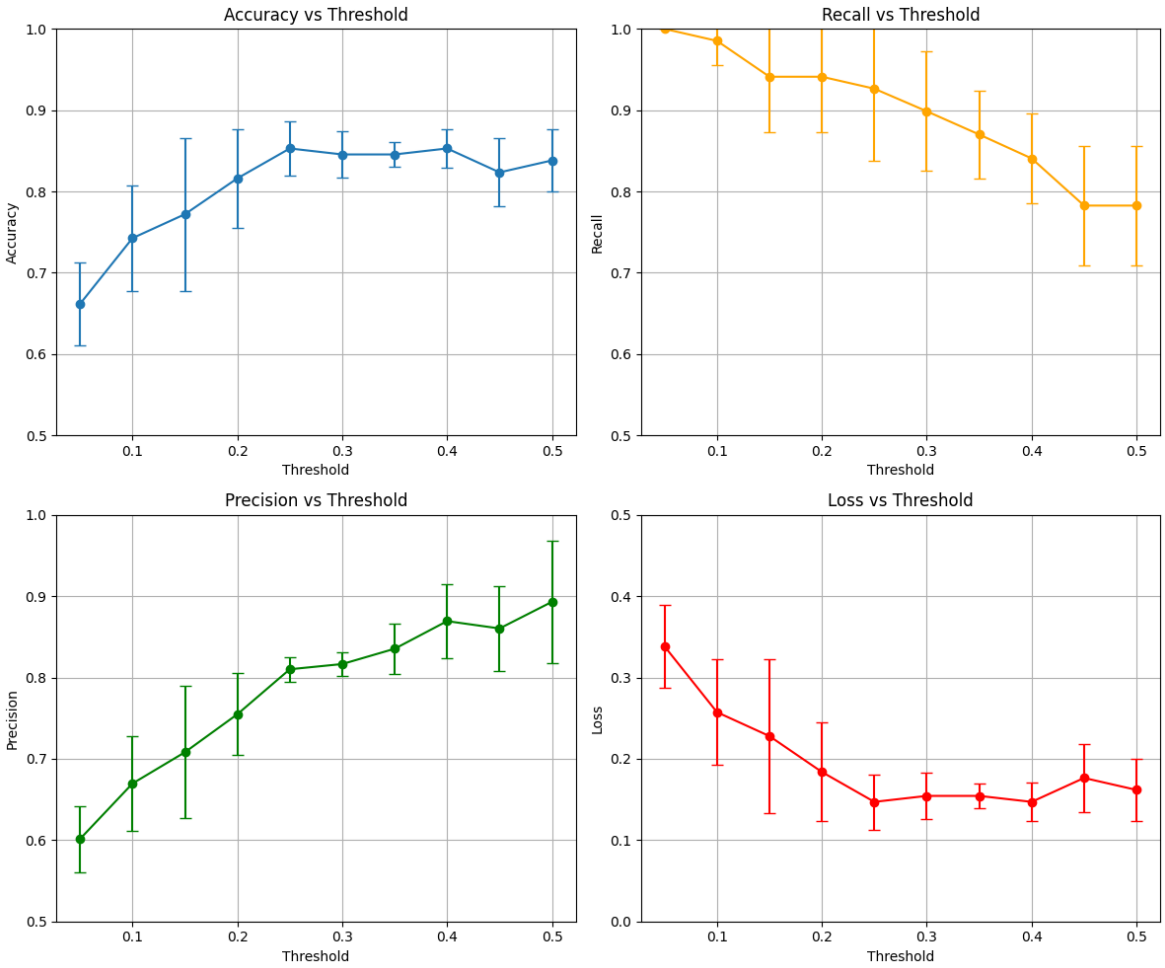


Figura 4.3: Confronto metriche aggregate al variare della threshold scelta. Oltre alle medie per ogni valore è stato riportato anche l'intervallo della deviazione standard.

In figura 4.4 sono riportati gli scatter plot dei valori di probabilità delle bag dell'insieme di validation del fold 2, divise per label. Sono presenti gli scatter plot al variare della soglia tra 15%, 20%, 25% e 30%. Si può notare come, all'aumentare della soglia, tutte le probabilità medie delle bag tendano ad abbassarsi perchè le patch aggiunte rispetto al caso precedente avranno sempre probabilità minori. Per questa ragione, al crescere della soglia aumenta la Precision che valuta quante, tra le bag classificate come positive, lo sono effettivamente, mentre diminuisce la Recall che conta il numero di bag con label 1 correttamente classificate.

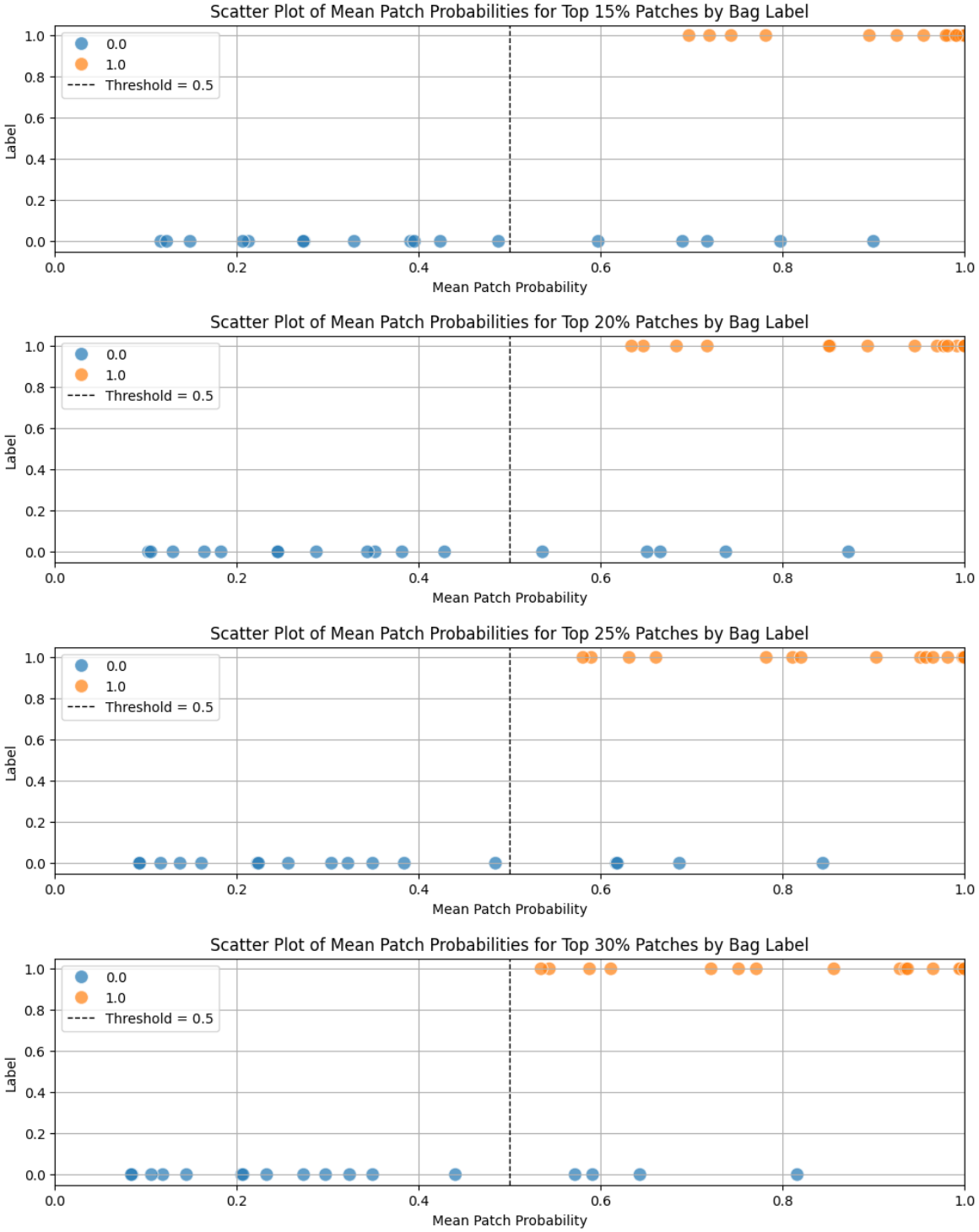


Figura 4.4: Scatter plot delle probabilità delle bag a diverse soglie percentuali di scelta del numero di patch da considerare

4.4 Risultati e considerazioni

Una volta scelta la soglia sono stati valutati i test set di ogni fold con AUC, Accuracy, F1 e Average Precision. Quest'ultima metrica misura l'area sotto la Precision-Recall curve (PR curve); valuta quindi la capacità del modello di mantenere alta precision al crescere della recall. L'AP si ottiene come

$$\mathrm{AP} = \sum_n (R_n - R_{n-1}) \cdot P_n$$

dove R_n e P_n sono rispettivamente Recall e Precision e l'indice n rappresenta i diversi punti della curva Precision-Recall, ottenuti variando la soglia decisionale o, equivalentemente, scorrendo gli esempi ordinati per score predetto.
Nella tabella che segue sono stati riportate le metriche appena elencate e i valori aggregati con media e deviazione standard.

Fold	**AUC**	**AP**	**Accuracy**	**F1-score**
Fold 1	0.859	0.884	**0.800**	0.800
Fold 2	1.000	1.000	**0.960**	0.963
Fold 3	0.878	0.930	**0.720**	0.759
Fold 4	0.949	0.958	**0.880**	0.897
Media ± Std	0.922 ± 0.056	0.943 ± 0.042	**0.840 ± 0.089**	0.855 ± 0.080

Tabella 4.3: Risultati della cross-validation sui diversi fold.

I risultati evidenziano una buona capacità discriminativa del modello, supportata non solo dagli elevati valori medi di AUC e AP, ma anche da un'Accuracy complessivamente elevata, che indica una corretta classificazione della maggior parte dei campioni. Tuttavia, si osserva una deviazione standard non trascurabile tra i fold, verosimilmente dovuta al numero limitato di bag disponibili, che rende le metriche più sensibili alla specifica suddivisione dei dati e introduce una maggiore variabilità nelle stime.
Per concludere, nel grafico 4.5, analogo a quello realizzato per i validation set, si osserva la maggiore deviazione standard appena giustificata ma anche che il valore migliore di Accuracy è ora in corrispondenza della soglia 30%. La scelta di utilizzare il 25% è stata effettuata senza considerare le prestazioni sul test set, al fine di evitare fenomeni di *data leakage* e garantire una valutazione imparziale del modello. In questo modo, la selezione degli iperparametri rimane confinata ai dati di validazione, preservando il test

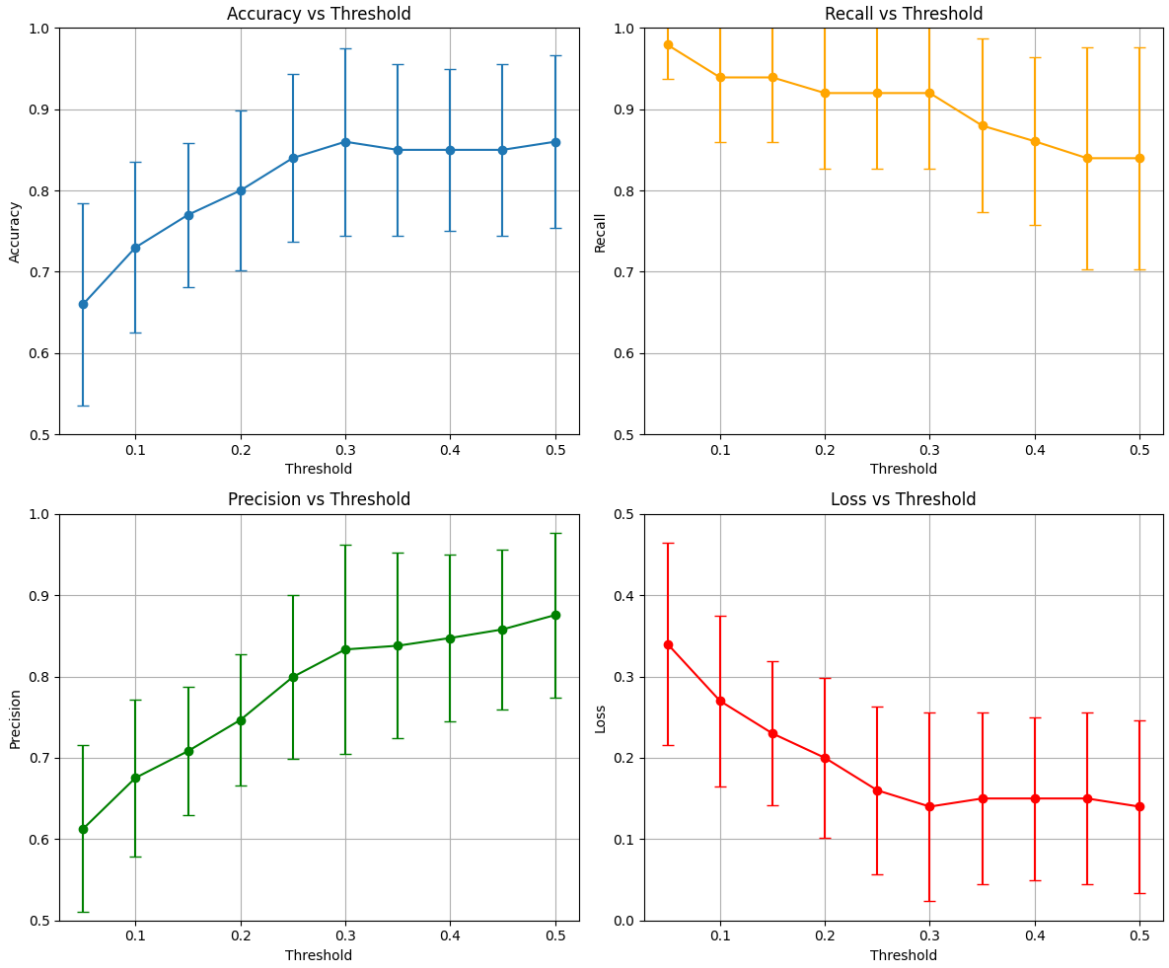


Figura 4.5: Confronto metriche aggregate al variare della threshold scelta. Oltre alle medie per ogni valore è stato riportato anche l'intervallo della deviazione standard.

set come riferimento indipendente per la stima finale delle performance. Anche questo comportamento è attribuibile al numero esiguo di bag nel test set, che rende il modello più sensibile a variazioni e amplifica la variabilità delle metriche osservate.

Capitolo 5

Conclusioni e lavori futuri

In questa tesi è stato affrontato il problema dell'analisi automatizzata di immagini istologiche attraverso una pipeline basata su tecniche di deep learning, articolata in due fasi principali: segmentazione cellulare ed elaborazione per la classificazione a livello di campione istologico.

Per quanto riguarda la segmentazione, è stato condotto un confronto tra modelli nello stato dell'arte, in particolare StarDist e Cellpose. I risultati hanno evidenziato come il fine-tuning dei parametri giochi un ruolo cruciale nel miglioramento delle prestazioni per entrambi i metodi. In generale, Cellpose si è dimostrato il modello più stabile e robusto, grazie alla sua maggiore capacità di generalizzazione su immagini non viste. Questo lo rende particolarmente adatto alla segmentazione di grandi quantità di immagini, anche a partire da un numero limitato di annotazioni. Tuttavia, è emerso come i tempi di calcolo rappresentino un fattore rilevante: Cellpose, sfruttando maggiormente la GPU, risulta più oneroso rispetto a StarDist, che rimane una valida alternativa in contesti con risorse computazionali limitate o dataset di grandi dimensioni.

È stato inoltre osservato come la scelta dei parametri di segmentazione debba essere effettuata con attenzione in funzione dell'obiettivo finale. In particolare, nel caso in cui le segmentazioni vengano utilizzate come input per modelli di classificazione, risulta preferibile adottare parametri globali, al fine di evitare l'introduzione di bias tra training e test set.

Nella seconda parte del lavoro è stata sviluppata una pipeline di classificazione basata su Multiple Instance Learning, che integra informazioni morfologiche derivate dalla

segmentazione con informazioni visive a livello di patch. Il modello, basato sul metodo Positive Instance Sampling (PINS), è stato opportunamente adattato al contesto applicativo e addestrato mediante validazione incrociata.

I risultati evidenziano una buona capacità discriminativa del modello, con metriche elevate su diversi fold di validazione incrociata. La variabilità osservata è principalmente attribuibile al numero limitato di bag disponibili, che rende le stime più sensibili alla suddivisione dei dati. La selezione della soglia è stata effettuata esclusivamente sul validation set, al fine di evitare fenomeni di *data leakage* e garantire una valutazione imparziale delle prestazioni.

Nel complesso, i risultati dimostrano la solidità della pipeline proposta, evidenziando come l'integrazione tra segmentazione cellulare e classificazione basata su Multiple Instance Learning rappresenti un approccio efficace per l'analisi di immagini istologiche.

Per quanto riguarda gli sviluppi futuri, si prospettano diverse direzioni di miglioramento. In primo luogo, l'utilizzo sistematico di Cellpose per la segmentazione potrebbe consentire di ottenere maschere più accurate e quindi feature morfologiche più informative. Inoltre, l'estrazione delle feature potrebbe essere ulteriormente arricchita includendo descrittori istologici più avanzati e informazioni contestuali. Dal punto di vista delle feature visive, l'impiego di modelli di feature extraction più avanzati rispetto a PathDINO potrebbe migliorare la rappresentazione delle patch. Infine, un'estensione naturale del lavoro consiste nell'applicazione della metodologia proposta a dataset più ampi e diversificati, al fine di valutarne la generalizzazione e la robustezza in contesti clinici differenti.

Bibliografia

[1] Jonathan Long, Evan Shelhamer e Trevor Darrell. *Fully Convolutional Networks for Semantic Segmentation*. 8 Mar. 2015.

[2] Christian Szegedy et al. *Going Deeper with Convolutions*. 2014.

[3] Wanni Xu, You-Lei Fu e Dongmei Zhu. «ResNet and Its Application to Medical Image Processing: Research Progress and Challenges». In: *Computer Methods and Programs in Biomedicine* 240 (2023), p. 107660.

[4] Vijay Badrinarayanan, Alex Kendall e Roberto Cipolla. *SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation*. 2016.

[5] Gao Huang et al. *Densely Connected Convolutional Networks*. 2018.

[6] Olaf Ronneberger, Philipp Fischer e Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 18 Mag. 2015.

[7] Liang-Chieh Chen et al. *DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs*. 2017.

[8] Hengshuang Zhao et al. *Pyramid Scene Parsing Network*. 2017.

[9] Robin M. Schmidt. *Recurrent Neural Networks (RNNs): A gentle Introduction and Overview*. 2019.

[10] Richard E. Turner. *An Introduction to Transformers*. 2026.

[11] Uwe Schmidt et al. *Cell Detection with Star-convex Polygons*.

[12] Martin Weigert e Uwe Schmidt. «Nuclei instance segmentation and classification in histopathology images with StarDist». In: *2022 IEEE International Symposium on Biomedical Imaging Challenges (ISBIC)*. 28 Mar. 2022.

[13] Simon Graham et al. *CoNIC: Colon Nuclei Identification and Counting Challenge 2022*. 2021.

[14] Juan C. Caicedo et al. «Nucleus Segmentation across Imaging Experiments: The 2018 Data Science Bowl». In: *Nature Methods* 16 (dic. 2019).

[15] Reka Hollandi et al. «nucleAIzer: A Parameter-free Deep Learning Framework for Nucleus Segmentation Using Image Style Transfer». In: *Cell Systems* 10.5 (20 mag. 2020), 453–458.e6.

[16] Carsen Stringer et al. *Cellpose: a generalist algorithm for cellular segmentation*.

[17] Politecnico di Torino. *Camel-Dataset*. URL: `https://www.kaggle.com/datasets/emanuelecarelli/camel-dataset`.

[18] Peter Bankhead et al. «QuPath: Open Source Software for Digital Pathology Image Analysis». In: *Scientific Reports* 7 (4 dic. 2017).

[19] Naylor Peter Jack et al. *Segmentation of Nuclei in Histopathology Images by Deep Regression of the Distance Map*. Zenodo, 16 feb. 2018.

[20] Amirreza Mahbod et al. «NuInsSeg: A Fully Annotated Dataset for Nuclei Instance Segmentation in H&E-Stained Histological Images». In: *arXiv preprint arXiv:2308.01760* (2023).

[21] *Portale Della Didattica*. URL: `https://didattica.polito.it/pls/portal30/gap.pkg_guide.viewGap?p_cod_ins=01URXOV&p_a_acc=2026&p_header=S&p_lang=IT&multi=N`.

[22] *OpenSlide Python: OpenSlide Python 1.4.3 Documentation*. URL: `https://openslide.org/api/python/`.

[23] *OpenCV: OpenCV Modules*. URL: `https://docs.opencv.org/4.x/`.

[24] Takuya Akiba et al. *Optuna: A Next-generation Hyperparameter Optimization Framework*. 2019.

[25] Mark Eastwood et al. «Malignant Mesothelioma subtyping via sampling driven multiple instance prediction on tissue image and cell morphology data». In: *Artificial Intelligence in Medicine* 143 (set. 2023).

[26] Saghir Alfasly et al. *Rotation-Agnostic Image Representation Learning for Digital Pathology*. 2024.

[27] Rene Y Choi et al. *Introduction to Machine Learning, Neural Networks, and Deep Learning*.

[28] Osval Antonio Montesinos López, Abelardo Montesinos López e Jose Crossa. «Fundamentals of Artificial Neural Networks and Deep Learning». In: Osval Antonio Montesinos López, Abelardo Montesinos López e José Crossa. *Multivariate Statistical Machine Learning Methods for Genomic Prediction*. Cham: Springer International Publishing.

[29] Giorgia Franchini e Marco Prato. *Dispense del corso di Computational and Statistical Learning*.

[30] IBM think. *La guida 2026 al machine learning*.

[31] Ying Yu et al. *Techniques and Challenges of Image Segmentation: A Review*.