UNIVERSITÀ POLITECNICA DELLE MARCHE

FACOLTÀ DI INGEGNERIA

Corso di laurea magistrale in Ingegneria Elettronica

STUDIO, ANALISI E SVILUPPO DI ARCHITETTURE DI FEDERATED LEARNING PER LA SEGMENTAZIONE DI IMMAGINI DI COLTURE CELLULARI

Study, analysis and development of Federated Learning architectures for the segmentation of cell culture images

Relatore:
Prof.ssa Pierleoni Paola

Correlatori:
Dott. Esposito Marco
Dott.ssa Raggiunto Sara

Candidato:
Sara Bruschi

Anno Accademico 2023-2024

Abstract

In questa tesi sono state progettate ed implementate architetture di Federated Learning (FL) per la segmentazione automatica di immagini di cellule ottenute da microscopio, un'attività fondamentale in ambito medico e biologico per la diagnosi ed il monitoraggio di diverse patologie. Attualmente l'analisi di queste immagini viene condotta principalmente in modo manuale, con problematiche legate a soggettività dell'annotazione e a un elevato dispendio di tempo e risorse. Ulteriori criticità si riscontrano nella condivisione sicura delle immagini tra laboratori ed organizzazioni differenti.
In questo lavoro sono state valutate diverse reti neurali per segmentare immagini ottenute da un esperimento che coinvolge linee cellulari di Non-Small Cell Lung Cancer (NSCLC), trattate con la citochina TGF-β per indurre la trasformazione epitelio-mesenchimale. Individuato il modello più performante in relazione a tali immagini e affinati i parametri, è stato sviluppato un tool per la segmentazione cellulare e per l'analisi automatica dei parametri morfologici e di fluorescenza.
Per affrontare le criticità legate alla privacy e alla condivisione di grandi dataset di training tipiche dell'addestramento centralizzato di reti neurali, è stato implementato un approccio distribuito di Federated Learning. Il FL è una tecnica sicura di training distribuito in cui il task di addestramento è trasferito tra diversi client, senza la condivisione di dati tra gli stessi. Questo permette di mantenere i dati sensibili nei laboratori locali, garantendo la privacy dei pazienti, facilitando la collaborazione tra enti di ricerca e distribuendo il carico di lavoro tra diverse strutture interconnesse. I risultati ottenuti con l'architettura di addestramento distribuito dimostrano che le prestazioni del modello addestrato con FL sono comparabili a quelle dell'addestramento centralizzato, con loss finale del modello distribuito perciò molto vicina a quella del modello centralizzato. Questo dimostra che il Federated Learning è in grado di garantire performance simili all'approccio classico, assicurando al contempo vantaggi legati alla privacy e alla gestione decentralizzata dei dati.
Il lavoro illustrato rappresenta, dunque, un passo in avanti nell'utilizzo di tecniche di intelligenza artificiale sicure ed efficienti in ambito medico e biologico, con particolare focus sull'ottimizzazione di aspetti quali la condivisione dei dati, l'apprendimento continuo (*continual learning*) e la semplificazione dei task manuali di annotazione.

Indice

Introduzione

La segmentazione di immagini di colture cellulari ottenute da microscopio è una pratica fondamentale in ambito medico e biologico, poiché prende parte soprattutto allo sviluppo di diagnosi e al monitoraggio di malattie. Ad oggi l'analisi e il processamento di tali immagini è ancora eseguito per lo più in maniera manuale, richiedendo un impegno significativo sia da parte di operatori di laboratorio che di medici, i quali da esse estraggono manualmente caratteristiche e informazioni utili. Tuttavia, questa tecnica non richiede solo un notevole dispendio di tempo e risorse, ma è anche inadatta a gestire l'enorme quantità di immagini di questo tipo che vengono prodotte ogni giorno. Inoltre, lo studio e l'estrazione manuale delle informazioni risulta soggetta a variabilità e soggettività umana, rendendola una pratica non ideale.
Questi fattori hanno contribuito a rendere la segmentazione cellulare un ambito di ricerca su cui molti studiosi si sono focalizzati. L'individuazione e l'implementazione di metodi automatici a supporto della segmentazione di immagini di colture cellulari eseguita da operatori, risulta essere di fondamentale importanza sia allo scopo di migliorarne l'efficienza e l'affidabilità, sia per cercare di automatizzare e rendere oggettivo tale processo.
In questo contesto il Deep Learning (DL), una ramificazione del Machine Learning (ML) che a sua volta è una branca dell'Intelligenza Artificiale (AI), assume un ruolo fondamentale nel rivoluzionare il campo della segmentazione di immagini. L'impiego di reti neurali permette di ottenere risultati che si avvicinano a quelli ottenuti dalla segmentazione manuale e spesso superano le prestazioni dei metodi tradizionali presenti nello stato dell'arte. Questi ultimi, infatti, realizzano algoritmi di Computer Vision che eseguono il riconoscimento delle cellule all'interno delle immagini basandosi sull'intensità dei pixel o sulle relazioni spaziali, i quali presentano limitazioni nella gestione di immagini complesse e diverse tra loro, rendendo necessario l'utilizzo di approcci più avanzati come, appunto, il DL.
L'addestramento delle reti neurali tipicamente segue un approccio centralizzato, per il quale i dati su cui la rete deve essere addestrata vengono raccolti all'interno di un server centrale, al fine di costruire il dataset di training. Per questo motivo la qualità delle predizioni ottenute dai modelli richiede la disponibilità di grandi dataset annotati, ovvero costituiti da coppie immagine-maschera. Queste ultime, tuttavia, risultano difficili da ottenere per la già menzionata necessità di impiegare la segmentazione manuale in una grandissima quantità di immagini. Inoltre, l'approccio classico risulta inadatto poiché in ambito medico la privacy e la sicurezza dei dati devono essere rispettate per preservare i dati sensibili dei pazienti, rappresentando un'ulteriore criticità.
A causa di queste problematiche risulta necessario un cambio di approccio costituito dall'adozione del Federated Learning (FL) che utilizza un sistema decentralizzato eseguendo un training distribuito. Con l'impiego di questa tecnica si è in grado di superare le criticità

poste dal paradigma precedente, poiché è il training ad essere trasferito ai dati (ovvero ai dispositivi locali) e non viceversa. Questo significa che la rete neurale viene addestrata sul dataset del singolo dispositivo, senza richiedere che le immagini vengano inviate e condivise ad un server centrale, garantendo il mantenimento della privacy per i dati sensibili di ogni paziente e riducendo il bisogno di aggregare grandi quantità di immagini all'interno di un unico bacino centrale. Questo approccio è particolarmente utile nel campo della segmentazione cellulare perché, tramite la decentralizzazione, consente una distribuzione del carico di lavoro tra i diversi laboratori partecipanti, riducendo le risorse computazionali richieste al singolo client.

Il presente lavoro ha come obiettivo lo sviluppo e l'implementazione di un tool per la segmentazione automatica ed oggettiva di immagini di colture cellulari. Poiché il tool si basa su reti neurali addestrate con un approccio centralizzato, questo obiettivo comprende l'applicazione del FL allo stesso task, confrontando le prestazioni del modello addestrato con entrambi gli approcci e dimostrando che siano comparabili. Questo consente di superare le limitazioni legate alla mancanza di privacy ed alla scarsità dei dati, mantenendo comunque le stesse performance dell'approccio classico.

Infine si ottiene un tool che realizza segmentazione cellulare di immagini di microscopia, le quali possono essere ulteriormente analizzate allo scopo di ottenere informazioni sia di tipo morfologico che di fluorescenza, utili allo studio, ad esempio, della progressione di malattie. Con il cambio di paradigma questo lavoro dimostra che il FL è una soluzione vantaggiosa per i contesti in cui la protezione dei dati e la gestione decentralizzata delle immagini risultano essere concetti fondamentali, pur mantenendo lo stesso livello di accuratezza dei risultati.

1 Stato dell'arte

1.1 Intelligenza Artificiale e Machine Learning

L'intelligenza artificiale (AI) è l'intelligenza delle macchine: si parla di AI quando i dispositivi sono in grado di mimare le funzioni cognitive che caratterizzano la mente umana, come l'apprendimento e la risoluzione dei problemi. Tra gli obiettivi presenti nell'ambito di ricerca riguardante l'intelligenza artificiale si hanno il ragionamento, la conoscenza, la previsione, l'apprendimento, la percezione e l'abilità di muovere e manipolare oggetti. L'ascesa dell'AI ha a che fare anche con la grandissima quantità di dati di diverso tipo che necessita di essere elaborata, incluse immagini, video, audio, testo, transazioni etc. Il machine learning (ML) è un sottocampo della computer science che "da' ai computer l'abilità di imparare senza essere programmati esplicitamente per farlo". Il ML studia la costruzione di algoritmi che possono apprendere dai dati le loro caratteristiche, al fine di effettuare predizioni sugli stessi; è dunque un metodo utilizzato per ideare modelli ed algoritmi complessi che effettuano previsioni [1].
I sistemi intelligenti che mettono a disposizione le capacità dell'intelligenza artificiale si basano spesso sul machine learning, il quale permette loro di apprendere da dati di training adatti allo specifico problema che si vuole risolvere, al fine di automatizzare il processo. Il ML si pone l'obiettivo di imparare in modo automatico relazioni e schemi a partire da esempi ed osservazioni. In questo senso esso rimuove dall'umano la necessità di spiegare e di formalizzare la propria conoscenza in un formato che sia comprensibile per la macchina e che permetta di sviluppare sistemi intelligenti in modo efficiente. Il machine learning, infatti, nel corso degli ultimi decenni ha permesso di ottenere tecniche di apprendimento e di pre-processing dei dati efficienti. L'intelligenza artificiale comprende tutte le tecniche che permettono ai computer di mimare il comportamento umano e di riprodurre, o superare, il processo decisionale necessario a risolvere problemi complessi indipendentemente, o dipendente in piccola parte, dall'azione umana. Nel ML il programma realizzato all'interno della macchina ha prestazioni che migliorano con l'esperienza e lo scopo consiste nell'automatizzare il task del modello analitico imparando da calcoli precedenti ed estraendo informazioni da grandi database che permettano di prendere decisioni ripetibili ed affidabili [2].
Le decisioni che gli algoritmi di machine learning eseguono autonomamente si basano su dei pattern che possono essere evidenziati tra dati complessi. A seconda del tipo di dato in input e in output, e del tipo di problema che gli algoritmi devono risolvere, si possono avere diversi approcci all'apprendimento che determinano diverse categorie di algoritmi di machine learning. Tra questi si hanno: supervised learning, unsupervised learning e reinforcement learning, a cui si aggiungono anche approcci ibridi [3].

- Supervised Learning: applicato quando i dati sono nella forma di variabili di input

e target di output. L'algoritmo in questo caso apprende effettuando un mapping tra input e output. La necessità di un gran numero di dati etichettati rende costoso questo approccio nel caso di task con una disponibilità di dati scarsa. A sua volta può essere suddiviso in altre due categorie:

- Classificazione: la variabile di output è una di un certo numero di categorie note (ad esempio cane, gatto o positivo, negativo).
- Regressione: la variabile di output è un valore reale o continuo (ad esempio prezzo o posizione geografica).

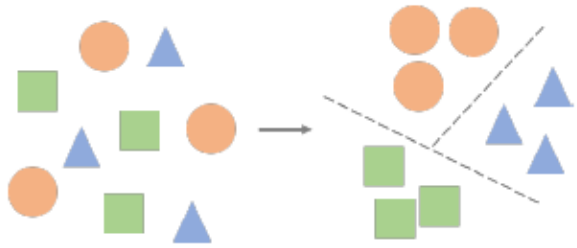


Figura 1: Supervised Learning: gli input sono categorizzati in un insieme di classi [3]

- Unsupervised Learning: applicato quando i dati sono disponibili soltanto nel formato di un input e non si ha a disposizione una corrispondente variabile di output. Questi tipi di algoritmi modellano gli schemi presenti nei dati al fine di apprendere le loro caratteristiche. Uno degli algoritmi principali di tipo unsupervised è il clustering dove i gruppi coerenti con i dati vengono ottenuti e poi utilizzati per effettuare predizioni in output su dati mai visti (ad esempio prevedere le abitudini di acquisto di un consumatore).

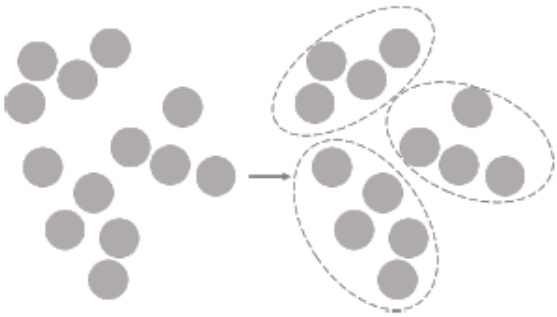


Figura 2: Unsupervised Learning: i dati in input sono raggruppati in cluster basati sugli schemi evidenziati dai primi [3]

- Reinforcement Learning: utilizzato quando il task coincide con la necessità di realizzare una sequenza di decisioni al fine di ottenere una ricompensa finale. Durante il processo di apprendimento un agente artificiale viene ricompensato o penalizzato a seconda delle azioni che esegue. Il suo scopo è quello di massimizzare la ricompensa

ottenuta al termine dell'apprendimento. Un esempio è rappresentato da agenti che giocano ai videogame o che eseguono task robotici con un obiettivo.

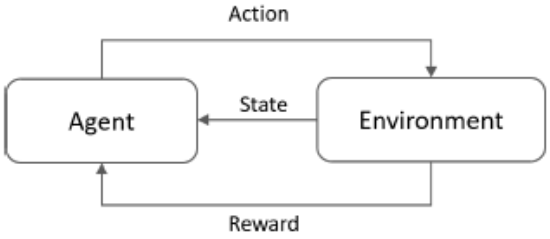


Figura 3: Reinforcement Learning: un agente osserva lo stato ed esegue azioni per massimizzare la ricompensa [3]

Tra gli approcci ibridi sono presenti: il semi-supervised learning in cui gli algoritmi sono addestrati utilizzando una combinazione di dati etichettati e non; il self-supervised learning che è una forma di unsupervised learning dove i dati di addestramento vengono etichettati in modo automatico e non in modo manuale e il self-taught learning utilizzato per risolvere task di supervised learning utilizzando dati etichettati e non, dove questi ultimi non condividono le stesse label [3].

1.1.1 Deep Learning e Reti Neurali

Il Deep Learning (DL) è una tecnica utilizzata per implementare algoritmi di machine learning sfruttando reti neurali "multi-layer". Questi strati eseguono un processamento multiplo di dati ed apprendono la loro rappresentazione attraverso diversi livelli di astrazione al fine di comprendere gli input [3].
Per introdurre le reti neurali occorre parlare di Artificial Neural Networks (ANN), le quali nascono dall'osservazione di come il cervello umano esegue calcoli su processi complessi in modo totalmente diverso dal modo in cui vengono eseguiti dai computer convenzionali. In generale, il funzionamento del cervello umano e animale permette di eseguire task complessi in un tempo molto ridotto e con grande efficienza. Una delle caratteristiche dei neuroni biologici che permette questo è il fatto di essere collegati a tutti gli altri da cui ricevono stimoli ogni volta che apprendono qualcosa di nuovo. I neuroni ricevono ed inviano segnali elettrici contenenti l'informazione che devono scambiare con quelli a cui sono direttamente collegati. Le ANN sono macchine realizzate per eseguire compiti specifici imitando il funzionamento del cervello e, a questo scopo, sono realizzate partendo da centinaia o migliaia di neuroni artificiali che fungono da unità di calcolo. In questo modo le reti realizzano algoritmi di apprendimento in grado di costruire le proprie regole di comportamento sfruttando l'"esperienza". L'algoritmo che permette alla rete di apprendere consiste nella modifica dei pesi delle sinapsi della rete in modo sequenziale e supervisionato, al fine di raggiungere un obiettivo specifico. I neuroni, lavorando insieme,

sono in grado di imparare le relazioni lineari e non lineari tra input e output utilizzando procedure di addestramento sequenziali. Gli elementi principali che costituiscono un modello di ANN sono: l'informazione (input) che il neurone riceve da sistemi esterni o da altri neuroni ad esso connessi; il vettore dei pesi della sinapsi che modifica l'informazione ricevuta emulando il comportamento dei neuroni biologici; l'output del neurone, generato valutando l'input della rete (ottenuto da un'unione dei due elementi precedenti) con la "funzione di attivazione", la quale determina l'attività o meno del neurone stesso. I segnali ottenuti vengono poi trasferiti tra i neuroni attraverso delle connessioni, le quali a loro volta hanno associati dei pesi che moltiplicano il segale trasmesso. Una ANN, dunque, è un sistema composto di tanti semplici elementi che processano i dati in parallelo e le cui funzioni sono determinate dalla struttura della rete e dal peso delle connessioni stesse [4].
Il Deep Learning è un sottogruppo del machine learning che fa riferimento alle deep artificial neural networks, dove il termine "deep" riguarda il numero di strati che costituiscono la rete neurale [5].
Il DL, infatti, utilizza ANN nelle quali sono presenti più di uno strato, il che implica che più neuroni vengono impiegati per l'implementazione del modello. Per questo motivo questo tipo di rete prende il nome di Deep Neural Network (DNN) e il tipo di addestramento che esegue è chiamato deep learning. In esso una topologia multi strato permette di mappare le relazioni tra le variabili in input e le variabili in output [4].
Per riassumere il legame che esiste tra l'AI e le diramazioni di cui si compone è possibile osservare lo schema in Figura 4.

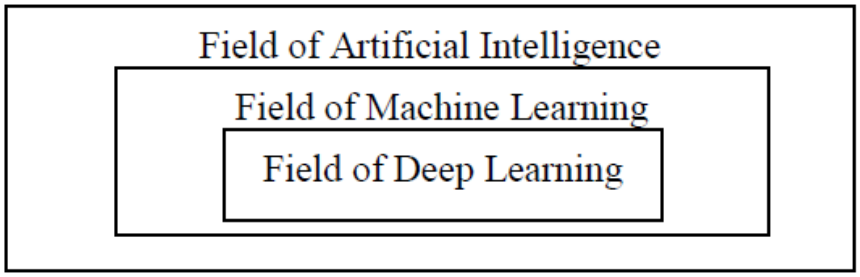


Figura 4: Relazione tra Intelligenza Artificiale, Machine Learning e Deep Learning [5]

La rete neurale convoluzionale (CNN) è la rete più utilizzata nel campo del deep learning: gli algoritmi di computer vision basati su CNN hanno permesso di raggiungere risultati considerati irraggiungibili in passato, tra cui il riconoscimento facciale, i veicoli autonomi, i supermercati self-service e trattamenti medici intelligenti [6].
Le CNN sono una forma di ANN utilizzate per risolvere problemi basati sul riconoscimento di immagini. Le prime, come le seconde, hanno una struttura basata su neuroni ottimizzabili attraverso l'addestramento: l'input è un vettore multidimensionale che viene distribuito attraverso gli hidden layers (più di uno nel DL) i quali eseguono le decisioni. La più grande differenza tra queste due architetture è che le CNN sfruttano le immagini

per il riconoscimento di pattern, permettendo di identificare caratteristiche specifiche al loro interno. Si rende necessario questo adattamento ai task basati su immagini perché le ANN presentano difficoltà sulla complessità computazionale richiesta dai dataset basati su immagini. Per questo motivo, il focus principale delle CNN è sull'input, il quale sarà costituito da vettori tridimensionali, richiedendo che nell'architettura i neuroni presentino anch'essi tre dimensioni (altezza, larghezza e profondità) [7].

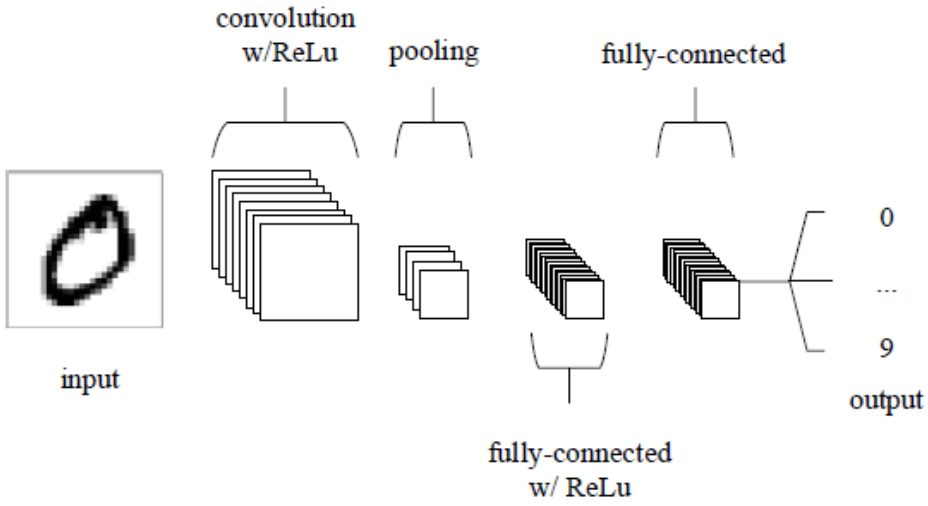


Figura 5: Architettura semplificata di una rete CNN composta di cinque layer [7]

1.1.2 Segmentazione Semantica

Uno dei problemi più difficili nel campo della computer vision è la segmentazione semantica, la quale si differenzia dal problema della classificazione di immagini o dal riconoscimento di oggetti dato che non è necessario conoscere il concetto o quale sia l'oggetto. La segmentazione semantica è l'abilità di segmentare un'immagine sconosciuta in parti o regioni diverse. Un algoritmo di questo tipo al manifestarsi di una nuova immagine dovrebbe restituire un insieme di pixel che appartengono semanticamente allo stesso gruppo. I metodi più recenti di segmentazione che ne costituiscono lo stato dell'arte possono essere divisi in tre categorie: segmentazione semantica basata su regioni, segmentazione semantica basata su FCN (Fully Convolutional Network) e segmentazione debolmente supervisionata. La prima segue l'approccio della segmentazione che si appoggia al riconoscimento, dunque estrae prima delle regioni di forma qualunque e le descrive effettuando una classificazione basata su regione, eseguendo poi delle predizioni sui pixel. La seconda prevede che la FCN apprenda un mapping tra i pixel senza necessità di estrarre le regioni, così da poter effettuare previsioni su input di dimensioni arbitrarie. L'ultimo metodo utilizza delle bounding box annotate o etichette a livello di immagine per rispondere alla necessità di un gran numero di immagini annotate (maschere) necessarie a compiere il ta-

sk, operazione impegnativa e costosa da eseguire manualmente. La capacità di performare un'annotazione automatica delle immagini può avere vantaggi sia pratici che teorici:

1. Aumenta l'efficienza computazionale;
2. Aumenta l'accuracy eliminando il rumore di fondo;
3. Permette di ottenere informazioni sia sul funzionamento dei sistemi visuali che sulle limitazioni degli algoritmi di segmentazione;
4. Può diventare più generalizzato di algoritmi di riconoscimento o di individuazione di oggetti.

Per quanto riguarda le sfide e le debolezze degli algoritmi di segmentazione semantica, esse sono:

- Non è ancora chiaro quanto bene performano gli algoritmi descritti su immagini generiche;
- Richiedono una enorme quantità di dati etichettati per poter essere addestrati in modo adeguato;
- Per l'addestramento è richiesta una grande capacità computazionale, non disponibile in molti contesti;

In conclusione, la segmentazione di immagini ha compiuto grandi passi avanti negli ultimi anni e risulta essere un campo di ricerca che permetterebbe di fornire grande contributi nel campo dell'analisi di immagini [8].

1.2 Machine Learning Centralizzato vs Distribuito

1.2.1 Machine Learning Centralizzato

Nel machine learning il modello (rete neurale) viene addestrato utilizzando i dati a disposizione al fine di compiere un task (individuazione di oggetti, trascrizione di un audio...). I dati non vengono originati dalla macchina all'interno della quale avviene l'addestramento, ma prendono vita in dispositivi esterni come ad esempio smartphone o pc di utenti utilizzatori. Di conseguenza per realizzare l'algoritmo di ML è necessario raccoglierli tutti all'interno di un server centrale. A questo punto risulterà poi possibile addestrare il modello su di essi [9].
Oggi la quantità di dati generata da ogni tipo di dispositivo è enorme ed ancora in crescita; la rapida evoluzione del DL supporta ancora approcci basati su un'architettura centralizzata dove i dati vengono memorizzati e processati. Questo tipo di applicazione presenta però delle problematiche: latenza elevata, costo computazionale in crescita e mancato mantenimento di privacy e di sicurezza dei dati. L'addestramento di tipo centralizzato è l'approccio convenzionale che prevede un flusso di dati continuo verso il cloud, all'interno del quale possono essere analizzati ed utilizzati per l'estrazione delle features. In particolare, sono impiegati nell'addestramento di modelli ad alte performance direttamente sul server. In questa modalità, però, la privacy dei dati degli utenti risulta essere un problema, poiché con il loro spostamento non può essere mantenuta [10].
L'approccio classico, utilizzato tipicamente in casi dove i dati sono naturalmente disponibili su un server centrale (come l'analisi del traffico web) non può essere utilizzato in alcune situazioni per diversi motivi:

1. Esistono regolamentazioni che impediscono ai dati sensibili di essere spostati (GDPR in Europa) e, dunque, a singole organizzazioni di unire i dati di addestramento quando gli utenti vivono in diverse parti del mondo, poiché i loro dati saranno protetti da regolamentazioni differenti;

2. In diverse applicazioni è proprio l'utente a non volere la diffusione dei propri dati a partire dal dispositivo di appartenenza (ad esempio password o coordinate bancarie);

3. Per il grande volume di dati prodotto dai sensori non risulta fattibile né economico collezionarli tutti in un'unica posizione.

Esempi di applicazioni in cui il ML centralizzato non risulta applicabile né funzionante per i motivi sopra elencati sono: cartelle sanitarie ospedaliere, informazioni finanziarie di organizzazioni differenti, dati di localizzazione provenienti da auto, messaggi cifrati end-to-end eccetera [9].
Per superare i problemi del learning classico si introduce un approccio differente, il quale si pone l'obiettivo di preservare la privacy degli utenti e di ridurre il grande overhead dovuto alla raccolta dei dati [10].

1.2.2 Machine Learning Distribuito

Il Federated Learning (FL) è un approccio decentralizzato che permette di preservare la privacy mantenendo il dato grezzo nel dispositivo dell'utente, all'interno del quale si esegue un addestramento locale. Questo consente anche una riduzione dell'overhead dovuto alla comunicazione dei dati [10].
In questo senso il FL inverte l'approccio classico permettendo di eseguire machine learning su dati distribuiti.

- Centralized Machine Learning: porta i dati alla computazione;
- Federated (Machine) Learning: porta la computazione ai dati.

Questo permette di utilizzare il ML e qualsiasi altro approccio di data science anche in quelle aree in cui prima non risultava possibile [9].
Il federated learning è una tecnica sicura di machine learning distribuito che esegue algoritmi su dispositivi sparsi, assicurandosi che le informazioni private rimangano in locale. In questo senso, il FL trasferisce il task di addestramento ad ogni client e, di conseguenza, la comunicazione tra client e server prevede soltanto uno scambio di parametri piuttosto che di dati. Il ruolo del secondo si riduce a quello di aggregazione dei parametri, con lo scopo di aggiornare il modello globale. Questo tipo di schema permette di proteggere i dati degli utenti oltre che di ridurre lo sforzo computazionale e le risorse di storage richieste dal server. Come già anticipato l'approccio presentato entra in contrasto con il ML centralizzato, poiché i dataset locali non vengono raccolti in un "bacino" centrale, ma utilizzati singolarmente. Nonostante questa diversità le prestazioni del modello "federato" sono simili a quelle del modello addestrato su dati condivisi e raccolti su un unico cloud centrale [11].

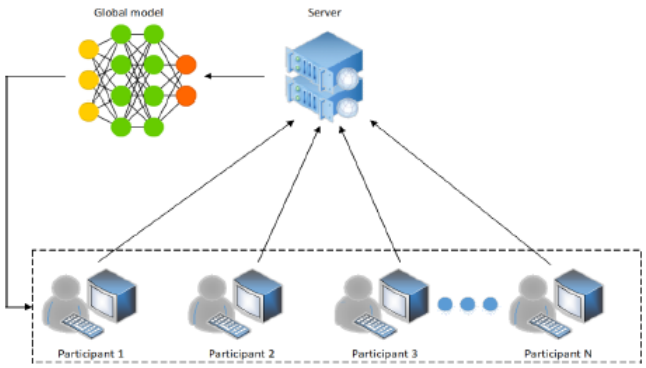


Figura 6: Framework base del Federated Learning [11]

Il federated learning è dunque una struttura distribuita in cui può essere costruito un "modello virtuale" che si incarica della collaborazione tra i client senza necessità di condividere dati grezzi. Questo è un modello globale ottimo che compie aggregazione su ciò che

riceve da ogni partecipante; quello che si ottiene in questo modo viene poi utilizzato dai client come modello di partenza ad ogni nuovo training. Per riassumere, il FL fornisce un meccanismo di protezione della privacy dei dati che assicura la trasmissione del modello senza necessità di esporre dati sensibili [11].
Come già accennato, questo consente di ottenere un avvicinamento al mondo del deep learning anche da parte di numerose applicazioni precedentemente penalizzate dalla non accessibilità dei dati. Il FL si basa su cinque step fondamentali:

Step 0 Inizializzazione del modello globale: è analogo a ciò che avviene nel centralizzato, dove si inizializzano i parametri del modello in modo randomico o utilizzando dei pesi precedentemente salvati.

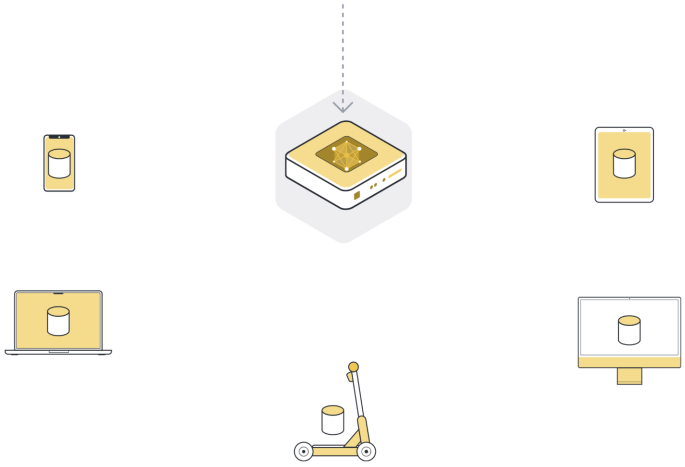


Figura 7: Inizializzazione del modello globale [9]

Step 1 Invio del modello ai nodi client (organizzazioni/dispositivi collegati al server): necessario per far sì che tutti i nodi partecipanti eseguano l'addestramento locale partendo dagli stessi parametri.

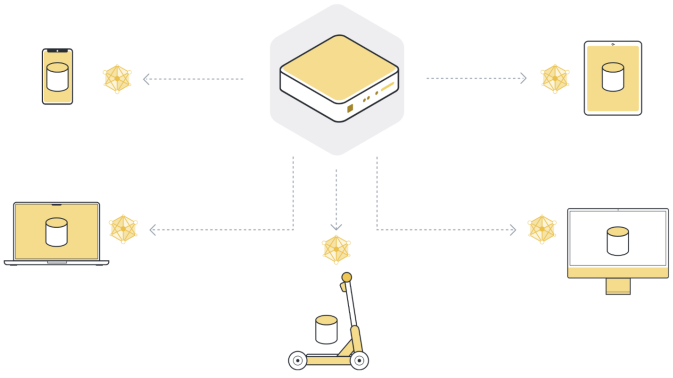


Figura 8: Invio del modello globale [9]

Step 2 Addestramento del modello in locale sui dati del nodo client: una volta che ognuno possiede l'ultima versione dei parametri del modello globale si può eseguire il training sul dispositivo.

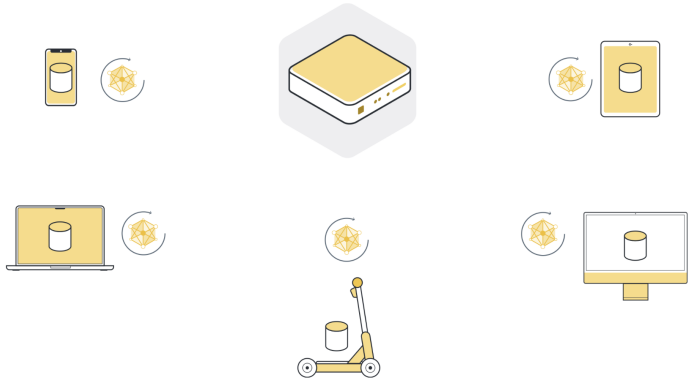


Figura 9: Addestramento locale [9]

Step 3 Restituzione al server degli aggiornamenti del modello: dopo il training locale ogni client ha una diversa versione dei parametri del modello a causa della diversità dei dataset. Tutti i nuovi aggiornamenti vengono inviati al server.

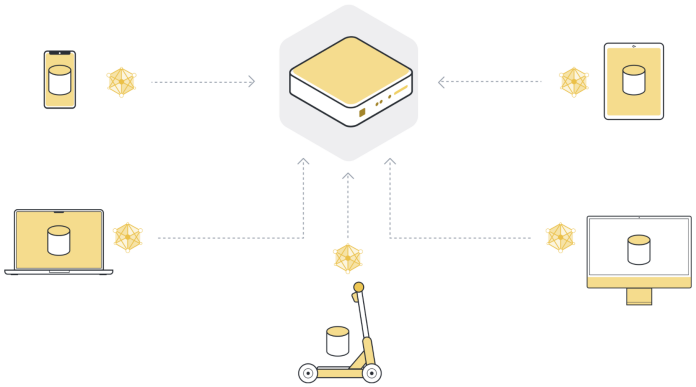


Figura 10: Restituzione dei parametri al server dopo l'addestramento locale [9]

Step 4 Aggregazione degli aggiornamenti in un nuovo modello globale: per ottenere un singolo modello i pesi di ogni client vengono combinati durante un processo chiamato "aggregazione". L'approccio base è rappresentato dal "Federated Averaging" (*FedAvg*) che esegue una media pesata degli update ricevuti, dove il peso è dato dal numero di dati utilizzato da ogni client per l'addestramento. In questo modo si assicura che ogni aggiornamento abbia la giusta influenza sul modello globale risultante.

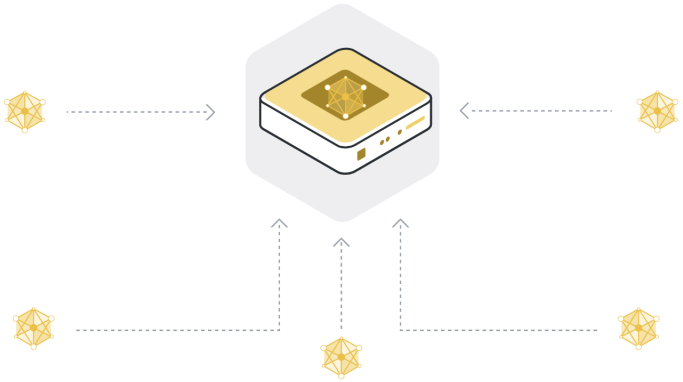


Figura 11: Aggregazione degli aggiornamenti al modello [9]

Step 5 Ripetizione degli step 1-4 fino alla convergenza del modello: la sequenza degli step precedenti costituisce un round di FL. Durante un round ogni nodo client partecipa all'iterazione tramite l'addestramento, dunque a seguito dell'aggregazione si ottiene un modello che è stato addestrato su tutti i dati dei nodi partecipanti, anche se solo per poche epoche. Il processo va ripetuto per arrivare ad un modello che abbia buone performance sui dati di tutti i client.

Alcuni esempi di applicazione in cui il FL ha favorito l'utilizzo del machine learning superando le problematiche elencate, sono modelli AI medicali che permettono a diversi ospedali di lavorare insieme o modelli AI in grado di risolvere frodi finanziarie se addestrati su dati di istituzioni diverse [9].
In Figura 12 viene presentata una schematizzazione della successione degli step che costituiscono un round di Federated Learning.

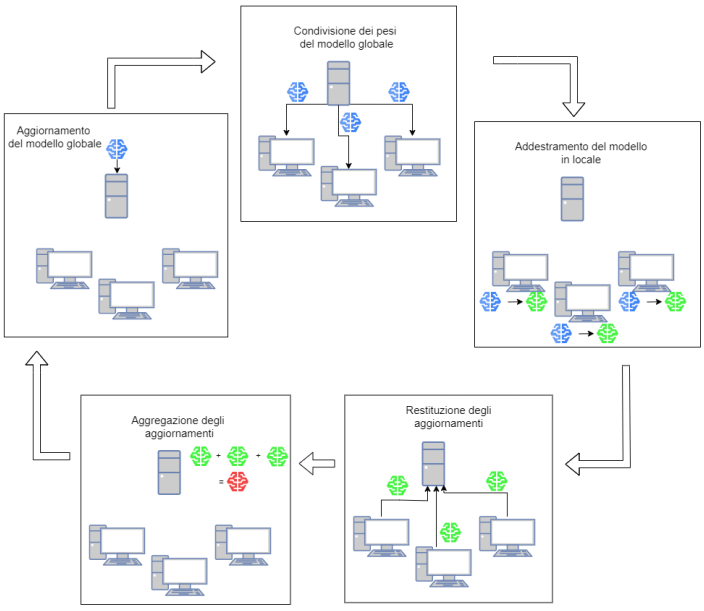


Figura 12: Schematizzazione di un round di Federated Learning

In conclusione, l'utilizzo e l'implementazione del Federated Learning rappresenta un grande passo avanti nell'ambito del Machine Learning poiché permette di preservare la privacy in ambiti in cui i dati di ogni client/istituzione sono sensibili, e permette anche di limitarne la trasmissione, operazione che richiede uno sforzo computazionale piuttosto elevato.

1.3 Segmentazione Cellulare

L'analisi computerizzata di immagini ottenute da microscopio assume un ruolo importante nella diagnosi e nella prognosi di malattie assistite dall'utilizzo del computer come, ad esempio, tumore del seno, dei polmoni, del cervello etc. Il Deep Learning rappresenta un tool fondamentale nella computer vision, riuscendo ad essere di supporto anche nell'analisi di immagini biomedicali. Al giorno d'oggi l'elevatissima quantità di immagini rende inefficiente, o addirittura impossibile, il loro processamento manuale, di conseguenza i metodi che coinvolgono l'utilizzo del ML ne migliorano l'efficienza e l'oggettività. Il Machine Learning acquisisce conoscenza dalla rappresentazione dei dati, dunque non interagisce direttamente con il dato grezzo; il Deep Learning, invece, processa direttamente il dato grezzo e ne apprende la rappresentazione. Questo risulta utile nell'individuazione di oggetti, nella segmentazione di immagini o nella classificazione di un target. Confrontando questo approccio con la rappresentazione manuale delle caratteristiche risulta evidente che l'utilizzo del DL richiede un intervento umano minore, consentendo anche di ottenere performance migliori. In questo senso il Deep Learning risulta essere una soluzione per l'analisi di immagini di microscopia. Queste devono essere fornite, insieme alle corrispondenti versioni etichettate, ad un modello di CNN che possa essere addestrato al fine di generare la rappresentazione di nuovi dati. L'addestramento può essere sia "non supervisionato" (il che non richiederebbe l'annotazione da parte dell'uomo) che "supervisionato". Nell'ambito della segmentazione delle immagini, specialmente nella segmentazione del nucleo o della cellula, risulta importante avere a disposizione numerose immagini associate alle proprie label, necessarie per l'addestramento del modello [12].
L'utilizzo di reti neurali può migliorare in modo significativo le prestazioni dell'analisi di immagini ottenute da microscopio. A seguito dell'addestramento basta fornire al modello addestrato una nuova immagine di colture cellulari, senza modificarne le caratteristiche o il design, al fine di ottenere in uscita un'etichettatura ad essa corrispondente [13].

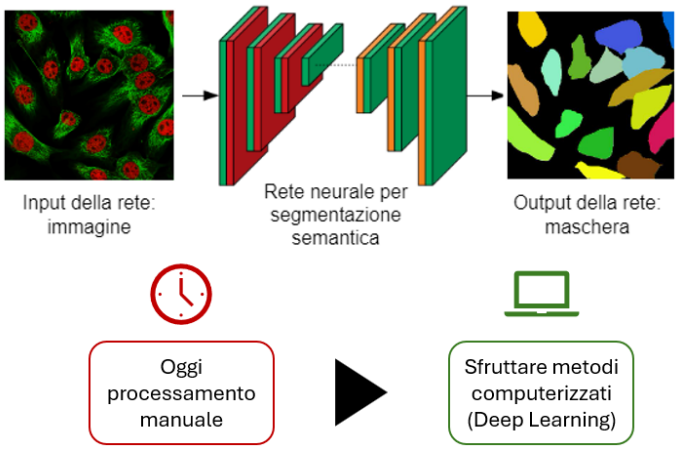


Figura 13: Segmentazione cellulare: dal processamento manuale all'utilizzo di metodi computerizzati

Molte delle linee cellulari attualmente utilizzate nella ricerca medica, come ad esempio cellule di cancro o cellule staminali, crescono in colonie confluenti. La biologia di ogni cellula permette di ottenere informazioni importanti, dunque la loro separazione all'interno delle immagini è critica al fine di poterle contare, identificare e misurare. La segmentazione delle cellule oggi è ancora un problema a causa degli alti livelli di rumore presenti nelle immagini: risulta necessario avere un metodo che sia robusto su una grande varietà di immagini biologiche e che sia in grado di separare le cellule in modo accurato. Ogni cellula è costituita da un differente fenotipo: diversi modelli di migrazione, diverse dimensioni eccetera, per cui riuscire ad identificare quello di ognuna contribuirebbe ad una migliore comprensione dei fenomeni biologici (metastasi di tumori, differenziazione di staminali o plasticità cellulare) [14].
I metodi tradizionali per la segmentazione cellulare sono basati sull'intensità dei pixel e sulle relazioni spaziali, ma presentano delle limitazioni. Negli ultimi tempi, come già menzionato, i metodi di ML e DL consentono di ottenere delle soluzioni più efficienti ed accurate, dato che permettono di realizzare dei tool software che eseguono la segmentazione di immagini di colture cellulari. Per realizzare il task richiesto è necessario effettuare un'analisi della morfologia delle cellule, della loro posizione, ottenere una conta delle cellule presenti ed individuare variazioni intracellulari. Le tecniche più comuni per la raccolta di immagini includono la microscopia in campo chiaro, la microscopia di fluorescenza, la microscopia confocale e la microscopia a contrasto di fase. Nei metodi tradizionali l'annotazione precisa della tipologia di ogni cellula è richiesta per effettuarne la segmentazione, poiché richiede un'ottimizzazione manuale ed un adattamento del codice, rendendola quindi altamente soggettiva. Gli algoritmi fondamentali per realizzare dei tool in grado di effettuare segmentazione stanno gradualmente diventando reti di Deep Learning sempre più complesse: inizialmente tool più basici come CellProfiler utilizzavano algoritmi integrati di tipo watershed. Successivamente si sono iniziati a realizzare tool che integrassero modelli di DL per un miglioramento delle prestazioni [15].

1.3.1 CellProfiler

I microscopi avanzati permettono di collezionare migliaia di immagini cellulari ogni giorno attraverso esperimenti in time-lapse. Esaminare tali immagini è richiesto per studiare le funzioni cellulari e questo rappresenta la sfida che devono affrontare gli algoritmi di segmentazione cellulare. CellProfiler è il primo sistema gratis ed open-source progettato per l'analisi flessibile e ad alto rendimento di immagini cellulari. CellProfiler affronta una varietà di questioni biologiche, inclusi test standard (come conta di cellule, dimensioni delle cellule e livelli di proteina per cellula) ed analisi morfologiche complesse (come dimensioni delle cellule/degli organelli o schemi subcellulari di DNA o colorazione delle proteine). Fino all'avvento di CellProfiler l'analisi di immagini ad alto rendimento ha rappresentato un problema affrontato solo tramite soluzioni custom realizzate da esperti. Da qui deriva

la necessità di una piattaforma open-source potente e flessibile. CellProfiler misura simultaneamente la dimensione, la forma, l'intensità e la struttura di diverse tipologie di cellule: è un software disponibile a tutti e capace di gestire centinaia di migliaia di immagini. Il software contiene metodi già sviluppati per lo studio delle cellule, costituiti da algoritmi avanzati per l'analisi di immagini; è sviluppato in ambiente Matlab ed è stato ottimizzato per il formato di immagini più comune, ovvero bidimensionale [16].
CellProfiler raggruppa diversi algoritmi di segmentazione cellulare, come ad esempio algoritmi a soglia, algoritmi di morfologia, algoritmi watershed ecc, ed è l'utente a dover scegliere quello a lui più conveniente in termini di immagini da analizzare [15].
In Figura 14 è riportata la schermata della GUI di CellProfiler, da cui l'utente può effettuare la scelta dell'algoritmo e da cui può caricare le immagini del proprio dataset.

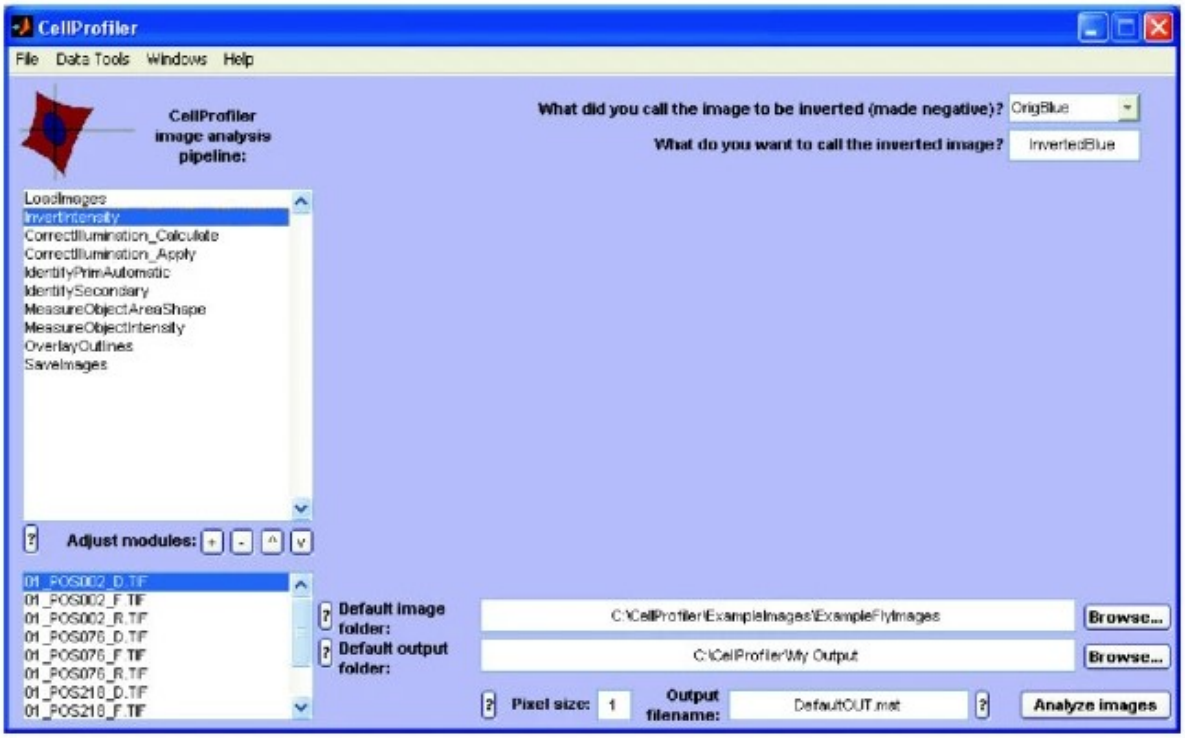


Figura 14: Interfaccia grafica CellProfiler [16]

1.3.2 FogBank

In una singola coltura le cellule individuali possono avere modelli di migrazione diversi, forme differenti, espressioni di proteine differenti e così via. Identificare il fenotipo delle singole cellule può contribuire alla comprensione del fenomeno biologico (ad esempio metastasi di tumori o differenziazione di cellule staminali). Nell'ambito dei metodi per la segmentazione cellulare, se si necessita operare una separazione di oggetti ci si basa sulla loro morfologia e gli algoritmi che ne derivano prendono il nome di "algoritmi watershed". Molti di questi operano effettuando una suddivisione della superficie dell'immagine in regioni differenti, basata sul gradiente dell'intensità dei pixel dei loro contorni. Tuttavia

l'elevato livello di rumore presente nelle immagini biologiche porta ad una problematica definita "over-segmentation". A causa di quest'ultima si presenta una creazione di minimi attorno alle regioni di interesse dell'immagine, generandone numerose di piccola dimensione non aventi significato biologico. Da qui deriva la necessità di un nuovo metodo di segmentazione in grado di separare accuratamente le cellule confluenti in una singola. FogBank è un metodo di segmentazione automatico che separa le cellule quando confluiscono o si toccano l'un l'altra; è una tecnica che può essere applicata alla microscopia a contrasto di fase, alla microscopia in campo chiaro, a quella di fluorescenza e alle immagini binarie. Esso si basa sul principio del watershed morfologico, sfruttando caratteristiche che permettono di aumentare l'accuratezza e di minimizzare l'over-segmentation. L'algoritmo implementato da FogBank riesce a rilevare automaticamente bacini distinti utilizzando le distanze geodetiche (che tengono conto della curvatura) per preservare la forma della singola cellula. Per realizzare tale rilevamento implementa due tecniche:

1. Quantizzazione dell'istogramma con il vincolo sulla dimensione del centro del bacino (seed point);

2. Rilevamento del "seed" dei nucleoli che forniscono informazioni biologiche sulla localizzazione del nucleo e sul raggruppamento delle cellule.

La distanza geodetica viene utilizzata per assegnare i pixel all'oggetto con il seed point più vicino nell'immagine. Questo permette di individuare la forma della cellula a cui tale pixel fa riferimento, similmente alla tecnica manuale. Altri tipi di algoritmi si differenziano da FogBank proprio perché producono i bordi lineari delle cellule.
La segmentazione avviene in cinque step:

1. Separazione dello sfondo dal resto, definendo così la Region of Interest (ROI);

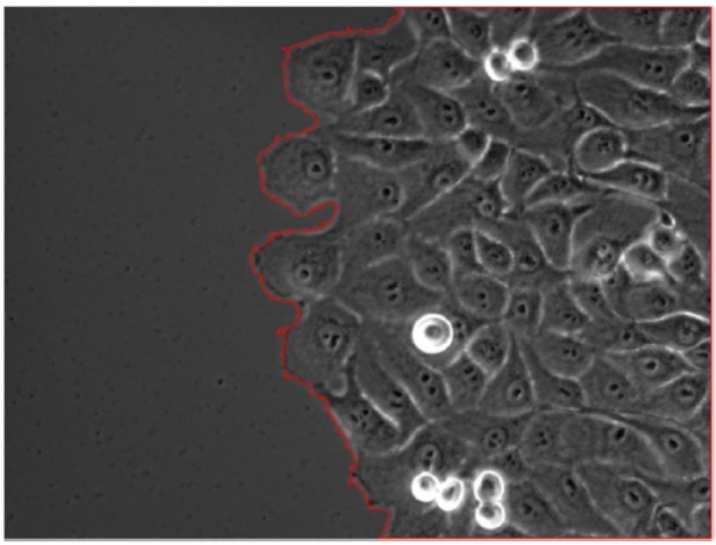


Figura 15: Risultato della segmentazione dei bordi [14]

2. Identificazione di possibili confini cellulari nell'immagine, i quali verranno utilizzati come "barriere" nel calcolo della maschera;

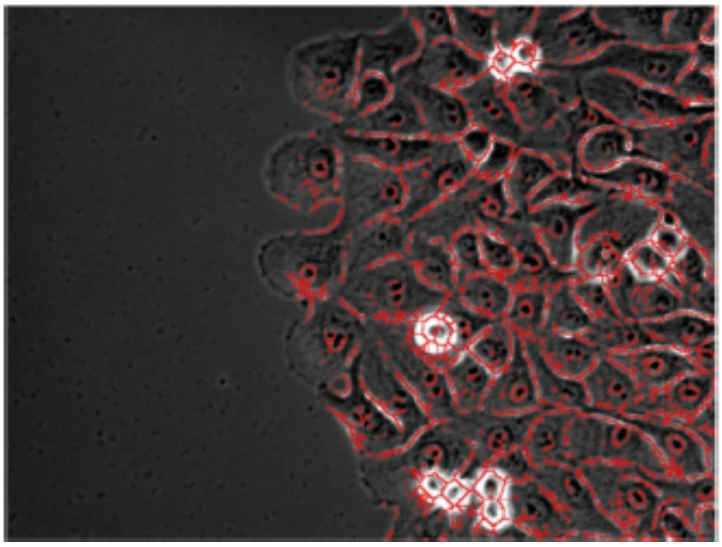


Figura 16: Maschera geodetica che definisce i bordi delle cellule [14]

3. Individuazione dei seed point o dei bacini all'interno delle ROI;

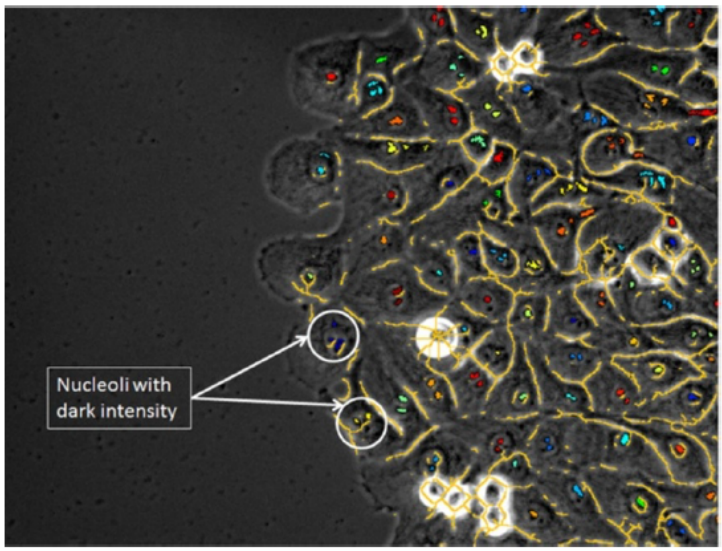


Figura 17: Individuazione dei seed: detection dei nucleoli e dei gruppi attraverso la distanza geodetica [14]

4. Separazione dei bordi delle singole cellule all'interno delle ROI utilizzando i seed point e le maschere applicati sulle immagini in scala di grigio;

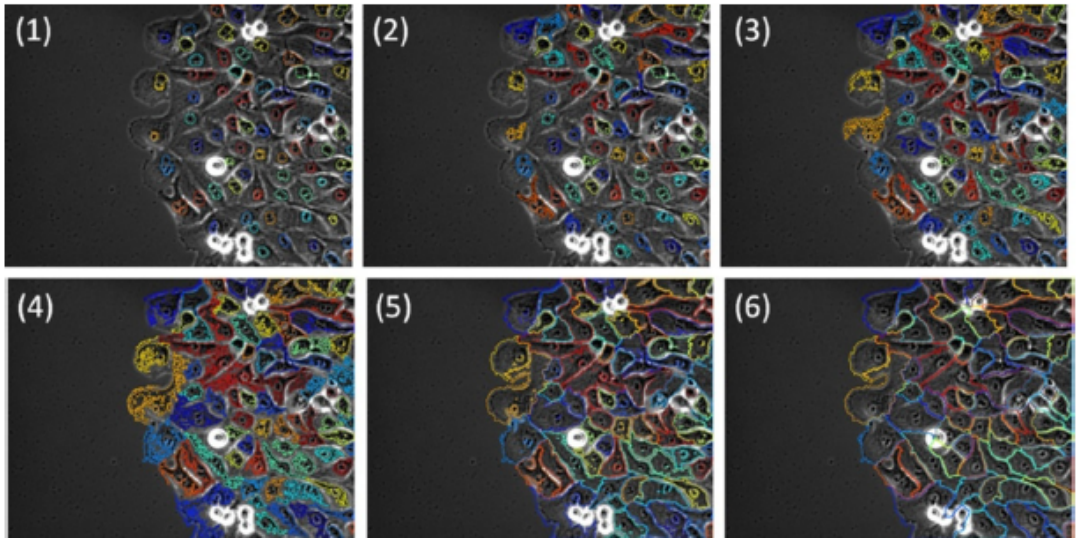
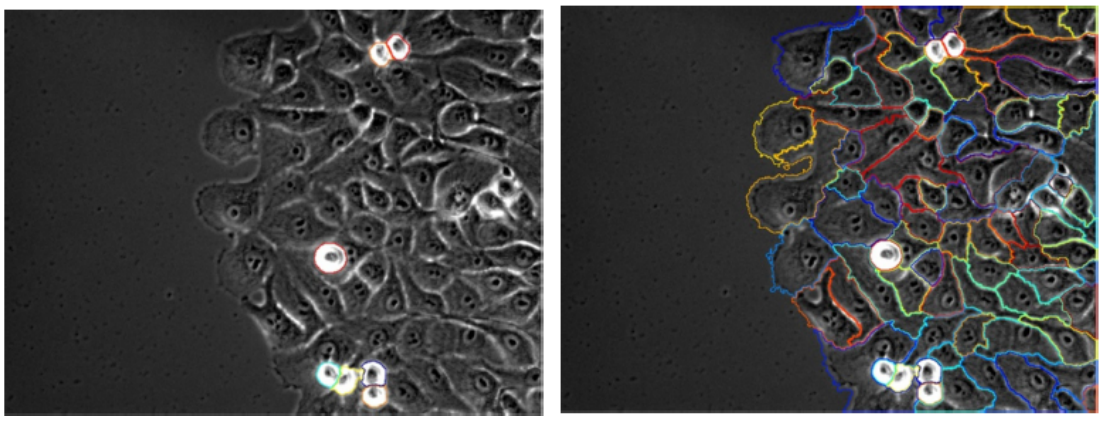


Figura 18: Step di crescita delle regioni ottenute dalla distanza geodetica [14]

5. Individuazione delle cellule mitotiche, aggiunte infine alla maschera.



(a) (b)

Figura 19: (a) Individuazione delle cellule mitotiche, (b) Risultato finale della segmentazione [14]

Il metodo implementato da FogBank per la rilevazione del contorno della cellula ha permesso di ottenere accuratezze più elevate rispetto ad altre tecniche. I risultati, infatti, assomigliano molto a quelli che è possibile ottenere con una segmentazione manuale. Il merito di questa prestazione è da attribuire sia al processo di quantizzazione, che elimina il problema dell'over-segmentation, che al metodo utilizzato per tracciare le cellule individuali all'interno dell'immagine tramite distanza geodetica [14].

1.3.3 ZeroCostDL4Mic

Quando si lavora con intelligenza artificiale e deep learning si ha necessità di creare reti neurali ad hoc per il task richiesto. Queste devono poi essere opportunamente addestrate con dataset adeguati al modello scelto. Se non si avesse a disposizione un buon dataset, o non si avesse la giusta capacità computazionale richiesta, o l'adeguata conoscenza del codice, si può ricorrere a delle soluzioni che mettono a disposizione strumenti utili alla realizzazione del task, anche in assenza dei requisiti sopra elencati.
ZeroCostDL4Mic è una piattaforma che semplifica l'accesso alle reti di DL sfruttando le risorse computazionali gratuite e basate sul cloud di Google Colab. Tale tool permette di addestrare ed utilizzare reti in grado di eseguire task di segmentazione (ad esempio U-Net e StarDist), di detection di oggetti (YOLOv2), di denoising (CARE e Noise2Void), di microscopia ad altissima risoluzione (con Deep-STORM) e di traduzione da immagine ad immagine (utilizzando ad esempio fnet, pix2pix...). ZeroCostDL4Mic applica queste tecnologie allo studio di processi biologici diversi, con lo scopo di analizzare immagini di colture cellulari. Il tool utilizza modelli pre-addestrati per alleviare i requisiti di costo computazionale necessari per il training effettuato da zero, rendendo più semplici, in particolare, i task di segmentazione nucleare e cellulare. Questo metodo comporta, però, il rischio di effettuare predizioni non corrette se utilizzato su dati troppo diversi da quelli originali sfruttati per l'addestramento della rete. Per questo motivo ZeroCostDL4Mic realizza uno strumento che permette agli utenti di addestrare, testare e sperimentare con modelli di DL che realizzano task diversi per l'analisi di immagini. Esso è costituito da una collezione di notebook jupyter che sfruttano le risorse di Google Colab e che non richiedono una conoscenza approfondita di programmazione.
In Figura 20 è mostrata la logica che si trova dietro la realizzazione dei diversi notebook. Nel caso specifico del task di segmentazione cellulare, è evidente che la segmentazione manuale sia un compito complicato, per questo motivo ZeroCostDL4Mic mette a disposizione modelli di reti neurali implementati in Google Colab e disponibili all'utilizzo da parte dell'utente [17].

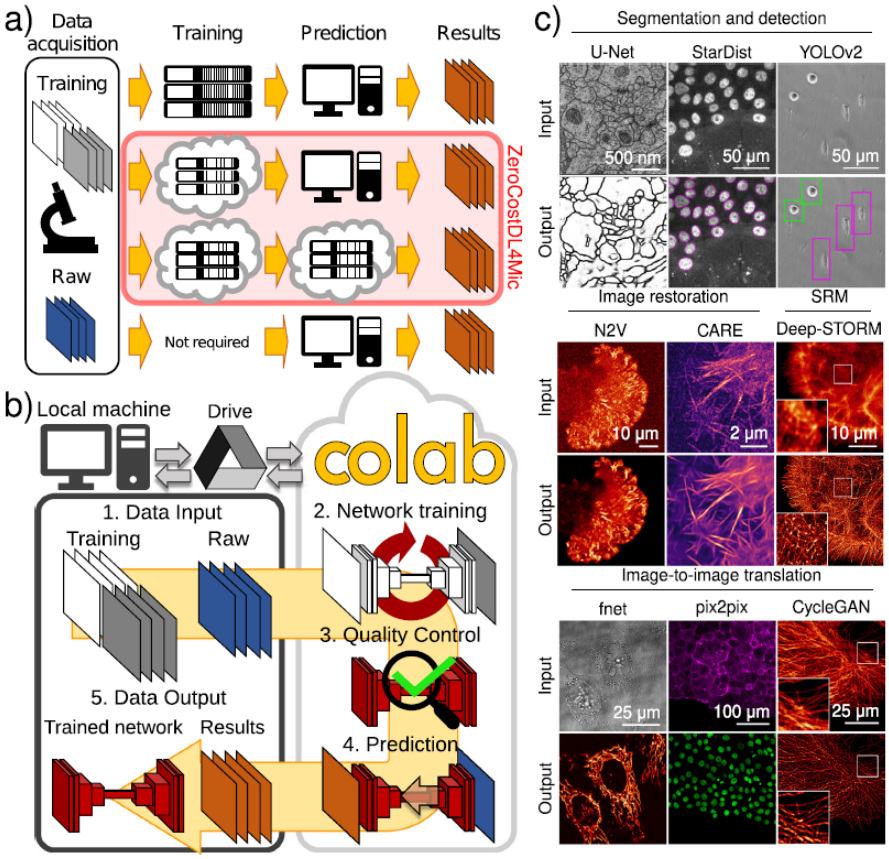


Figura 20: Logica di utilizzo di ZeroCostDL4Mic [17]

Nella tabella 1 sono riassunte le caratteristiche dei tool presenti allo stato dell'arte per la segmentazione cellulare.

Tool	**Prima Versione**	**Linguaggio**	**Architettura**	**Tipologia di metodo**	**Modello pre-trainato**	**Necessità di training**	**Funzione**
CellProfiler	2006	Matlab	NA	Computer Vision	NA	x	Conta di cellule e misura dimensioni
AI FogBank	2014	Matlab	NA	Computer Vision	NA	x	Individuazione di bacini cellulari
ZeroCostDL4Mic	2021	Python	U-Net	Deep Learning	Sì	v	Segmentazione cellulare

Tabella 1: Confronto tra gli strumenti per la realizzazione del task di segmentazione cellulare

2 Materiali e Metodi

2.1 Immagini di colture cellulari

Le A549 e PC9 sono due linee cellulari di Non-Small Cell Lung Cancer (NSCLC, carcinoma polmonare), in particolare le seconde sono caratterizzate dalla mutazione del gene Epidermal Growth Factor Receptor (EGFR). La sua mutazione fa sì che negli anni siano stati sviluppati farmaci inibitori chiamati EGFR-TKI diretti contro di essa, in grado di arrestare la crescita tumorale. Con il tempo però le cellule tumorali NSCLC sviluppano una resistenza contro questi farmaci, tale da indurre la trasformazione da cellule con fenotipo epiteliale (cellule adese tra di loro e alla piastra su cui vengono fatte crescere) a cellule con fenotipo mesenchimale (che tendono a staccarsi tra loro e quindi a perdere contatti le une con le altre). Questo comportamento, conosciuto con il nome di transizione epitelio-mesenchimale (EMT) si traduce "in vivo" nella possibilità di un tumore solido di rilasciare parte di esso (ovvero cellule tumorali) in circolo nell'organismo, dando luogo alla formazione di metastasi.
Per studiare questo fenomeno e replicare questa trasformazione "in vitro", nella prima parte dell'esperimento le linee cellulari di NSCLC A549 e PC9 sono state trattate con una citochina, il TGF-β in grado di indurre "in vitro" la trasformazione epitelio-mesenchimale. Le cellule sono state trattate a diversi tempi con la stessa concentrazione di TGF-β aggiunto al terreno di coltura (24h e 96h) per valutare l'evolversi della trasformazione nelle primissime fasi. Per accertare l'avvenuta trasformazione sono state utilizzate diverse indagini molecolari volte a verificare l'espressione di marcatori specifici sia del fenotipo epiteliale, come E-caderina, che di quello mesenchimale, per il quale è stata testata la Vimentina. In entrambi i casi si tratta di proteine che vengono più espresse rispettivamente dal fenotipo epiteliale e da quello mesenchimale.
Al fine di riconoscere l'evolversi di questi fenotipi da un punto di vista morfologico è stata impiegata la tecnica di immunofluorescenza o immunocitochimica (IF o IC) così da poter sottoporre le immagini acquisite ad un algoritmo di AI che sia in grado di riconoscere i due diversi fenotipi a seconda della morfologia cellulare riscontrata. L'immunofluorescenza è una tecnica che si basa sulla capacità di specifici anticorpi (Ab) di legarsi a proteine presenti nelle cellule, sulla superficie cellulare come la E-caderina, o all'interno dello spazio citoplasmatico come la Vimentina. Questi Ab, definiti primari, possono essere riconosciuti da Ab secondari marcati con fluoroforo e capaci di emettere fluorescenza quando irraggiati da una sorgente laser. In tal modo è possibile rilevare la presenza e la quantità di proteina espressa, oltre che la localizzazione della stessa a livello cellulare.
L'obiettivo della raccolta di immagini cellulari consiste quindi nel sottoporre all'algoritmo le immagini acquisite sia in campo chiaro che in fluorescenza nei due tempi (24h, 96h) e nei due trattamenti (senza e con TGF-β). In questo modo si vuole correlare la morfologia delle cellule con la fluorescenza emessa: tanto più la cellula conserva la sua forma ton-

deggiante/simmetrica (a contatto con altre cellule) tanto più ci si aspetta una maggiore espressione della E-caderina (fluorescenza rossa). Al contrario se la cellula presenta una forma allungata/affusolata e quindi più asimmetrica, perdendo il contatto con le cellule adiacenti, ci si aspetta una maggiore intensità nell'espressione della Vimentina (fluorescenza verde).
Le immagini in fluorescenza sono state acquisite ad un ingrandimento 20x con microscopio invertito Eclipse Ti2-E dotato di sistema confocale AX (Nikon, Japan). Le immagini in campo chiaro sono state invece acquisite con un microscopio ottico invertito e gli ingrandimenti utilizzati sono 20x e 40x.
Le immagini fornite dal laboratorio di Medicina con questa tecnica sono:

- 57 immagini di PC9, 20x, 24h;
- 21 immagini di PC9, 20x, 48h;
- 52 immagini di PC9, 20x, 72h;
- 35 immagini di PC9, 40x, 72h;
- 55 immagini di PC9, 20x, 96h;
- 7 immagini di PC9, 20x, 120h;
- 56 immagini di A549, 20x, 24h;
- 30 immagini di A549, 20x, 48h;
- 57 immagini di A549, 20x, 72h;
- 27 immagini di A549, 20x, 96h;
- 2 immagini con relativa fluorescenza (rosso, verde, blu).

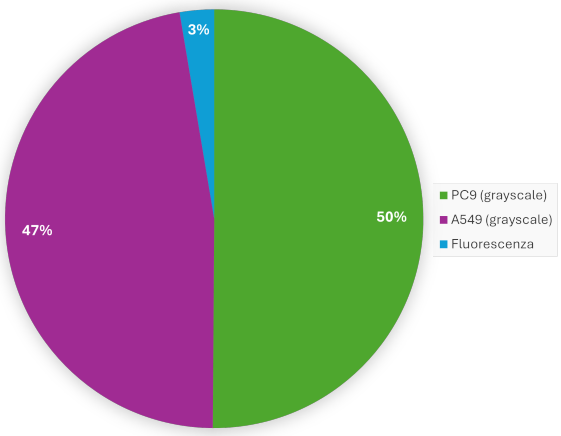


Figura 21: Rappresentazione della distribuzione del dataset di testing

Di queste sono state fornite le relative annotazioni utili all'addestramento della rete neurale di: 3 immagini PC9 (20x, 48h) mostrate in Figure 22-24, 5 immagini A549 (20x, 48h), mostrate in Figure 25-29 e 1 immagine a fluorescenza, presentata in Figure 30-34 con le rispettive fluorescenze.

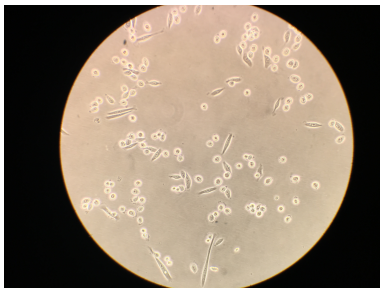


Figura 22: Immagine PC9 P1 20X 48h_1

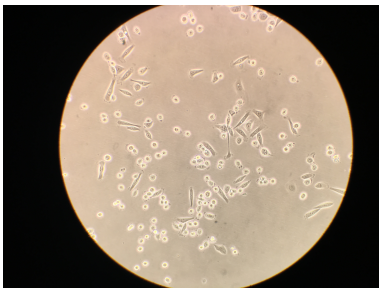


Figura 23: Immagine PC9 P1 20X 48h_4

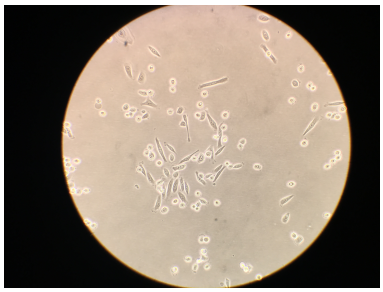


Figura 24: Immagine PC9 P1 20X 48h_7

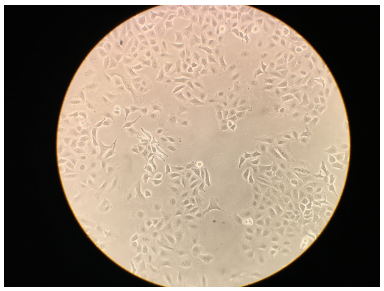


Figura 25: Immagine IMG_4105

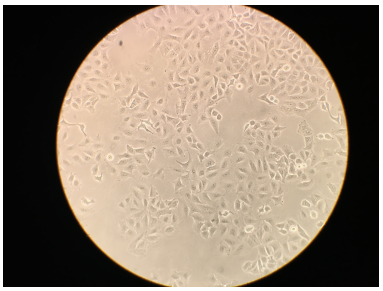


Figura 26: Immagine IMG_4106

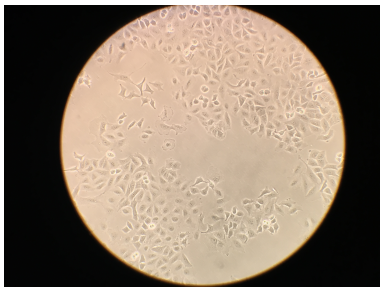


Figura 27: Immagine IMG_4111

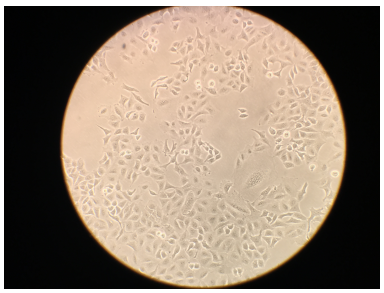


Figura 28: Immagine IMG_4113

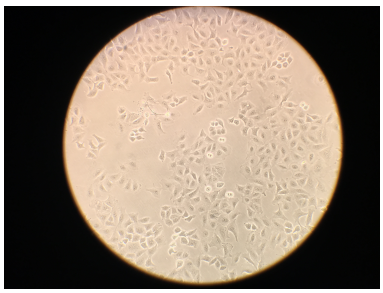


Figura 29: Immagine IMG_4114

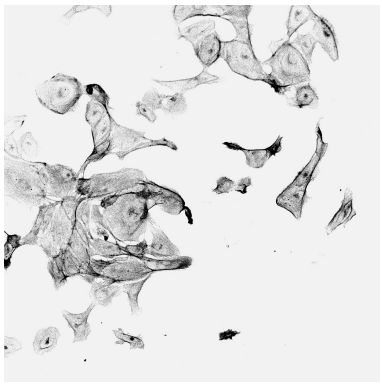


Figura 30: Immagine in campo chiaro

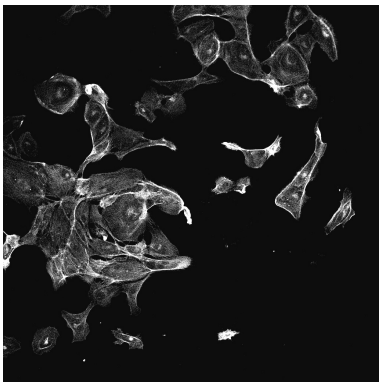


Figura 31: Immagine in campo scuro

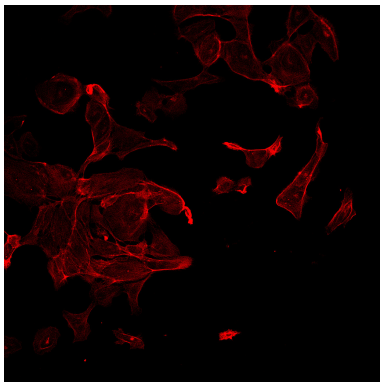


Figura 32: Fluorescenza rossa

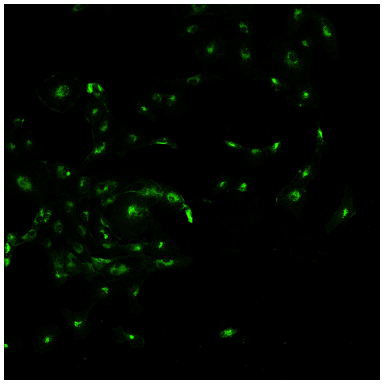


Figura 33: Fluorescenza verde

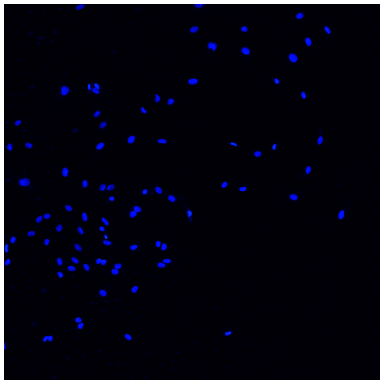


Figura 34: Fluorescenza blu

2.2 Zero Cost Deep Learning for Microscopy

Il tool ZeroCosDL4Mic, già introdotto nel capitolo precedente, è stato utile a testare e studiare diverse reti neurali dedicate all'analisi di immagini di differenti configurazioni cellulari ottenute da microscopio. ZeroCost mette a disposizione notebook preconfigurati su Google Colab per l'addestramento e l'applicazione di modelli di DL nel contesto di microscopia. Questo risulta utile nel caso in esame soprattutto per la possibilità di utilizzare reti neurali pre-addestrate ed eventualmente ri-addestrabili su dati forniti dall'utente. I principali vantaggi sono: l'accessibilità a risorse computazionali avanzate a basso costo e la facilità d'uso per gli utenti anche con bassa esperienza di programmazione.
ZeroCostDL4Mic (ottenibile in [18]) prevede l'upload dei dati di train all'interno del notebook di Google Colab tramite l'utilizzo di Google Drive. La predizione (*inference*) su immagini di test può essere eseguita successivamente sempre attraverso lo stesso notebook, senza la necessità di utilizzare hardware locali [19].
La logica operativa è presentata in Figura 35.

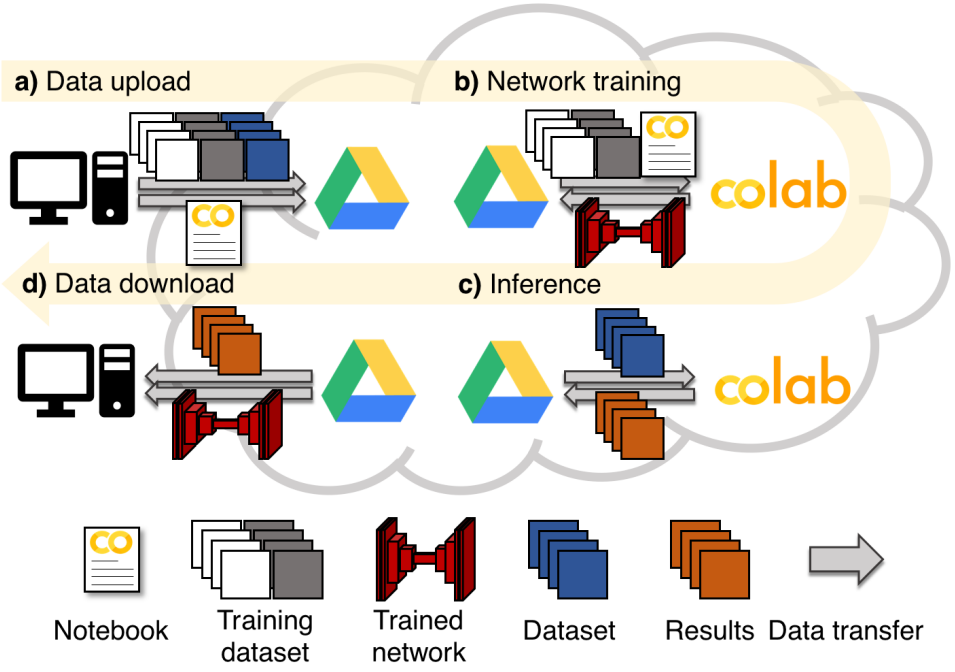


Figura 35: Logica di ZeroCostDL4Mic [19]

I notebook messi a disposizione dal tool, implementano reti note pre-esistenti, riuscendo così a coprire un grande range di compiti utili all'analisi di immagini cellulari (segmentation, denoising, restoration, label-free prediction). Le reti implementate possono essere:

- Fully supported: considerate mature e correttamente testate;
- Under beta-testing: primo prototipo della rete che potrebbe non essere ancora del tutto stabile.

Per tutti i tipi di rete viene messo a disposizione un link di Google Colab per permettere a chiunque di utilizzare il notebook, insieme ad un link relativo ai dataset di training e testing utili allo studio del modello scelto [19].
Grazie all'utilizzo di ZeroCostDL4Mic è stato possibile analizzare diverse reti disponibili per il task di segmentazione cellulare. L'obiettivo di questo studio consiste nell'identificare il modello in grado di produrre una maschera il più attendibile possibile rispetto a quelle generate manualmente nel laboratorio di Medicina. Per questo motivo, le reti neurali sono state testate eseguendo il training sui dataset messi a disposizione da ZeroCost. Successivamente, i risultati ottenuti durante la fase di test sul modello ottenuto (predizione sul dataset fornito dallo staff medico) sono stati confrontati con le etichettature eseguite in maniera manuale all'interno del laboratorio.

2.3 FlowerAI

Come già introdotto, il Federated Learning è una tecnica utile a dispositivi diversi al fine di collaborare nell'addestramento di un modello di predizione, mantenendo però i dati localmente su ciascun dispositivo. Nella pratica il FL risulta difficile da implementare, sia in termini di scalabilità che di eterogeneità del sistema. Per affrontare queste sfide è stato utilizzato Flower, un framework completo per il FL disponibile in [20], che offre nuove strutture avanzate, utili ad effettuare esperimenti su larga scala, oltre che a considerare molteplici scenari eterogenei in termini di device coinvolti. Flower permette di ottenere un sistema che implementa le componenti di FL ed è un linguaggio stabile che fa ML indipendentemente dal framework che si utilizza. Realizza un livello di astrazione elevato che permette ai ricercatori di sperimentare e di sviluppare nuove idee su un'infrastruttura affidabile. Con Flower è possibile eseguire una transizione degli algoritmi di addestramento di ML ad una logica di FL, al fine di valutarne le proprietà di convergenza e il tempo necessario al training in una logica distribuita. Flower, dunque, mette a disposizione tool integrati utili a simulare condizioni particolari in ambiente cloud [21].
Nel lavoro proposto, Flower è stato utile nella fase di transizione tra apprendimento centralizzato e apprendimento distribuito, necessario soprattutto al mantenimento della privacy dei singoli client. In ambito medico ed ospedaliero, infatti, la privacy dei pazienti è il focus principale, rendendone spesso complicata la diffusione dei dati all'esterno dell'ospedale stesso. Oltre a questo si è richiesto il passaggio al learning federato anche a causa della limitata disponibilità di immagini utili e necessarie al riaddestramento delle reti neurali sperimentate tramite l'utilizzo del tool ZeroCostDL4Mic. Una volta individuato il miglior modello di ML per le immagini fornite dal laboratorio di Medicina, infatti, esso è stato implementato con logica distribuita, sfruttando i tool messi a disposizione da Flower. Il tutto è stato svolto sempre in ambiente Google Colab per sfruttare le capacità computazionali e l'efficienza del linguaggio Python. Per poter realizzare una simulazione di Federated Learning è necessario configurare Flower nel modo più adeguato al task che si vuole compiere.
L'architettura del framework Flower si basa su due livelli: una logica globale, tramite l'astrazione di una "Strategy", e una logica locale, tramite l'astrazione a livello client. Poiché il server non è a conoscenza della natura dei client ad esso collegati, questo permette a quest'ultimi di effettuare un addestramento del modello indipendentemente dalla piattaforma che utilizzano e dall'implementazione seguita [21].
I principali parametri da configurare per adattare Flower al caso d'uso sono:

1. Funzioni di get e set dei parametri del modello: necessarie in fase di invio e restituzione dei pesi aggiornati da server a client e viceversa;

2. Funzione di addestramento: adattata al modello di ML utilizzato ed eseguita da ogni client in fase di training locale;

3. Flower Client: contiene i metodi necessari a realizzare le operazioni locali, come addestramento ed invio dei pesi aggiornati;

4. Strategia: specifica il numero di client da selezionare per ogni round di addestramento;

5. Simulazione: comprende la configurazione dell'istanza dei client, il numero di client, il numero di round, la strategia e le risorse dei client.

Le impostazioni specifiche per la simulazione includono:

- Il numero di epoche per cui ogni client deve eseguire l'addestramento sul proprio dataset;

- Il numero di round per cui la simulazione deve proseguire;

- Il numero di client che, rispetto a quelli totali, devono essere coinvolti in ogni round;

- Il tipo di aggregazione da eseguire per l'aggiornamento dei pesi del modello globale.

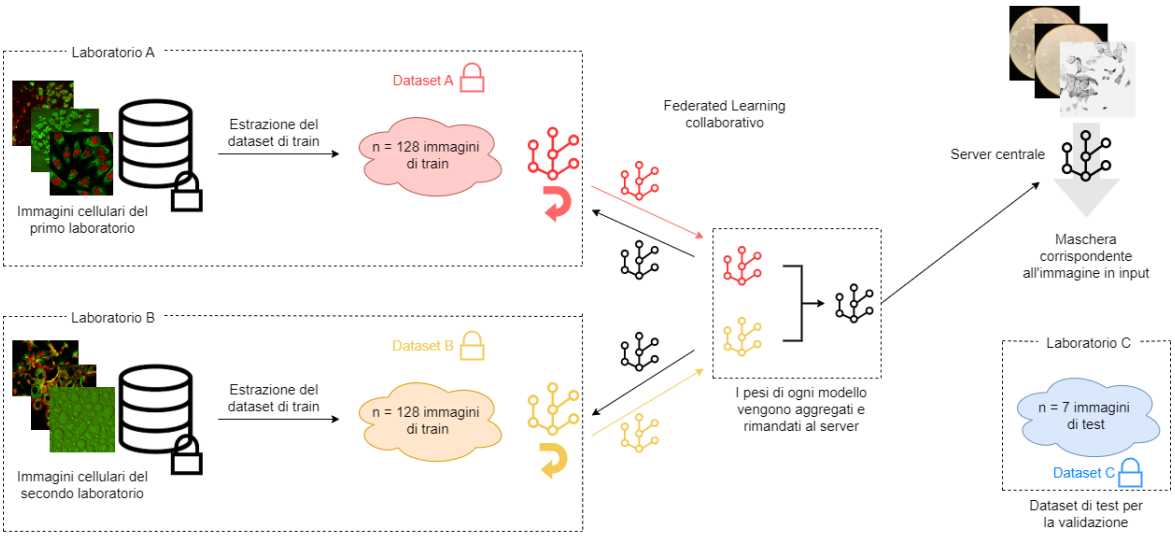


Figura 36: Federated Learning applicato al task di segmentazione cellulare

In Figura 36 è rappresentata la logica di applicazione del Federated Learning al task di segmentazione cellulare. Si nota che la privacy viene rispettata dalla possibilità di mantenere privato il dataset del singolo laboratorio. I client, infatti, devono condividere solamente i pesi aggiornati del modello a seguito dell'addestramento locale, i quali verranno aggregati dal server allo scopo di ottenere il modello globale.

3 Benchmark di Modelli Centralizzati per la Segmentazione Cellulare

Nella seguente sezione si analizzeranno le principali famiglie di modelli disponibili in ZeroCost per la segmentazione cellulare, per poi focalizzarsi sulle performance della famiglia dei modelli di Cellpose, le quali presentano le migliori prestazioni in risposta alle immagini di test a disposizione.
Ognuno dei notebook implementati in ZeroCostDL4Mic presenta una struttura analoga: al loro interno è possibile trovare una suddivisione in sezioni che aiuta l'utente ad orientarsi tra le diverse operazioni che può compiere.
Le sezioni sono:

1. Installazione del modello di rete e delle dipendenze;
2. Inizializzazione della sessione di Colab: controllo della GPU e montaggio di Google Drive;
3. Selezione dei parametri e del percorso file: scelta dei principali parametri utili all'addestramento, selezione della data augmentation e caricamento dei pesi di una rete pre-trainata (se scelta);
4. Addestramento della rete;
5. Valutazione del modello: ispezione della funzione di loss e delle metriche di qualità;
6. Utilizzo del modello addestrato: predizione su dati sconosciuti alla rete.

A seconda del compito che l'utente vuole compiere attraverso l'utilizzo del notebook può seguire diversi passi:

- Se si desidera addestrare la rete da zero utilizzando un dataset personalizzato è necessario eseguire le sezioni 1-4. Queste devono poi essere seguite dalla sezione 5, al fine di ottenere informazioni sulla qualità del modello, e dalla sezione 6 per eseguire un test su dati diversi;
- Se si desidera semplicemente valutare un modello precedentemente generato e salvato in Google Drive, basta eseguire le sezioni 1-2 per il setup del notebook e la sezione 5 per l'analisi di qualità;
- Se, invece, si desidera eseguire predizioni su delle immagini utilizzando un modello pre-trainato, basterà eseguire le sezioni 1-2 e 6.

Nel lavoro presentato si analizzano i modelli di ML dedicati alla segmentazione cellulare. Essi vengono addestrati sui dataset messi a disposizione dal tool e successivamente testati sulle immagini fornite dal laboratorio di Medicina, allo scopo di individuare il modello migliore per il caso di studio.

3.1 Confronto tra Reti Neurali

Per effettuare un benchmarking al fine di individuare il miglior modello di ML per le immagini cellulari a disposizione, si fa affidamento all'annotazione manuale effettuata all'interno del laboratorio. Tale annotazione consiste di una maschera ottenuta a partire dall'immagine originale che individua le singole cellule nell'immagine, effettuandone anche una conta (Figura 37). Inoltre viene fornito un file CSV, mostrato in Figura 38, che descrive ognuna di queste regioni individuate all'interno della maschera. Ciascuna di esse è una cellula di cui possono essere descritte le caratteristiche principali espresse in pixel: area, perimetro, lunghezza asse maggiore, lunghezza asse minore, circonferenza ed altre non utili allo scopo qui presentato.

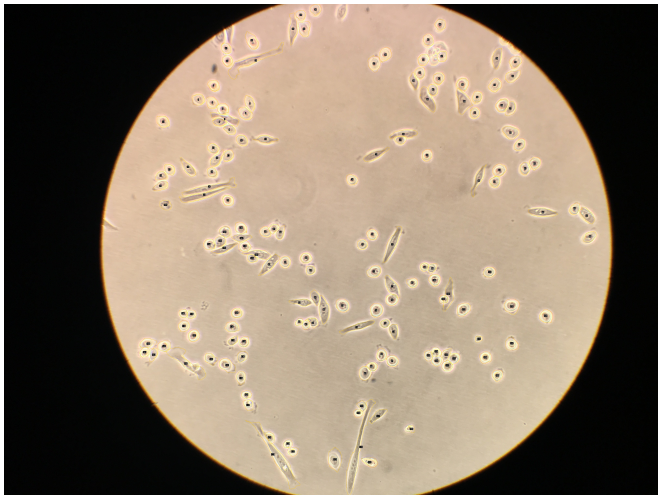


Figura 37: Maschera fornita dai laboratori della facoltà di Medicina per un'immagine PC9

	Area	Perim.	Major	Minor	Angle	Circ.	Feret	Median	FeretX	FeretY	FeretAngle	MinFeret	AR	Round	Solidity
1	4087	242,09	78,37	66,4	178,54	0,88	85,09	207	1159	568	156,45	64	1,18	0,85	0,96
2	3696	230,83	76,28	61,69	139,79	0,87	80,72	218	1269	470	131,99	63,9	1,24	0,81	0,94
3	4223	249,67	80,34	66,93	137,37	0,85	87,68	216	1247	578	135	70	1,2	0,83	0,95
4	3842	237,33	72,95	67,06	164,85	0,86	83,35	222	1313	624	149,74	68	1,09	0,92	0,95
5	4230	273,43	103,67	51,95	111,89	0,71	111,43	212	1493	562	111,04	56,69	2	0,5	0,92
6	4826	271,73	99,92	61,5	143,96	0,82	102,22	212	1447	646	149,42	64,25	1,62	0,62	0,94
7	5760	459,97	174,79	41,96	159,46	0,34	186,46	187	1271	668	157,29	54,67	4,17	0,24	0,68
8	2767	253,13	81,72	43,11	172,51	0,54	88,29	188	1285	718	166,91	46	1,9	0,53	0,92
9	2770	232,2	74,12	47,58	151,2	0,65	83,26	202	1351	762	155,9	52	1,56	0,64	0,93
10	3570	224,06	76,58	59,35	138,65	0,89	82,1	202	1431	820	145,92	62,43	1,29	0,78	0,96
11	4036	347,52	122,76	41,86	177,28	0,42	156,05	185	1535	834	178,53	47,17	2,93	0,34	0,83
12	3359	225,32	78,2	54,69	57,23	0,83	84,38	207	1359	970	58,57	59,4	1,43	0,7	0,95
13	4217	303,24	116,43	46,11	51,56	0,58	124,02	200	1275	996	52,21	54,89	2,52	0,4	0,89
14	2503	186,21	60,28	52,87	146,22	0,91	65,51	211	1253	1018	148,74	54	1,14	0,88	0,96
15	3168	214,97	72,59	55,57	157,13	0,86	75,29	210	1255	874	163,01	58,65	1,31	0,77	0,94
16	4685	334,28	128,59	46,39	128,41	0,53	141,53	196	1087	946	132,71	52,93	2,77	0,36	0,87
17	4445	254,02	86,72	65,26	141,4	0,87	94,87	215	943	694	145,3	68,83	1,33	0,75	0,96
18	3264	218,29	75,52	55,03	130,63	0,86	80,92	217	993	990	129,99	59,68	1,37	0,73	0,95
19	2542	195,5	67,07	48,25	21,71	0,84	72,11	201	935	1082	33,69	50	1,39	0,72	0,95
20	3750	250,52	98,49	48,48	20,37	0,75	101,13	208	919	1144	24,54	52,02	2,03	0,49	0,95
21	10006	793,94	342,82	37,16	12,69	0,2	342,94	181	1097	1170	9,74	62,98	9,22	0,11	0,61
22	7939	691,46	282,13	35,83	13,62	0,21	317,35	173	1079	1188	10,53	46,46	7,87	0,13	0,71
23	3483	221,29	71,67	61,87	140,24	0,89	77,67	220	1351	1186	145,49	64	1,16	0,86	0,95
24	3623	228,13	74,55	61,87	154,76	0,87	78,82	215	2111	1058	144,29	65,19	1,2	0,83	0,95
25	5041	392.28	163.5	39.26	23.85	0.41	175.45	165	2215	956	24.23	48.3	4.16	0.24	0.83

Figura 38: CSV rappresentante le caratteristiche della maschera

L'obiettivo consiste nel generare una maschera ed un CSV analoghi attraverso l'utilizzo dei vari modelli di ML per le immagini messe a disposizione dai laboratori della facoltà di Medicina, effettuare un confronto con il materiale sopra presentato, il quale costituisce il "ground truth", ed infine individuare il miglior modello per realizzare la segmentazione cellulare.
I parametri utili al confronto per valutare quanto il modello è in grado di avvicinarsi al ground truth sono:

- Il numero di cellule individuate;
- La percentuale di cellule oblunghe;
- La percentuale di cellule circolari.

Per identificare la forma di una cellula si utilizza un parametro definito Spindle Index.
Lo spindle index S_i è una misura di morfologia della cellula che si ricava dal rapporto tra la massima lunghezza e la massima larghezza del miglior contorno che riesce ad approssimare il perimetro della cellula [23].
In questo ambito, considerando che sia un'ellisse il contorno che approssima una cellula, lo spindle index si ottiene come:

$$S_i = \frac{Lunghezza\ asse\ maggiore}{Lunghezza\ asse\ minore}$$

Dal calcolo dell'S_i poi è possibile ottenere un parametro che viene definito Spindle Flag S_f. Lo Spindle Flag determina se la cellula ha una forma oblunga o circolare a partire dal valore dello Spindle Index. In particolare:

$$Spindle\ \ Flag: \begin{cases} Se\ \ S_i > 3\ \ cellula\ \ oblunga & \implies S_f = 1 \\ Se\ \ S_i < 3\ \ cellula\ \ circolare & \implies S_f = 0 \end{cases}$$

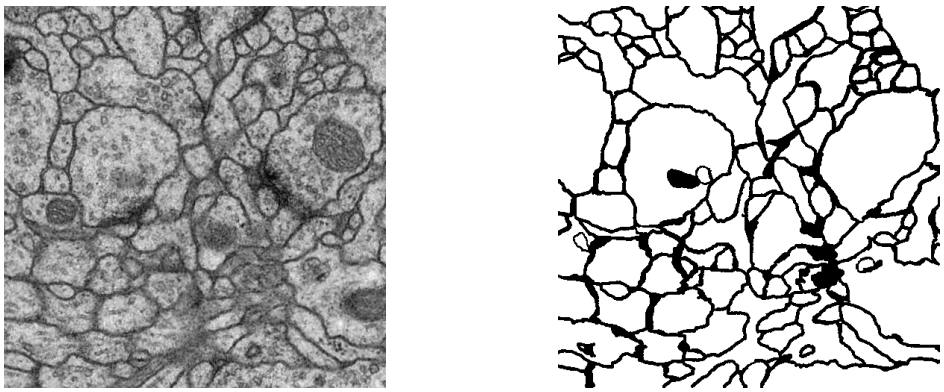
Risulta infine importante sottolineare il fatto che in ingresso ai vari modelli, sia per la fase di training che per la fase di testing, è necessario fornire le immagini in formato TIFF ed in grayscale. Questo comporta dunque la necessità di effettuare un preprocessing delle immagini.

3.1.1 Valutazione delle performance dei vari modelli

Dei diversi modelli di ML messi a disposizione da ZeroCostDL4Mic per effettuare il task di segmentazione cellulare, quelli testati con le immagini a disposizione sono: U-Net (2D, 2D multilabel), StarDist (2D), Cellpose (2D), SplineDist (2D) e EmbedSeg (2D). La valutazione delle performance è stata effettuata addestrando la rete neurale scelta con il dataset fornito dal tool. Questa è stata poi testata sulle immagini PC9 (Figure 22-24) per

confrontare il risultato con le maschere realizzate dal laboratorio di Medicina. Si testa, come prima cosa, l'efficacia della segmentazione del modello in termine di individuazione e separazione degli oggetti all'interno dell'immagine.

U-Net 2D La rete U-Net è una soluzione di deep learning per task di quantificazione, come il rilevamento di cellule e la misura di forma e dimensioni su immagini biomedicali. La sua struttura di tipo encoder-decoder la rende ideale al task di segmentazione che permette di individuare le singole cellule nell'immagine [24].
Il notebook realizzato in ZeroCost implementa una U-Net pre-trainata su un set di immagini appartenenti al dataset messo a disposizione da ImageJ. Un esempio di coppia immagine-maschera è mostrato in Figura 39.



(a) (b)

Figura 39: (a) Immagine e (b) Maschera del dataset di training per il modello U-Net 2D

La rete è stata addestrata su queste immagini per 100 epoche, prevedendo data augmentation e partendo dai pesi del modello pre-addestrato presente nel notebook. Il modello ottenuto è stato utilizzato per il testing con le immagini PC9.

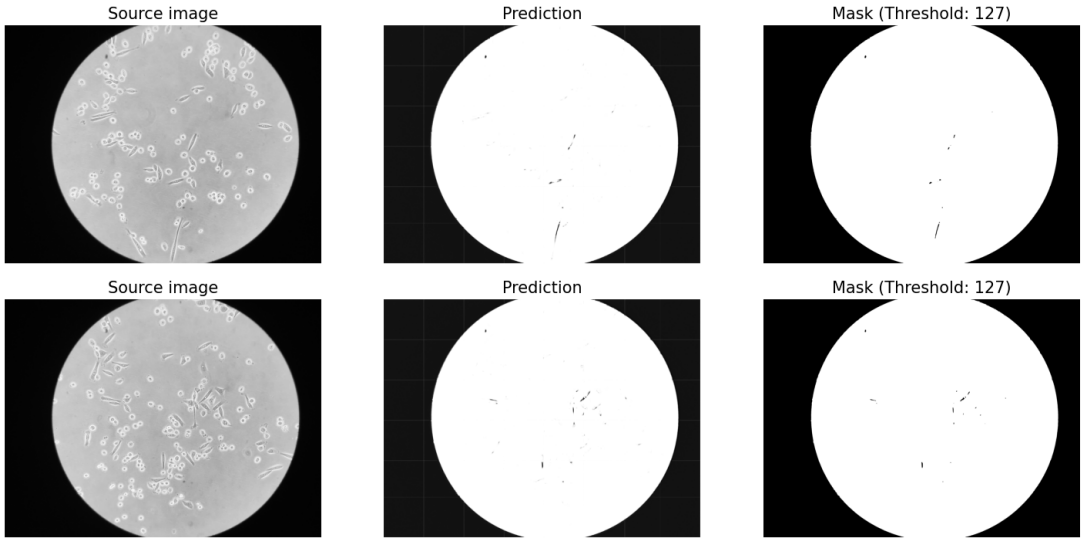


Figura 40: Risultato del testing sul modello U-Net 2D

I risultati che si ottengono su due delle tre immagini a disposizione dal laboratorio, sono mostrati in Figura 40. Si osserva che il risultato è molto lontano da quello che si vorrebbe in relazione alla maschera costruita manualmente. Per cui la U-Net 2D non riesce ad effettuare una segmentazione adeguata dell'immagine fornita, rendendola inutilizzabile ai fini della seguente analisi.

U-Net 2D Multilabel Questo modello di DL è una variante del precedente. Mentre la U-Net esegue una classificazione di ogni pixel in una singola classe (segmentazione binaria o multiclasse), la U-Net Multilabel permette ad un singolo pixel di appartenere a più classi contemporaneamente.
Anche in questo caso il modello è stato prima addestrato sul dataset fornito da Zero-Cost (di cui un esempio di immagine-maschera è mostrato in Figura 41), scegliendo 10 epoche, a causa di restrizioni dovute alla capacità computazionale, e poi testato sulle immagini ottenute dal laboratorio di Medicina.

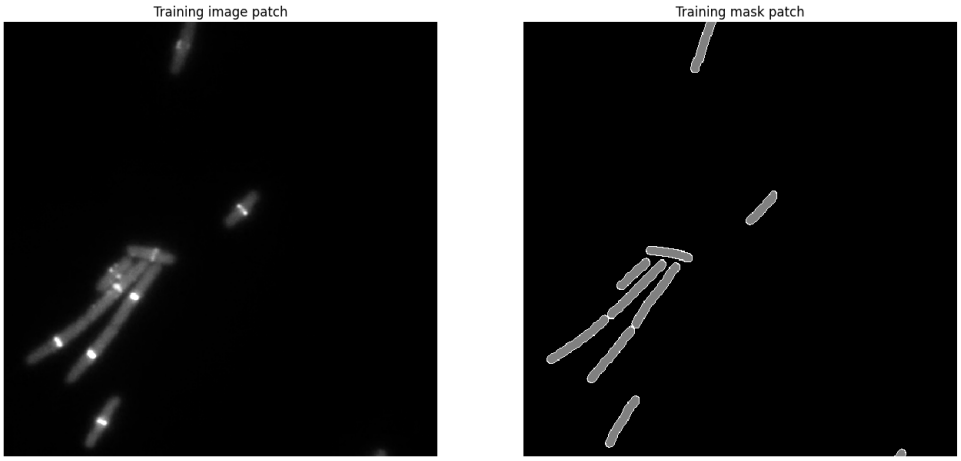


Figura 41: Immagine e maschera del dataset di training per la U-Net 2D Multilabel

Il risultato restituito dalla fase di test sembra visivamente corretto se confrontato con l'immagine originale corrispondente, come può essere osservato in Figura 42.

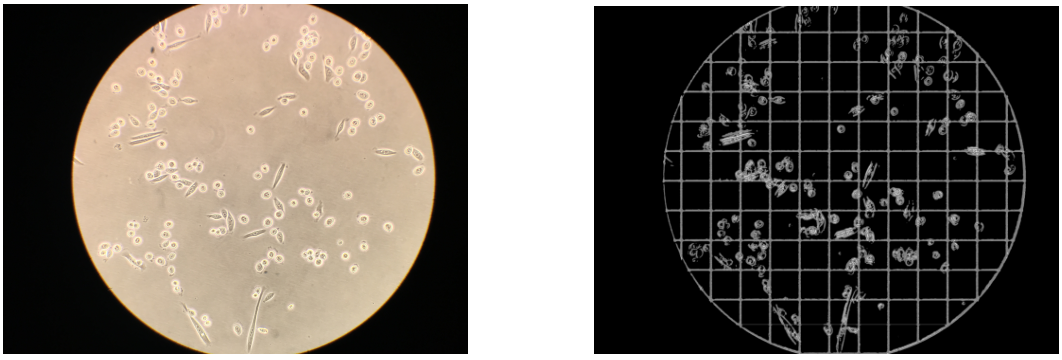


Figura 42: Risultato del testing sul modello U-Net 2D Multilabel

In realtà, analizzando più nel dettaglio le regioni individuate all'interno della ma-

schera si vede come siano state segmentate solo due regioni: sfondo e non sfondo. Questo non si adatta dunque al task di segmentazione che si vuole realizzare sulle immagini a disposizione.
Un tentativo per "forzare" la maschera a segmentare più regioni al suo interno consiste nell'applicare la funzione *measure.label*() del pacchetto skimage, la quale serve ad etichettare le regioni connesse in un'immagine binaria. Da quest'ultima la funzione identifica gruppi di pixel connessi (con valore "1" o True); ogni gruppo di pixel connessi viene etichettato con un numero intero unico, mentre i pixel che non fanno parte di alcuna regione (con valore "0" o False) rimangono a 0 [25].
Applicando *measure.label*() alla maschera restituita da U-Net 2D Multilabel, le regioni individuate passano da un totale di 2 ad un totale di 18, risultato non ancora accettabile in confronto alle 226 regioni segmentate manualmente dallo staff medico.

StarDist 2D In situazioni in cui nell'immagine le cellule sono molto raggruppate potrebbero esserci degli errori di segmentazione, dovuti a cellule confinanti erroneamente unite o ad una scarsa approssimazione che porta all'esclusione di una di esse. StarDist cerca di superare queste problematiche localizzando il nucleo della cellula attraverso poligoni convessi a stella, una migliore riproduzione della forma rispetto alle bounding box. Il modello è realizzato addestrando una rete neurale convoluzionale che esegue predizioni per ogni pixel appartenente ad un poligono corrispondente all'istanza della cellula presente in quella posizione. Questo metodo presenta buone performance in immagini dove i nuclei sono molto raggruppati. StarDist utilizza una rete neurale "lightweight" basata sulla U-Net [26].

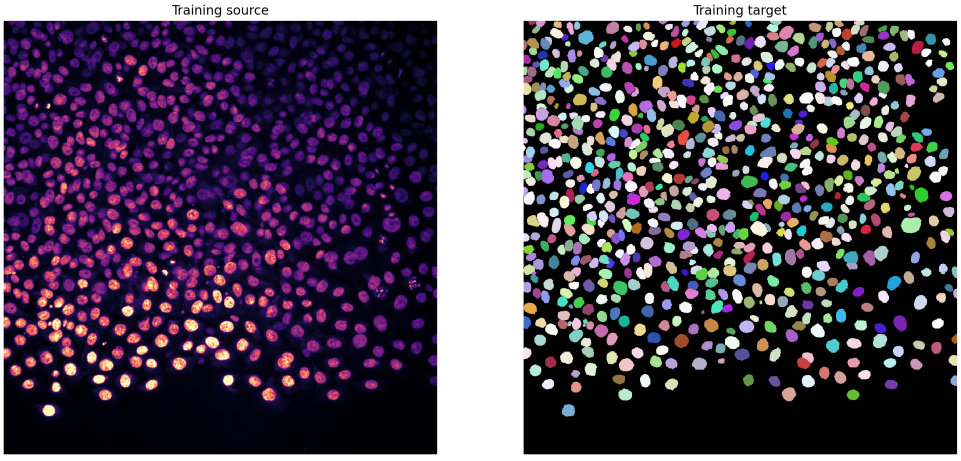


Figura 43: Immagine e maschera del dataset di training per StarDist

Utilizzando il notebook di ZeroCostDL4Mic il modello StarDist viene addestrato sul dataset fornito (un esempio è presentato in Figura 43) per un totale di 50 epoche, con data augmentation e sfruttando i pesi del modello pre-trainato.
In questo caso il testing effettuato sulle immagini PC9 evidenzia che StarDist non

risulta adatta al task richiesto, poiché nella maschera restituita non viene individuata nessuna regione. Questo dimostra che la segmentazione con il modello analizzato non va a buon fine.

SplineDist 2D SplineDist è una rete neurale convoluzionale per bioimmagini che estende il metodo implementato da StarDist. Utilizza una rappresentazione più generale e flessibile poiché modella gli oggetti come curve parametriche planari. Questo gli permette di segmentare anche elementi che non hanno forma a stella convessa, a differenza del modello da cui deriva [27].
Per la similarità con il modello precedentemente analizzato, il training viene effettuato con lo stesso dataset e con le stesse impostazioni. In fase di testing, come deducibile, i risultati sono analoghi a quelli osservati per StarDist, rendendo quindi SplineDist inadatto al task che si sta cercando di implementare.

EmbedSeg 2D EmbedSeg è un metodo di segmentazione delle istanze basato sull'embedding. In questo modello ogni pixel esegue una predizione del proprio embedding spaziale, ovvero individua la locazione univoca di un solo altro pixel che ha il ruolo di rappresentare l'intera istanza dell'oggetto di cui fa parte. Una particolarità di EmbedSeg è la piccola impronta sulla memoria GPU, il che rende possibile l'addestramento anche su computer personali ed il tool accessibile a chiunque [28].

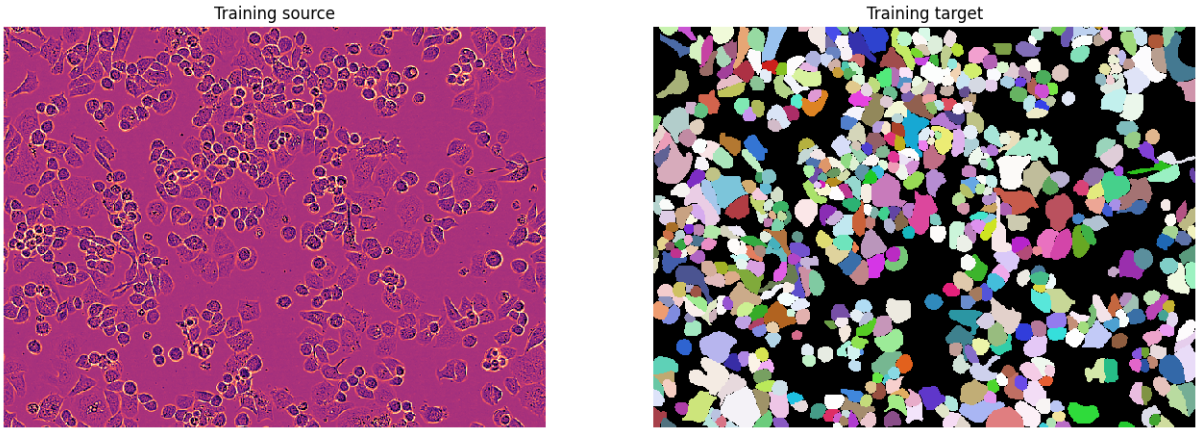


Figura 44: Immagine e maschera del dataset di training per EmbedSeg

Anche con EmbedSeg è stato previsto data augmentation e l'utilizzo di pesi di un modello pre-trainato per addestrarlo sul dataset fornito da ZeroCost (Figura 44) per 50 epoche.
In questo caso, però, non si può procedere alla fase di testing sulle immagini del laboratorio di Medicina a causa di problematiche riscontrate durante l'addestramento del modello. Queste potrebbero essere riconducibili allo stato "under beta-testing" della rete segnalato da ZeroCostDL4Mic, rendendo quindi EmbedSeg inadatto all'analisi qui necessaria.

Cellpose 2D Cellpose è un metodo di segmentazione generalizzato basato sul Deep Learning, in grado si segmentare cellule provenienti da un ampio range di immagini diverse. Non richiede un riaddestramento del modello, né l'aggiustamento dei parametri, poiché la rete è stata già addestrata su un dataset vasto e diversificato. Pertanto, i modelli pre-addestrati di Cellpose possono essere utilizzati direttamente per la predizione ed il testing [29].

Il dataset di training di Cellpose può essere suddiviso in gruppi più piccoli (forniti da da ZeroCost) in base alle necessità dell'utente e del task da implementare. Inoltre Cellpose mette a disposizione diversi modelli pre-addestrati che possono essere utilizzati dall'utente in fase di testing o di re-training.

Allo scopo di confrontare le performance di Cellpose rispetto ai modelli fino a qui presentati il primo tentativo è stato eseguito con il dataset SkBr3 (di cui un esempio immagine-maschera è riportato in Figura 45) che appare similare alle immagini PC9 utilizzate.

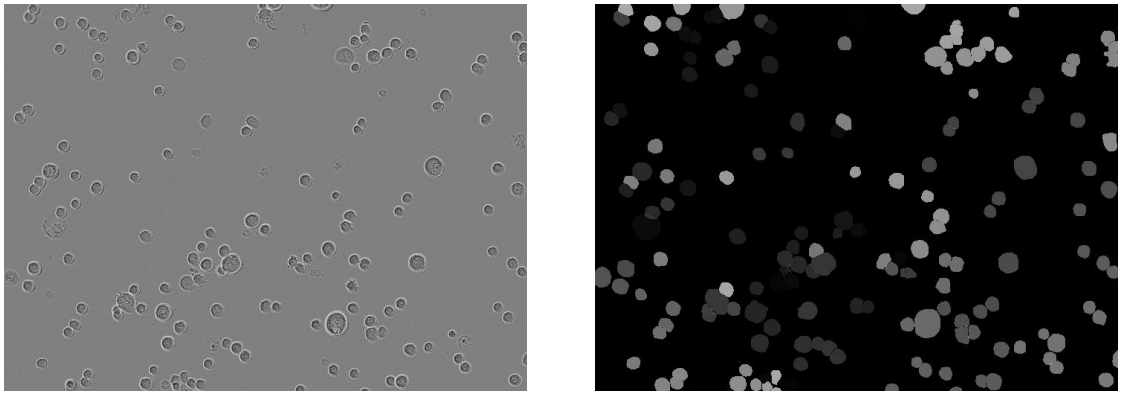


Figura 45: Immagine e maschera del dataset di training SkBr3 per Cellpose

In questo caso il modello è stato addestrato su 200 epoche, con data augmentation e considerando i pesi di un modello preaddestrato. Successivamente è stato testato sulle immagini fornite dal laboratorio di Medicina e si è ottenuto il risultato presentato in Figura 46.

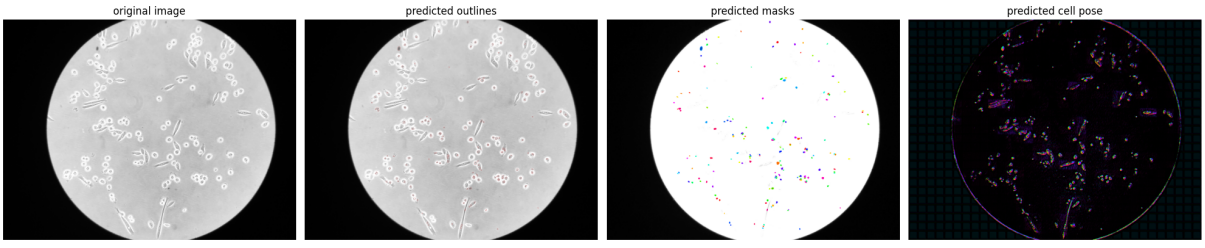


Figura 46: Risultato del testing sul modello Cellpose addestrato con il dataset SkBr3

Il risultato risulta soddisfacente sia dal punto di vista visivo che dal punto di vista delle regioni individuate al suo interno, le quali possono essere accostate a quelle

evidenziate nella maschera disegnata manualmente dagli operatori di laboratorio. La Tabella 2 ne mostra il confronto per le tre immagini PC9.

Anche se risulta essere ancora lontano dal ground truth, fornisce uno strumento su cui poter lavorare, rendendo quindi Cellpose la soluzione da seguire al fine di realizzare il task richiesto. Per questo motivo è stata presa in esame la rete neurale costruita da Cellpose per la segmentazione cellulare, cercando di ricavarne la combinazione di parametri migliore per lo scopo qui richiesto.

Immagine PC9 P1 20X 48h 1			
	Number of cells	**Spindle Index** > 3 [%]	**Spindle Index** < 3 [%]
Cellpose 2D	156	1.28	98.72
Ground truth	155	12.26	87.74
Immagine PC9 P1 20X 48h 4			
Cellpose 2D	232	0.86	99.14
Ground truth	164	15.24	84.76
Immagine PC9 P1 20X 48h 7			
Cellpose 2D	163	0.0	100.0
Ground truth	109	25.69	74.31

Tabella 2: Confronto tra la maschera ottenuta con Cellpose e la ground truth (maschera fornita da medicina)

Avendo dimostrato che Cellpose è la rete che offre le migliori performance sulle immagini di test, nelle sezioni successive verrà analizzata più nel dettaglio. L'obiettivo consiste nell'effettuare un confronto più approfondito delle diverse configurazioni di Cellpose per il task specifico di cell segmentation, qui in esame.

3.2 Cellpose 2D

Cellpose offre i risultati migliori a seguito della segmentazione se confrontati alla maschera ottenuta manualmente. Per questo motivo nella seguente sezione verranno meglio analizzate le prestazioni della rete neurale menzionata, in particolare verranno studiati i risultati che si possono ottenere dai diversi modelli messi a disposizione da Cellpose. Questo permetterà di individuare la miglior configurazione di modello di rete neurale pre-addestrato e dei parametri per l'inferenza utili alla realizzazione del tool di supporto alla segmentazione cellulare.

Molte applicazioni biologiche richiedono la necessità di segmentare corpi cellulari, membrane e nuclei a partire da immagini ottenute dal microscopio. Come già introdotto, il Deep Learning ha permesso di ottenere un grande progresso in questo ambito, ma la maggior parte dei metodi proposti in letteratura risultano essere specifici e specializzati su particolari immagini raccolte in grandi dataset di training. Cellpose si propone di realizzare un metodo di segmentazione che abbia lo scopo di segmentare cellule all'interno di un ampio range di tipi di immagini. La biologia cellulare richiede la misurazione di diverse proprietà, tra cui forma, posizione, espressione dell'RNA e della proteina, da assegnare alla singola cellula effettuando una segmentazione dell'intero volume dell'immagine in corpi cellulari. Questo step risulta semplice se le cellule sono sufficientemente distanziate, ma in molte applicazioni questo non accade e le cellule risultano difficili da separare. I metodi che si utilizzano per la segmentazione, in generale, possono essere completamente manuali dove l'etichettatura dell'immagine è effettuata dall'utente, o completamente automatici e basati su reti neurali che stimano i valori dei parametri. Gli ultimi presentano vantaggi rispetto ai primi, ma necessitano di essere addestrati su dataset specializzati, i quali non riescono a generalizzare bene su altre tipologie di dati. Questo richiede di avere grandi quantità di immagini segmentate manualmente dall'uomo, necessarie ad aumentare le performance dei modelli di DL. Poiché le cellule, soprattutto se raccolte da differenti laboratori, possono avere forme e caratteristiche diverse, un unico modello potrebbe non essere in grado di segmentarle tutte [29].

Cellpose è un insieme di modelli di segmentazione basati su architettura U-Net, che a partire da immagini al microscopio in input genera una mappa di probabilità cellulare e un vettore di flusso che indirizza ogni pixel verso il centro della cellula. Cellpose mette a disposizione modelli pre-addestrati per ciascun sottoinsieme del dataset di training, in modo che ogni modello corrisponda ad una specifica modalità di acquisizione delle immagini al microscopio. Questo approccio lo rende particolarmente adatto a segmentare immagini di cellule con caratteristiche diverse, risolvendo il problema della variabilità tra tipi di cellule [30].

I modelli classici di segmentazione basati sull'algoritmo "watershed" utilizzano i valori della scala di grigi dell'immagine per creare una mappa topologica, all'interno della quale

i bacini identificati rappresentano le regioni segmentate. Questi metodi funzionano bene quando gli oggetti individuati hanno profili di intensità che decadono dal centro, formando così un singolo bacino. Il problema si presenta quando alcune forme cellulari hanno bacini costituiti di multiple intensità. Cellpose si propone di utilizzare maschere "ground truth" disegnate dall'uomo all'interno dell'architettura U-Net per effettuare predizioni sui gradienti spaziali di ogni pixel. Questa procedura ha l'obiettivo di assegnare ogni pixel dell'immagine allo specifico centro di appartenenza, riuscendo ad individuare le cellule e le loro forme. L'architettura U-Net, per la realizzazione del modello Cellpose, è stata modificata al fine di processare in modo differente immagini aventi stili diversi [29].
La rete è rappresentata in Figura 47 insieme alla logica che si trova dietro la realizzazione di Cellpose.

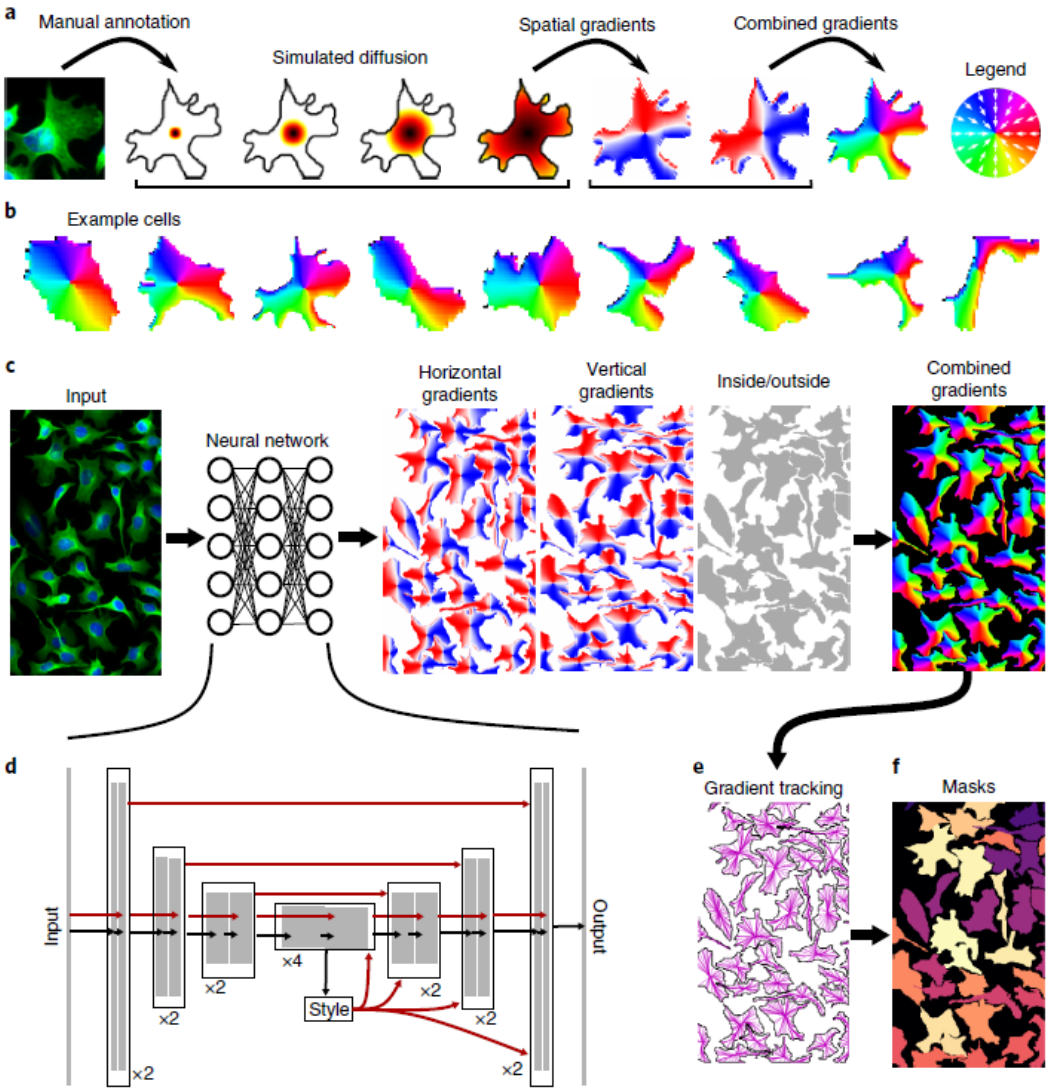


Figura 47: Logica Cellpose e rete U-Net modificata per realizzare il modello [29]

Il dataset utilizzato da Cellpose è costituito da immagini raccolte da una varietà di fonti e, in particolare, ottenute da ricerche su internet effettuate utilizzando parole chiave,

come ad esempio "citoplasma", "microscopia cellulare", "cellule fluorescenti" eccetera. All'interno del dataset si possono individuare:

- 316 immagini di proteine marcate con fluorescenza e localizzate nel citoplasma;
- 50 immagini di microscopia in campo chiaro;
- 58 immagini di cellule con membrana etichettate;
- 86 immagini provenienti da altre tipologie di microscopia;
- 98 immagini non di microscopia, contenenti un gran numero di oggetti ripetuti (ad esempio frutta, pietre, meduse...) che hanno lo scopo di aumentare la generalizzazione della rete.

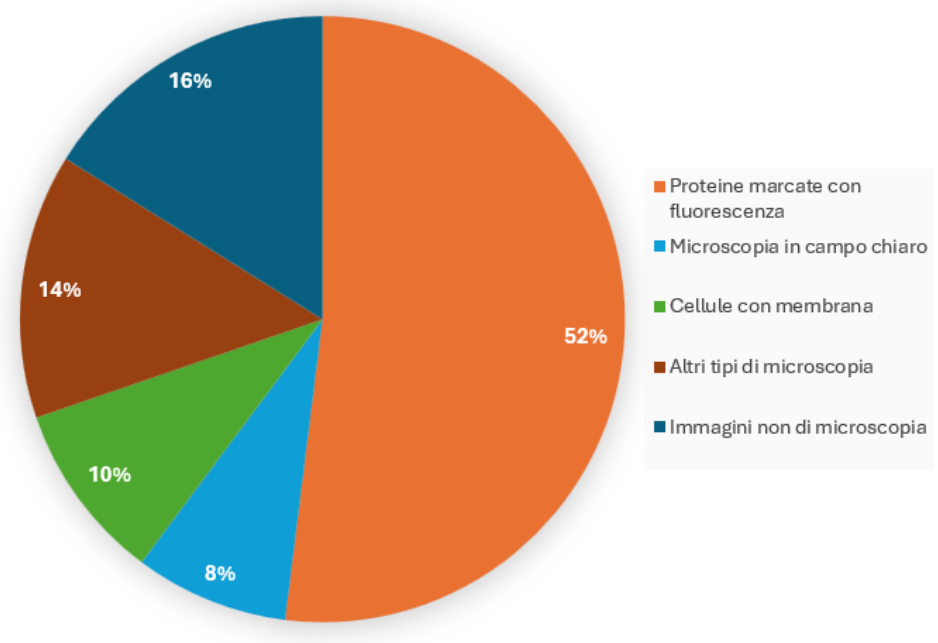


Figura 48: Rappresentazione della distribuzione del dataset di training di Cellpose

In Figura 48 è riportata una rappresentazione della distribuzione del dataset costruito da Cellpose.
Nella sezione precedente è stato mostrato che Cellpose, quando addestrato e testato su dati specializzati, si dimostra più performante rispetto ad altri metodi di segmentazione come U-Net e StarDist, poiché riesce a catturare forme cellulari più complesse. Tuttavia, tutti i modelli, compreso Cellpose, mostrano prestazioni inferiori se valutati sull'intero dataset, evidenziando la necessità di realizzare algoritmi più generalizzati per la segmentazione cellulare. In particolare, Cellpose si comporta meglio su immagini di cellule non fluorescenti, di membrane cellulari e sulle immagini della "Cell Image Library". Questo suggerisce che la segmentazione di immagini, nelle quali si presentano cellule di dimensioni

molto variabili, rappresenti un compito complesso. Nonostante ciò, Cellpose ha il potenziale per migliorare l'analisi quantitativa delle cellule, rendendola riproducibile e scalabile su grandi dataset [29].
Un confronto tra le performance di Cellpose ed altri modelli presenti allo stato dell'arte per la segmentazione cellulare, risulta utile per comprendere i punti di forza e le debolezze di ciascun approccio. L'algoritmo watershed che genera una mappa topologica basata sui valori della scala di grigi dell'immagine confrontati con una soglia, presenta una variante sensibile al rumore, diventando inadeguato quando i bordi della cellula non sono evidenti. Questo lo rende non adatto a dataset che richiedono una segmentazione accurata. I metodi che implementano una rete U-Net, invece, differenziano lo sfondo dalle cellule e dai loro bordi, ma presentano la problematica che alcuni pixel dei bordi potrebbero essere rilevati erroneamente come cellule. Altri modelli che utilizzano forme a stella o poligonali producono più di un unico poligono per le cellule allungate, risultando in una maschera che non riempie l'intera cellula. Cellpose, rispetto alle precedenti soluzioni, è un metodo più robusto al rumore ed a morfologie differenti delle cellule, il che lo rende in grado di distinguerle dallo sfondo pixel per pixel. Cellpose fa predizioni sul gradiente del flusso rispetto al centro della cellula, per cui lavora meglio anche su cellule allungate, rendendolo la scelta migliore in termini di modello da utilizzare per la segmentazione di immagini cellulari [30].

3.2.1 Confronto tra Reti e Dataset di Cellpose

Scelto Cellpose per la realizzazione del tool che permette di segmentare le immagini appartenenti al dataset di test, si vuole ora individuare un metodo per ottenere correlazione tra informazione morfologica e di fluorescenza, dati informativi sull'evoluzione della cellula. In questo senso risulta necessario selezionare il miglior modello pre-addestrato e le migliori impostazioni parametriche di Cellpose che possano assicurare elevate prestazioni sulle immagini a disposizione.
Cellpose è costituito da diverse reti neurali pre-trainate su specifici dataset che si differenziano a seconda dell'applicazione a cui più si adattano. Tra queste si hanno:

- Il modello Cytoplasm: addestrato su immagini a due canali dove il primo è il canale da segmentare ed il secondo è un canale rappresentante il nucleo della cellula (opzionale);

- Il modello Cytoplasm2: è un modello Cytoplasm aggiornato, poiché addestrato anche su immagini inserite da utenti;

- Il modello Nuclei: addestrato su immagini a due canali, dove il primo rappresenta il canale da segmentare ed il secondo è sempre impostato pari ad un array di zeri;

- Il modello LiveCell: addestrato su immagini aventi solo un canale citoplasmatico;

- Il modello TissueNet: addestrato su immagini aventi sia un canale citoplasmatico che un canale nucleare;

Queste possono essere utilizzate direttamente come modelli per effettuare predizioni su delle immagini fornite dall'utente, oppure possono essere riaddestrate su dataset differenti. ZeroCostDL4Mic, insieme al notebook che implementa Cellpose, mette a disposizione anche dei dataset su cui possono essere addestrate le reti: A172, BT474, BV2, Huh7, MCF7, SHSY5Y, SkBr3, SKOV3. Un esempio per ogni rispettiva coppia immagine-maschera è presentato in Figura 49.

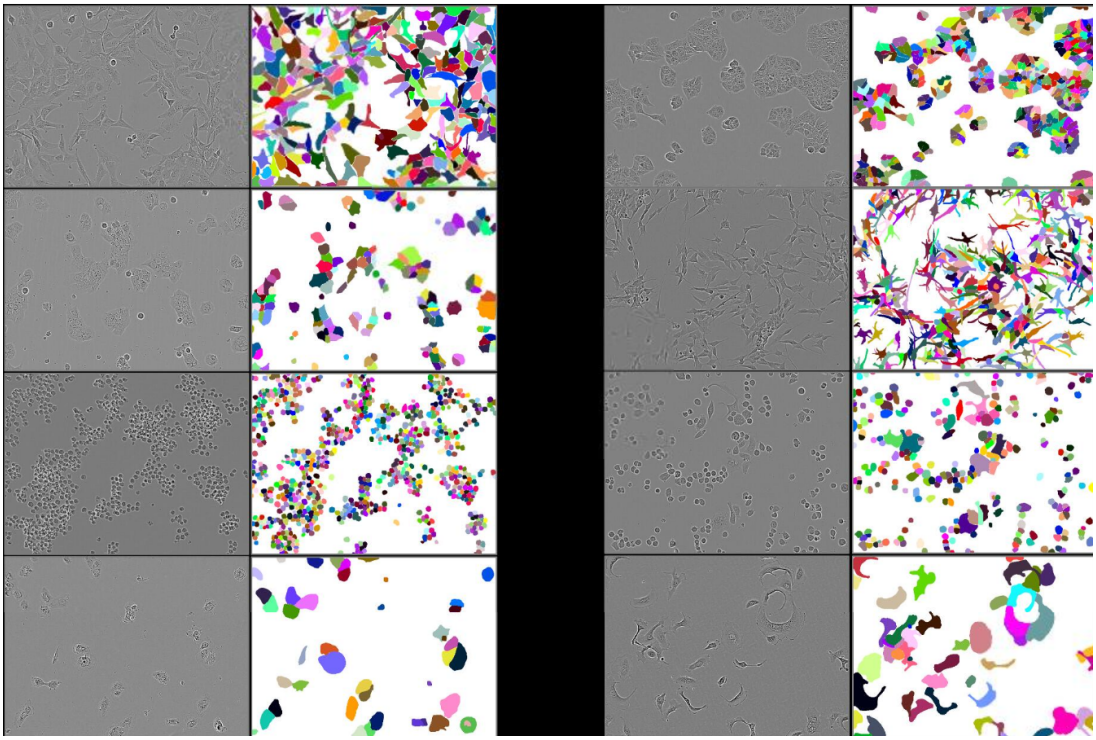


Figura 49: Sommario dataset per il training di Cellpose [31]

Il notebook di ZeroCost permette inoltre di aggiustare alcuni parametri per la predizione, consentendo all'utente di individuare l'impostazione più adatta ai dati a sua disposizione. I parametri sono:

- Data folder: cartella all'interno della quale si trovano le immagini su cui effettuare le predizioni. Le immagini devono essere grayscale e in formato TIFF;

- Result folder: cartella all'interno della quale verranno salvate le predizioni (maschere) risultanti;

- Data type: tipologia di immagine, a scelta tra "Single images" o "Stacks". In questo caso si utilizza la prima impostazione;

- Model choice: il modello da utilizzare per effettuare le predizioni. Può essere un proprio modello precedentemente addestrato o un modello pre-trainato messo a disposizione da Cellpose;

- Prediction model: da specificare solo se il precedente parametro è impostato ad "own model";

- Channel to segment: canale da segmentare, può essere "grayscale", "blue", "green", "red". Nel caso qui presentato è stato impostato a "grayscale";

- Nuclear channel: da indicare se nelle immagini è presente un canale nucleare, può essere "none" (qui impostato), "green", "red" o "blue";

- Object diameter: indica il diametro degli oggetti da segmentare in pixel (cellule o nuclei). Se impostato a 0 viene stimato automaticamente dall'algoritmo per ogni immagine, impostazione scelta nel caso in esame;

- Flow threshold: è un parametro che controlla il massimo errore permesso sui flussi di gradiente delle maschere. Questi definiscono i contorni degli oggetti segmentati nell'immagine in base alle variazioni di intensità rilevate. La soglia va aumentata se Cellpose non restituisce un numero sufficiente di maschere e va diminuito se il modello ne genera troppe di forma errata. Il valore di default è 0.4;

- Mask threshold: i pixel con valore superiore a questo parametro vengono utilizzati per determinare le maschere. Se Cellpose non restituisce tante maschere quante se ne vorrebbero, la soglia va diminuita; similmente va aumentata se Cellpose restituisce troppe maschere. Il valore di default è 0.0.

3.3 Analisi del Confronto tra modelli Cellpose

Ora l'obiettivo consiste nell'individuare la miglior variante di Cellpose, ovvero il modello che consente di ottenere le migliori prestazioni dal punto di vista della segmentazione sulle immagini fornite dal laboratorio (dataset di test). Per far questo si segue una procedura simile a quella già applicata per individuare la miglior tipologia di rete neurale da utilizzare: addestramento del modello con il dataset a disposizione, prediction sul dataset di testing, confronto con il CSV appartenente ai dati messi a disposizione dai laboratori della facoltà di Medicina e aggiustamento degli iper-parametri. L'analisi viene divisa sulle due differenti tipologie di immagini a disposizione, PC9 e A549, per evidenziare eventuali differenze dovute alla diversa coltura cellulare.
I parametri utilizzati per il confronto sono ancora il numero totale di cellule individuate sulla maschera e la percentuale di cellule oblunghe o circolari. Per semplicità vengono riportate le formule dei parametri.

$$Spindle\ \ Index: \quad S_i = \frac{d_{maggiore}}{d_{minore}}$$

$$Spindle\ \ Flag: \begin{cases} Se\ \ S_i > 3\ \ cellula\ \ oblunga & \implies S_f = 1 \\ Se\ \ S_i < 3\ \ cellula\ \ circolare & \implies S_f = 0 \end{cases}$$

3.3.1 Immagini PC9

Le immagini PC9 mostrate in Figure 22-24 vengono date in ingresso ai diversi modelli di Cellpose che si vogliono analizzare, inizialmente senza nessuna modifica nei parametri ovvero lasciando tutto di default. In particolare la "flow threshold", parametro modificabile in relazione al numero di cellule che si vogliono individuare, è impostato a 0.4. Ottenute le relative maschere, queste vengono elaborate per ottenere un file CSV contenente tutte le caratteristiche utili di ogni cellula individuata all'interno della maschera.
Da queste ultime si possono estrarre sia il numero totale di regioni (cellule) individuate, che le misure delle diagonali maggiori e minori, utili al calcolo dell'S_i e del conseguente S_f.
Ottenute le informazioni necessarie, è poi possibile confrontare i valori ottenuti con quelli ricavati dal file CSV fornito dal laboratorio di Medicina, il quale costituisce il ground truth. Il risultato è mostrato nella Tabella 3.
Successivamente si esegue la stessa procedura modificando il parametro "flow threshold" ed impostandolo al minimo (0.0), per osservare se questo porta a variazioni nella predizione e dunque nelle caratteristiche della maschera. Il risultato è mostrato nella Tabella 4.
Ad ogni risultato presentato nelle tabelle corrisponde una maschera che Cellpose restituisce a partire dall'immagine fornita in ingresso. Alcune di queste possono essere osservate in Figura 50.

Immagine PC9 P1 20X 48h 1, Flow Threshold = 0.4			
	Number of cells	**Spindle Index > 3 [%]**	**Spindle Index < 3 [%]**
Ground truth	155	12.26	87.74
Cytoplasm	146	6.16	93.84
Cytoplasm2	**160**	**8.13**	**91.87**
LiveCell	0	NaN	NaN
Nuclei	129	1.55	98.45
TissueNet	0	NaN	NaN
SkBr3	163	0	100
SHSY5Y	316	0	100
A172	54	1.85	98.15
BT474	182	0.55	99.45
BV2	26	0	100
HUH7	98	0	100
MCF7	105	0.95	99.05
SKOV3	35	0	100

Tabella 3: Confronto ground truth - modelli Cellpose per la prima immagine PC9, con flow threshold impostata a 0.4

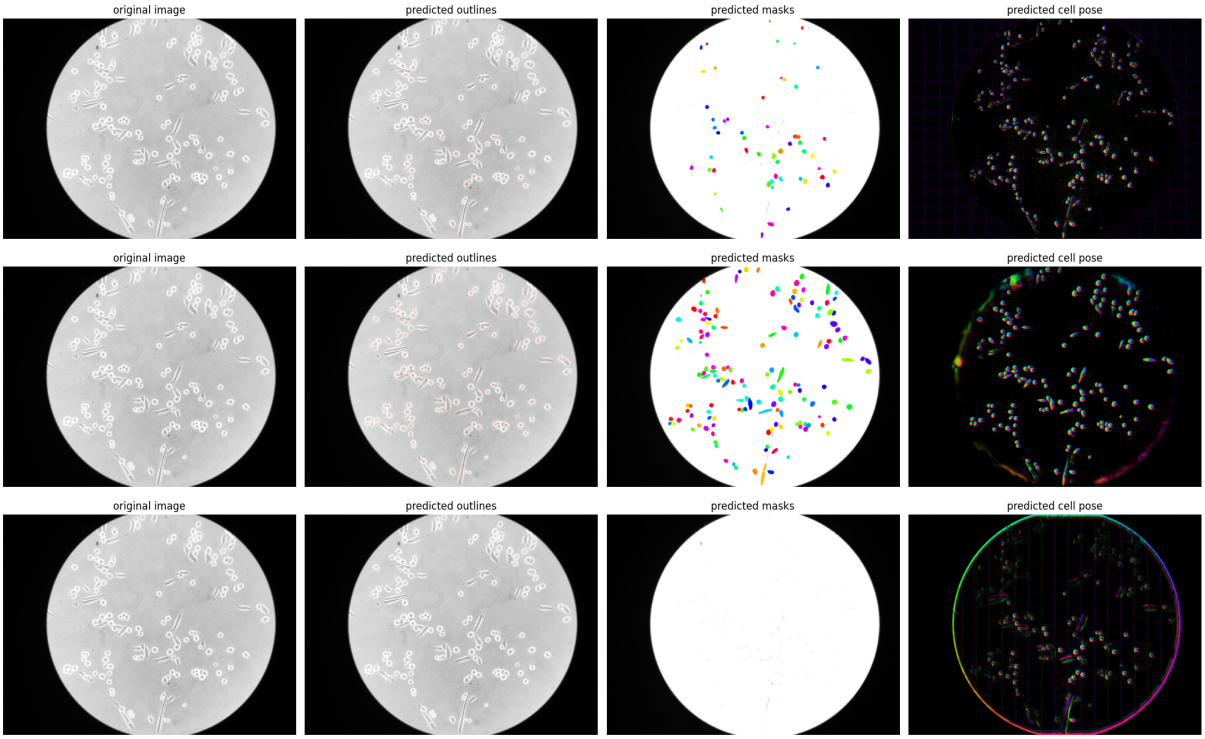


Figura 50: Maschere ottenute con flow threshold a 0.4 con i modelli: in alto A172, al centro Cytoplasm, in basso TissueNet

Immagine PC9 P1 20X 48h 1, Flow Threshold = 0.0			
	Number of cells	**Spindle Index** > 3 [%]	**Spindle Index** < 3 [%]
Ground truth	155	12.26	87.74
Cytoplasm	377	40.85	59.15
Cytoplasm2	**170**	**8.82**	**91.18**
LiveCell	225	22.67	77.33
Nuclei	157	5.73	94.27
TissueNet	0	NaN	NaN
SkBr3	938	9.10	90.9
SHSY5Y	316	5.06	94.94
A172	507	12.43	87.57
BT474	508	12.80	87.20
BV2	204	0.98	99.02
HUH7	239	16.74	83.26
MCF7	647	5.56	94.44
SKOV3	1012	18.38	81.62

Tabella 4: Confronto ground truth - modelli Cellpose per la prima immagine PC9, con flow threshold impostata a 0

Dalle tabelle si nota che il modello che fornisce i risultati più coerenti con il ground truth è la rete pre-trainata Cytoplasm2. Per un'analisi più approfondita si osservano poi i risultati anche con valori intermedi della flow threshold, i quali evidenziano che la scelta più corretta è l'impostazione a 0.0. La Tabella 5 mostra questa analisi.

Immagine PC9 P1 20X 48h 1			
	Number of cells	**Spindle Index** > 3 [%]	**Spindle Index** < 3 [%]
Cytoplasm2 Th = 0.0	170	8.82	91.18
Cytoplasm2 Th = 0.1	109	5.50	94.50
Cytoplasm2 Th = 0.2	148	6.76	93.24
Cytoplasm2 Th = 0.3	157	7.64	92.36
Cytoplasm2 Th = 0.4	160	8.13	91.87
Ground truth	155	12.26	87.74

Tabella 5: Confronto ground truth - modello preaddestrato Cellpose (Cytoplasm2) per prima immagine PC9, con diversi valori della flow threshold

Si può concludere che il modello che permette di ottenere i risultati più assimilabili al ground truth è Cytoplasm2 con flow threshold impostata a 0.0. L'analisi viene dunque estesa a tutte le immagini PC9, con i risultati riportati in Tabella 6.

Immagine PC9 P1 20X 48h 1			
	Number of cells	**Spindle Index** > 3 [%]	**Spindle Index** < 3 [%]
Cytoplasm2, Th = 0.0	170	8.82	91.18
Ground truth	155	12.26	87.74
Immagine PC9 P1 20X 48h 4			
Cytoplasm2, Th = 0.0	205	10.73	89.27
Ground truth	164	15.24	84.76
Immagine PC9 P1 20X 48h 7			
Cytoplasm2, Th = 0.0	190	18.95	81.05
Ground truth	109	25.69	74.31

Tabella 6: Confronto ground truth - modello preaddestrato Cytoplasm2 di Cellpose2D per le tre immagini PC9, con flow threshold impostata a 0

Le maschere ottenute con Cytoplasm2 e flow threshold = 0.0 sono mostrate in Figura 51 rispettivamente per le tre immagini PC9 a disposizione.

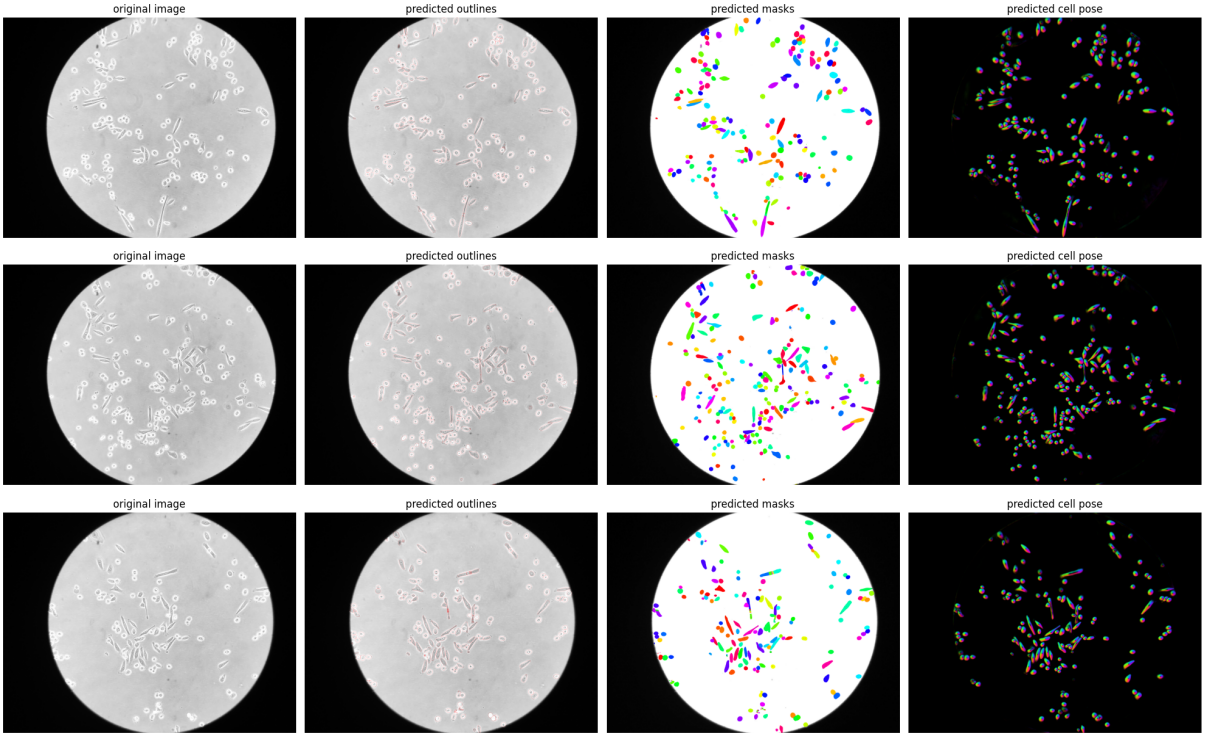


Figura 51: Maschere ottenute da Cytoplasm2 e flow threshold pari a 0 dall'alto verso il basso, rispettivamente, per le immagini PC9 P1 20X 48h 1, PC9 P1 20X 48h 4, PC9 P1 20X 48h 7

3.3.2 Immagini A549

Un'analisi analoga è stata poi eseguita per il successivo gruppo di immagini a disposizione (A549 in Figure 25-29) allo scopo di verificare se Cytoplasm2 sia adatto anche per un diverso tipo di coltura cellulare. Anche in questo caso si esegue un confronto tra il ground truth e i diversi modelli di Cellpose nel caso con flow threshold pari a 0.4 (valore di default) e pari a 0.0.

Immagine IMG 4105, Flow Threshold = 0.4			
	Number of cells	**Spindle Index** > 3 [%]	**Spindle Index** < 3 [%]
Ground truth	301	6.98	93.02
Cytoplasm	69	8.70	91.30
Cytoplasm2	**205**	**4.88**	**95.12**
LiveCell	0	NaN	NaN
Nuclei	65	27.69	72.31
TissueNet	0	NaN	NaN
SkBr3	247	0.0	100.0
SHSY5Y	247	0.0	100.0
A172	12	8.33	91.67
BT474	174	1.15	98.85
BV2	29	0.0	100.0
HUH7	26	3.85	96.15
MCF7	81	1.23	98.77
SKOV3	12	41.67	58.33

Tabella 7: Confronto ground truth - modelli Cellpose per la prima immagine A549, con flow threshold impostata a 0.4

Come per il caso precedente si nota dalle Tabelle 7, 8 che la scelta migliore e che più si avvicina ai risultati ottenibili dalla segmentazione manuale è l'utilizzo di Cytoplasm2.
Dalla Tabella 9, poi, si nota che anche in questa situazione è preferibile l'impostazione della soglia a 0.0 per avere una maggiore affinità con il risultato fornito dal ground truth.
Ne deriva quindi lo studio delle cinque immagini A549 fornite al modello Cytoplasm2 che ne restituisce le rispettive maschere. Il risultato è mostrato in Tabella 10.
In Figura 52 sono mostrate le maschere ottenute dalla predizione effettuata con Cytoplasm2 e flow threshold a 0.0 per le cinque immagini A549 costituenti il ground truth.

Immagine IMG 4105, Flow Threshold = 0.0			
	Number of cells	**Spindle Index** > 3 [%]	**Spindle Index** < 3 [%]
Ground truth	301	6.98	93.02
Cytoplasm	618	39.35	61.65
Cytoplasm2	**497**	**14.89**	**85.11**
LiveCell	136	31.62	68.38
Nuclei	414	27.05	72.95
TissueNet	0	NaN	NaN
SkBr3	2175	7.31	92.69
SHSY5Y	2175	7.31	92.69
A172	3384	14.86	85.14
BT474	729	8.09	91.91
BV2	220	5.45	94.55
HUH7	294	25.85	74.15
MCF7	1633	7.72	92.28
SKOV3	6190	18.30	81.70

Tabella 8: Confronto ground truth - diversi modelli Cellpose per prima immagine A549, con flow threshold impostata a 0

Immagine IMG 4105			
	Number of cells	**Spindle Index** > 3 [%]	**Spindle Index** < 3 [%]
Cytoplasm2 Th = 0.0	497	14.89	85.11
Cytoplasm2 Th = 0.1	59	5.08	94.92
Cytoplasm2 Th = 0.2	139	7.19	92.81
Cytoplasm2 Th = 0.3	139	7.19	92.81
Cytoplasm2 Th = 0.4	205	4.88	95.12
Ground truth	301	6.98	93.02

Tabella 9: Confronto ground truth - Cytoplasm2 per la prima immagine A549, con diversi valori della flow threshold

Immagine IMG_4105			
	Number of cells	**Spindle Index** > 3 [%]	**Spindle Index** < 3 [%]
Cytoplasm2, Th = 0.0	497	14.89	85.11
Ground truth	301	6.98	93.02
Immagine IMG_4106			
Cytoplasm2, Th = 0.0	458	13.54	86.46
Ground truth	312	5.45	94.55
Immagine IMG_4111			
Cytoplasm2, Th = 0.0	395	10.89	89.11
Ground truth	329	4.86	95.14
Immagine IMG_4113			
Cytoplasm2, Th = 0.0	319	12.23	87.77
Ground truth	284	5.99	94.01
Immagine IMG_4114			
Cytoplasm2, Th = 0.0	320	14.06	85.94
Ground truth	344	3.205	96.80

Tabella 10: Confronto ground truth - Cytoplasm2 per le cinque immagini A549, con flow threshold impostata a 0

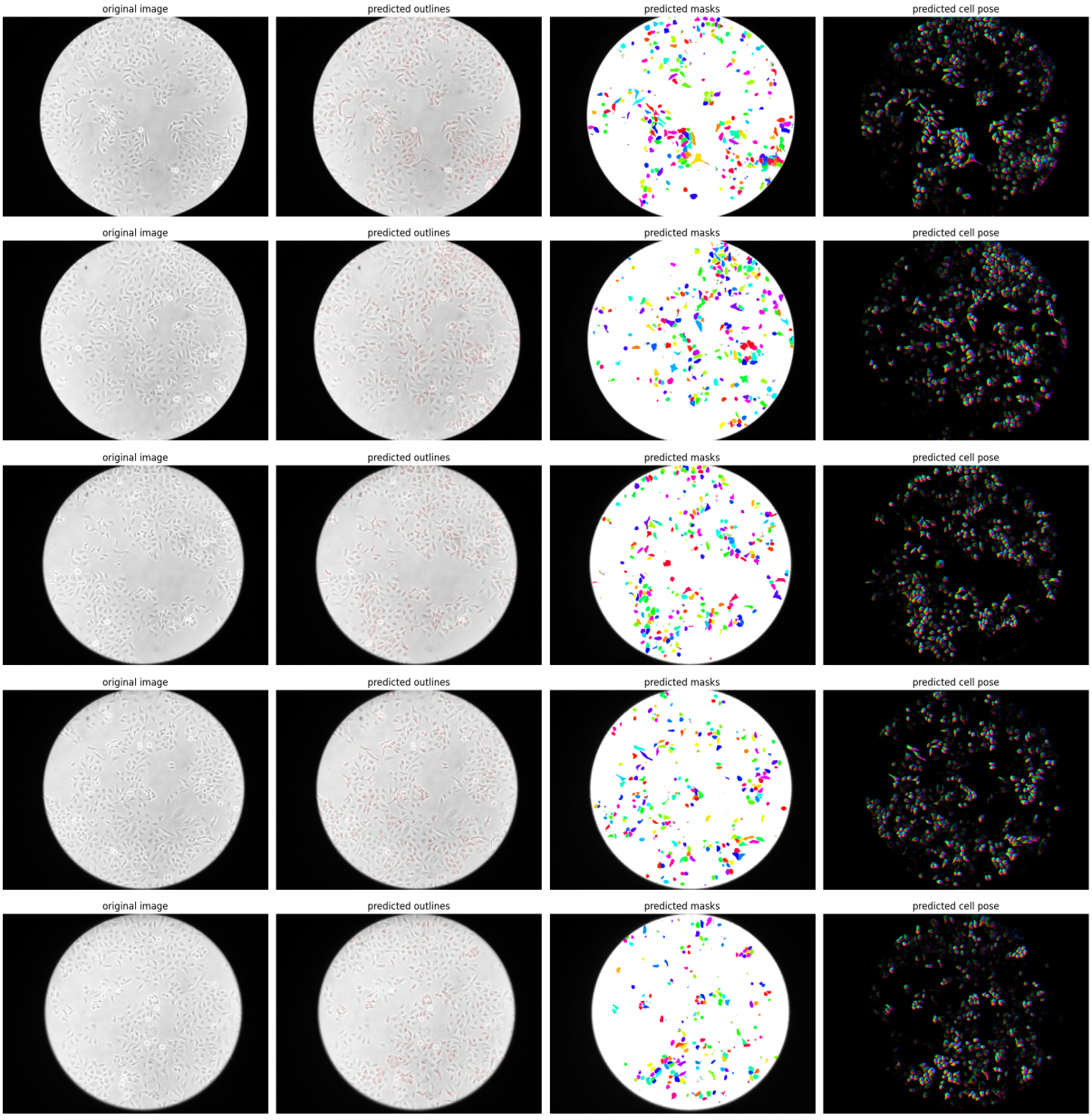


Figura 52: Maschere ottenute da Cytoplasm2 e flow threshold pari a 0 dall'alto verso il basso, rispettivamente, per le immagini IMG 4105, IMG 4106, IMG 4111, IMG 4113, IMG 4114

4 Federated Learning

Per poter ottenere prestazioni ancora migliori con il DL nel campo della segmentazione cellulare sarebbe ideale poter addestrare un modello interamente con immagini cellulari del tipo che si intende analizzare, o in alternativa si potrebbe riaddestrare un modello pre-trainato di Cellpose con queste immagini. In questo modo, le maschere ottenute sarebbero maggiormente confrontabili con quelle che si ottengono effettuando una segmentazione manuale. Per fare ciò, tuttavia, è necessaria una grande quantità di immagini di colture cellulari già segmentate ed abbinate alla rispettiva maschera. Questa operazione richiede un notevole sforzo umano, poiché la segmentazione deve essere necessariamente eseguita manualmente. Tale esigenza rappresenta uno dei principali motivi che ha portato a considerare un cambiamento di paradigma, passando da una logica di apprendimento centralizzato ad una di apprendimento distribuito (Federated Learning, FL).
Nel contesto fornito dal FL è possibile sfruttare dati provenienti da molteplici fonti e da centri di ricerca differenti, riducendo la necessità di avere un'unica base di dati centralizzata ed alleviando l'onere legato alla segmentazione manuale. Questo approccio consente quindi di ottimizzare l'addestramento dei modelli, sfruttando le risorse distribuite e migliorando la precisione delle predizioni, senza richiedere però un eccessivo impegno a livello locale. Un ulteriore vantaggio dato dall'uso del FL, particolarmente utile nel caso d'uso qui presentato, è la tutela della privacy garantita dal sistema implementato per l'addestramento distribuito. La condivisione dei pesi, effettuata dopo la fase di apprendimento locale, infatti, non implica la condivisione dei dati personali di ciascun client agli altri dispositivi connessi al server centrale. Questo consente ai laboratori di mantenere riservate le informazioni dei pazienti in un contesto in cui i dati sensibili sono presenti in grande quantità [11].
Il Federated Learning si basa sul concetto di "round", il quale consiste di cinque step fondamentali: inizializzazione del modello globale, distribuzione del modello ai nodi client, addestramento locale, invio dei pesi aggiornati al server e aggregazione dei pesi per l'aggiornamento del modello globale [9].
Nell'applicazione del FL al caso in esame, utilizzando le reti messe a disposizione da Cellpose, si sfrutta il framework FlowerAI che consente di implementare il Federated Learning tramite un insieme di tool integrati, configurabili in base alle esigenze specifiche del task. L'obiettivo consiste nell'implementare il FL tramite Flower sfruttando le reti di Cellpose, e nel dimostrare che le prestazioni del modello centralizzato e quelle del modello distribuito sono confrontabili. Questo permetterebbe di concludere che il Federated Learning è un valido sostituto dell'approccio classico, fornendo al contempo una serie di vantaggi (privacy, apprendimento continuo, semplificazione del ri-addestramento dei modelli utilizzando dati provenienti da organizzazioni diverse...) che consentono di applicare le tecniche di DL anche in quei campi in cui fino ad ora non è stato possibile.

4.1 Configurazione di Flower Framework

Per comprendere i procedimenti e i passaggi necessari all'implementazione del Federated Learning con Flower, si è fatto riferimento al tutorial riportato in [32].
Le sezioni principali comprendono:

1. Importazione delle librerie necessarie per l'implementazione del framework;
2. Definizione del modello utilizzato sia dal server che dai client in fase di, rispettivamente, aggregazione ed addestramento;
3. Caricamento dei dati necessari al training;
4. Funzioni di get e set dei parametri per l'aggiornamento dei pesi del modello;
5. Funzione di training eseguita da ogni singolo client;
6. Flower Client: simula le operazioni che esegue un client quando viene invocato durante l'esecuzione dei round;
7. Virtual Client Engine: consente di creare un Flower Client su richiesta, identificato da un ID;
8. Strategia: implementa l'algoritmo di FL, come ad esempio il Federated Averaging (FedAvg);
9. Simulazione: avvia la simulazione del FL.

La seconda prevede la definizione del modello, che in questo caso è fatto corrispondere ad una rete pre-addestrata di Cellpose, ovvero Cytoplasm. Per configurare il modello si segue l'approccio presentato nel tutorial, definendo una classe chiamata *CytoplasmModel* tramite l'utilizzo dei moduli messi a disposizione da PyTorch. In particolare questa classe è costituita da due metodi:

- Il primo è $\_\_init\_\_()$ e definisce la struttura della rete neurale, in cui il modello di Cellpose (Cytoplasm) è configurato a partire dai pesi di un modello pre-addestrato che deve essere definito dal client;
- Il secondo è $forward()$ che descrive come la rete si deve comportare per la valutazione su nuovi dati, restituendo come output la maschera predetta a partire dall'immagine di input.

Per le funzioni di get e set dei parametri è stata mantenuta l'implementazione presentata nel tutorial, dove semplicemente si prevede una condivisione dei parametri dal server verso i client e viceversa, utile per mantenere sempre aggiornato il modello globale e i pesi da cui i client devono partire per eseguire l'addestramento locale.

Per l'implementazione della funzione di training è stato necessario consultare il codice sorgente di Cellpose per identificare la funzione che esegue l'addestramento del modello, chiamata *train_seg()* e situata nel file *train.py*. A partire da essa viene dunque definita la funzione

$$train\_cellpose\_model(model, train\_dir, test\_dir, epochs : int)$$

la quale implementa le operazioni che il client deve effettuare per compiere l'addestramento in locale sul proprio dataset. Al suo interno vengono prima processate le immagini di train e test per poterle rendere adeguate e, successivamente, si richiama la funzione *train_seg()* che esegue l'addestramento vero e proprio. Infine i pesi del modello ottenuto vengono salvati poiché dovranno essere riutilizzati in fase di aggregazione.
Il Flower Client descrive la struttura di un client che partecipa al federated learning. Anche la classe *FlowerClient* implementa diversi metodi:

- il metodo *__init__()* permette di inizializzare il client specificando il modello che esso utilizza, la cartella di destinazione del dataset di train e quella del dataset di test;
- il metodo *get_parameters()* richiama alla funzione precedentemente descritta e serve al client per poter ottenere gli aggiornamenti del modello da parte del server;
- il metodo *fit()* richiama alla funzione *set_parameters()* per il settaggio dei parametri ed esegue il train locale tramite la funzione *train_cellpose_model()* introdotta prima.

Il Virtual Client Engine permette di realizzare un'istanza di client temporanea, utile per la fase di training e che evita di intasare la memoria del sistema, riducendo quindi il costo computazionale della simulazione. In questo caso all'interno del *client_fn* vengono definite le operazioni che il client deve compiere quando invocato. In questo caso specifica il modello pre-addestrato di partenza per il client e, inoltre, durante l'addestramento memorizza i valori di loss in un file di testo, utile all'analisi finale dei risultati.
La strategia scelta per questa analisi implementa l'algoritmo di Federated Averaging e specifica che i client che devono essere coinvolti nella simulazione sono il 100%. In fase di aggregazione degli aggiornamenti il FedAvg esegue una media pesata dei pesi ricevuti dai client collegati al server centrale. Nella media pesata si tiene conto della quantità di immagini utilizzate dai diversi client per effettuare l'addestramento locale, in questo modo si può assicurare che ogni aggiornamento abbia la giusta influenza sul modello globale finale.
Infine nell'avvio della simulazione si specifica il numero di client (in questo caso 2, 3, 4 e 6), il numero di round che in questo caso è stato impostato a 3 e la strategia di addestramento precedentemente definita.

4.2 Training

Per studiare la differenza tra i due diversi approcci, centralizzato e distribuito, il confronto viene effettuato su uno stesso modello per cui il training viene realizzato nelle due diverse metodologie. Si parte quindi da un modello di Cellpose, definito "Baseline model", che è di tipo "cyto" addestrato per 200 epoche sul dataset relativo alla rete pre-addestrata "Cytoplasm", costituito da $n = 540$ immagini. Da questo si possono ottenere i due modelli di cui si vogliono confrontare le prestazioni: il modello "Baseline" viene riaddestrato sul dataset "Cytoplasm2", avente $n = 256$ immagini, con due approcci differenti. Si ottengono così:

- Il "Centralized model" ottenuto addestrando con approccio centralizzato il "Baseline model" per 200 epoche sulla totalità delle immagini contenute nel dataset Cytoplasm2;
- Il "Federated model" ottenuto addestrando con approccio distribuito il "Baseline model" dove ogni client esegue il train per 50 epoche. In questo caso il dataset Cytoplasm2 viene suddiviso in modo random per i diversi client selezionati in fase di simulazione.

Per entrambi i modelli si osserva la funzione di loss che permette di realizzare un confronto tra i risultati delle due diverse tecniche di addestramento. Il valore della loss viene determinato combinando le funzioni Mean Squared Error (MSE) e Binary Cross Entropy (BCE) che misurano l'errore sulla base del confronto tra label predette e label reali.
La MSE calcola l'errore che si esegue nella segmentazione multi-classe (confronta i flussi di gradiente) ed è fornita dalla formula:

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

dove N è il numero di elementi del dataset, $\hat{y}_i$ è l'i-esimo elemento prodotto dal modello e y_i è il valore reale del dataset che $\hat{y}_i$ cerca di predire [33].
La BCE, invece, calcola l'errore che si esegue nella segmentazione binaria (confronta le probabilità cellulari) ed è definita dalla formula:

$$BCE = -\frac{1}{N}\sum_{i=1}^{N} y_i \log p(y_i) + (1 - y_i)\log\left[1 - p(y_i)\right]$$

dove N è il numero di elementi del dataset, y_i è la classe reale, $\log p(y_i)$ è la probabilità di quella classe, $p(y_i)$ è la probabilità di avere 1 e $1 - p(y_i)$ è la probabilità di avere 0 [34].
La loss totale si ottiene quindi come $LOSS = MSE + BCE$.
In Figura 53 è rappresentata la funzione di loss ottenuta a seguito dell'addestramento del "Centralized model" eseguito per 200 epoche.

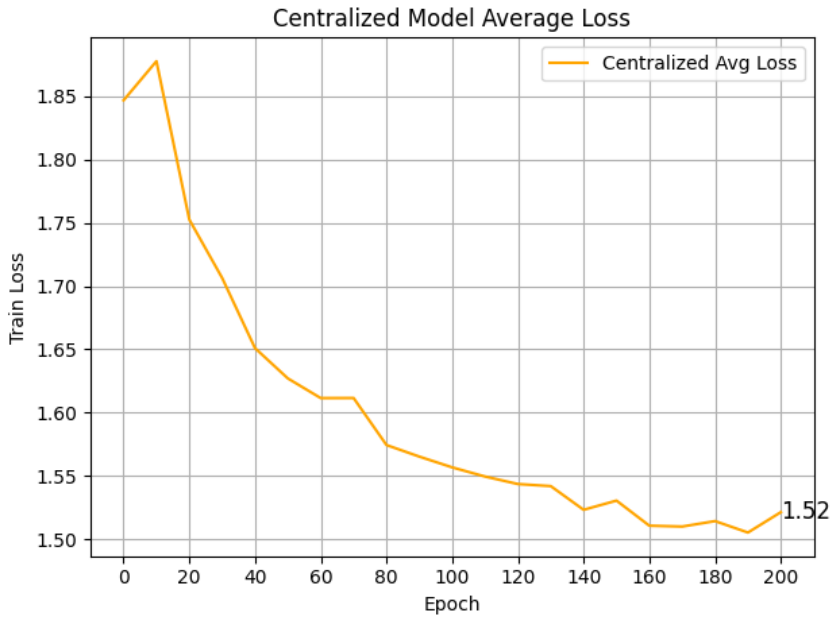


Figura 53: Funzione di loss media per il modello addestrato con approccio centralizzato

Si osserva in questo caso che la funzione di loss ottenuta a seguito di un addestramento eseguito seguendo l'approccio classico, presenta un andamento monotono e decrescente, tipico del centralizzato. Si evidenzia in particolare il valore finale di loss che corrisponde a 1.52.

4.2.1 2 client

L'addestramento di tipo distribuito del Baseline model su 2 client prevede che il dataset Cytoplasm2 venga suddiviso in due porzioni in modo randomico. Per questo motivo la distribuzione dei dati per ogni client non risulta necessariamente bilanciata.
In Figura 54 si osserva l'andamento della loss ottenuta dal training del modello, durante l'intera simulazione, per i due diversi client. In questo caso in ascissa si nota che le epoche arrivano ad un totale di 150, questo è dovuto al fatto che la simulazione esegue 3 round dove ogni client addestra il modello per 50 epoche, risultando quindi in $50 \cdot 3 = 150$ epoche totali.
Si può notare inoltre, che a differenza della loss del modello centralizzato la curva risulta avere un andamento "oscillante", probabilmente dovuto alla diversa distribuzione dei dati tra i due client.
Inoltre all'inizio di un nuovo round, quando avviene l'update del modello a seguito dell'aggregazione dei dati da parte del server, si osserva un cambio del trend della funzione per il secondo client. Questo potrebbe indicare che risente del modello dell'altro client, risultando quindi in un salto nella funzione di loss.

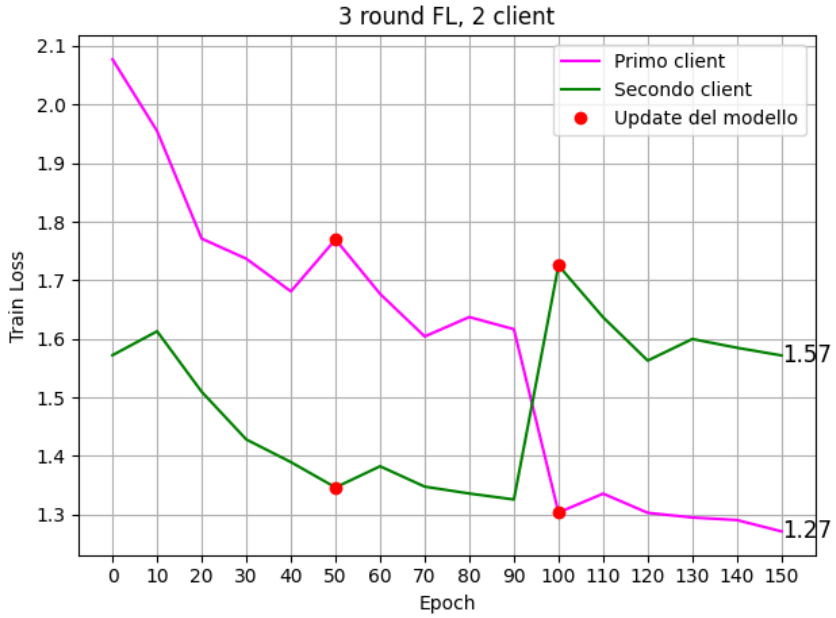


Figura 54: Funzione di loss per il modello addestrato con approccio distribuito su 2 client

4.2.2 3 client

L'applicazione del Federated Learning al modello baseline per ottenere il federated model a 3 client ha come risultato le curve di loss per singolo client mostrate in Figura 55. Anche in questo caso le $n = 256$ immagini del dataset Cytoplasm2 vengono suddivise per i 3 client in modo randomico.
Le considerazioni che si possono fare in relazione alle curve di loss sono analoghe a quelle già espresse per il caso a 2 client.

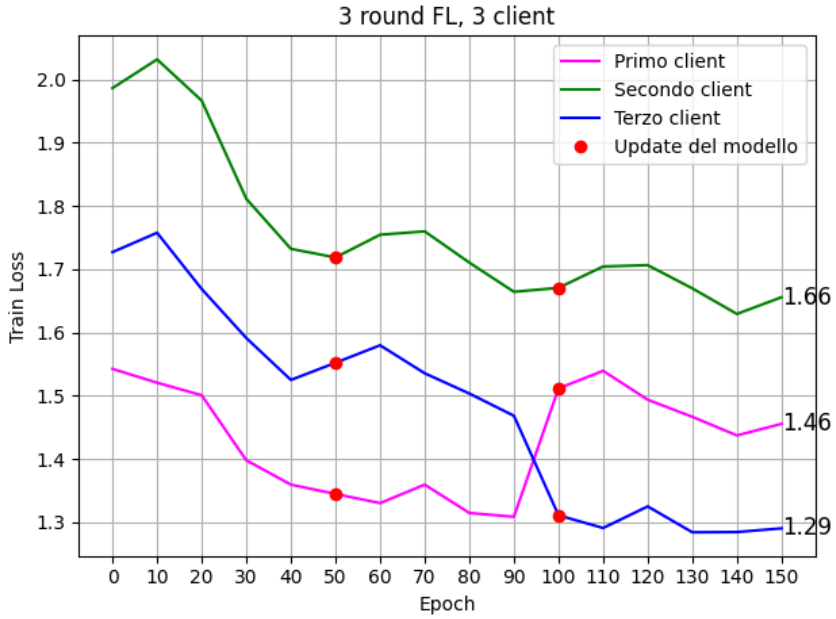


Figura 55: Funzione di loss per il modello addestrato con approccio distribuito su 3 client

4.2.3 4 client

Il discorso è analogo quando si aumenta il numero di client e si esegue l'addestramento distribuito su un totale di 4 client. In questo caso la divisione randomica del dataset porta ad una distribuzione dei dati ancora diversa tra i partecipanti all'addestramento. Questo si rispecchia in una maggiore oscillazione delle funzioni di loss di ogni client, come si può osservare in Figura 56. Inoltre, la loss più instabile rispetto ai casi precedenti può essere attribuita anche al fatto che, ora, i singoli client hanno meno dati a disposizione per il training.
Anche in questo caso, poi, si può notare che all'update del modello si osservano dei salti nell'andamento delle curve.

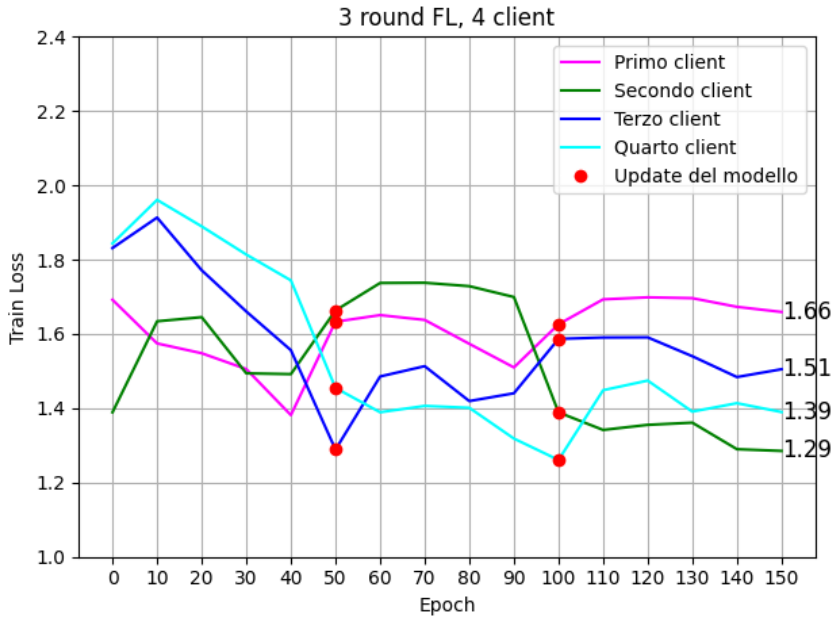


Figura 56: Funzione di loss per il modello addestrato con approccio distribuito su 4 client

4.2.4 6 client

Per il caso a 6 client, infine, l'analisi dei grafici in Figura 57 segue quanto detto per i precedenti.
Si può osservare anche che, analogamente agli andamenti precedenti, i valori di loss sono piuttosto stabili ed oscillano attorno agli stessi valori, i quali alla fine tendono ad un valore simile a quello centralizzato. Questo ci permette di dire che le prestazioni di un addestramento distribuito, dove si coinvolgano più o meno client, non ha prestazioni che vengono influenzate dalla numerosità dei partecipanti all'addestramento.
Nell'ambito della seguente trattazione ci si è fermati ad un massimo di 6 client per la realizzazione di Federated Learning a causa della ridotta dimensionalità del dataset considerato (Cytoplasm2, $n = 256$ immagini).

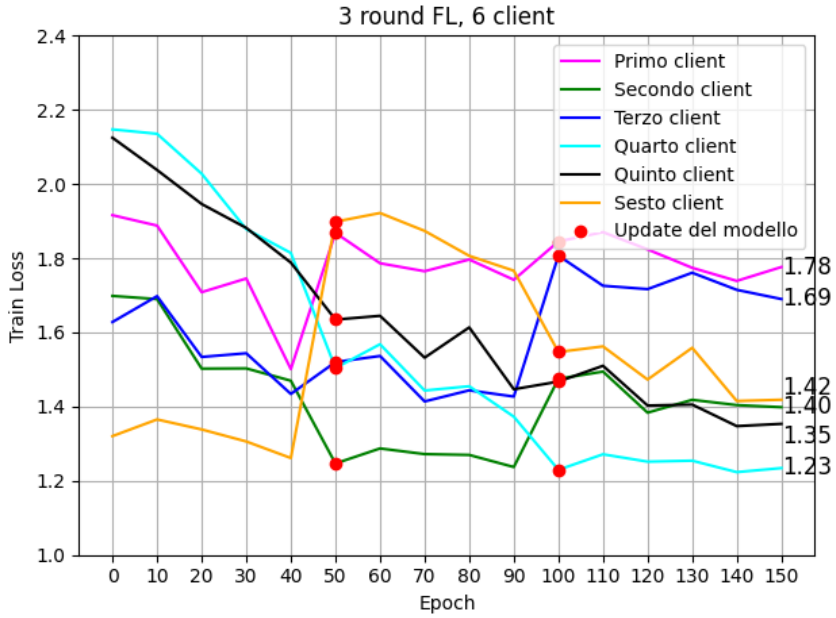


Figura 57: Funzione di loss per il modello addestrato con approccio distribuito su 6 client

In una situazione reale, dunque dove gli ospedali ed i laboratori partecipanti ad uno stesso framework di addestramento distribuito sono molteplici, si ha necessità di lavorare con dati che possono essere molto disomogenei tra di loro. Le immagini di ogni client, infatti, possono avere risoluzioni diverse, formati diversi o caratteristiche diverse. In questo caso è la strategia di aggregazione che si incarica di ristabilire la "linearità" del modello.
L'utilizzo della FedAvg, infatti, permette di realizzare un approccio robusto a distribuzioni di dati non i.i.d. (non indipendenti e identicamente distribuite) e sbilanciate. Il fatto che le immagini non siano i.i.d. significa che ogni dataset locale dei client non è rappresentativo della distribuzione dell'intera popolazione; la mancanza di equilibrio, invece, significa che ogni dataset è costituito da una diversa quantità di immagini. L'utilizzo della Federated Averaging permette alle prestazioni di non essere influenzate da questo tipo di distribuzione dei dati [35].

5 Risultati

5.1 Tool per la Segmentazione Cellulare

Precedentemente è stato individuato il modello in grado di fornire i risultati più coerenti con il ground truth (dataset di testing). A partire da questo si è realizzato un tool che permette di ottenere le maschere di immagini di colture cellulari con l'ausilio di Cellpose, rendendo quindi la procedura di segmentazione cellulare automatica ed oggettiva. Con il tool, poi, le maschere possono essere analizzate per individuare una correlazione tra informazione morfologica e di fluorescenza. Per questa fase, infatti, si utilizzano le immagini a fluorescenza ottenute dall'utilizzo di proteine che generano un'espressione di colore rosso, verde o blu, delle quali si ha a disposizione anche la versione in campo chiaro ed in campo scuro.
Il tool è suddiviso in diverse sezioni:

1. Selezione del modello pre-addestrato di Cellpose: a partire dal ground truth e dalle maschere ottenute con i diversi modelli si individua quello che fornisce il risultato più attendibile;

2. Selezione della flow threshold: individuazione del miglior valore della soglia per le immagini da analizzare;

3. Segmentazione/Inferenza: predizione con il modello Cellpose scelto per l'ottenimento delle maschere;

4. Correzione manuale delle label: utilizzo di un tool accessibile da riga di comando che permette di correggere manualmente le regioni (cellule) individuate dal modello durante la fase di segmentazione. Questo permette di avere un risultato migliore con un minore impiego di sforzo umano rispetto alla totale segmentazione manuale dell'immagine;

5. Calcolo delle informazioni morfologiche: dalle maschere ottenute si ricavano informazioni utili che descrivono le singole regioni (ovvero le cellule);

6. Calcolo delle informazioni di fluorescenza: dalle maschere si individua la posizione di ogni regione che viene replicata sulle immagini che specificano l'espressione di ogni proteina. In questo modo si possono estrarre le informazioni che descrivono la presenza di colore nell'immagine;

7. Calcolo del CSV finale: si raccolgono tutte le informazioni importanti ottenute all'interno di un file CSV che rappresenta quindi ogni maschera;

8. Calcolo istogrammi di intensità del colore: utili ad osservare la presenza di colore in relazione alla quantità di cellule;

9. Calcolo di correlazione tra informazione morfologica e di fluorescenza: utile per individuare un legame tra l'informazione di forma e di colore di ogni cellula.

In Figura 58 è riportato lo schema a blocchi che riassume la successione dei diversi step che compongono il tool realizzato per la segmentazione cellulare.

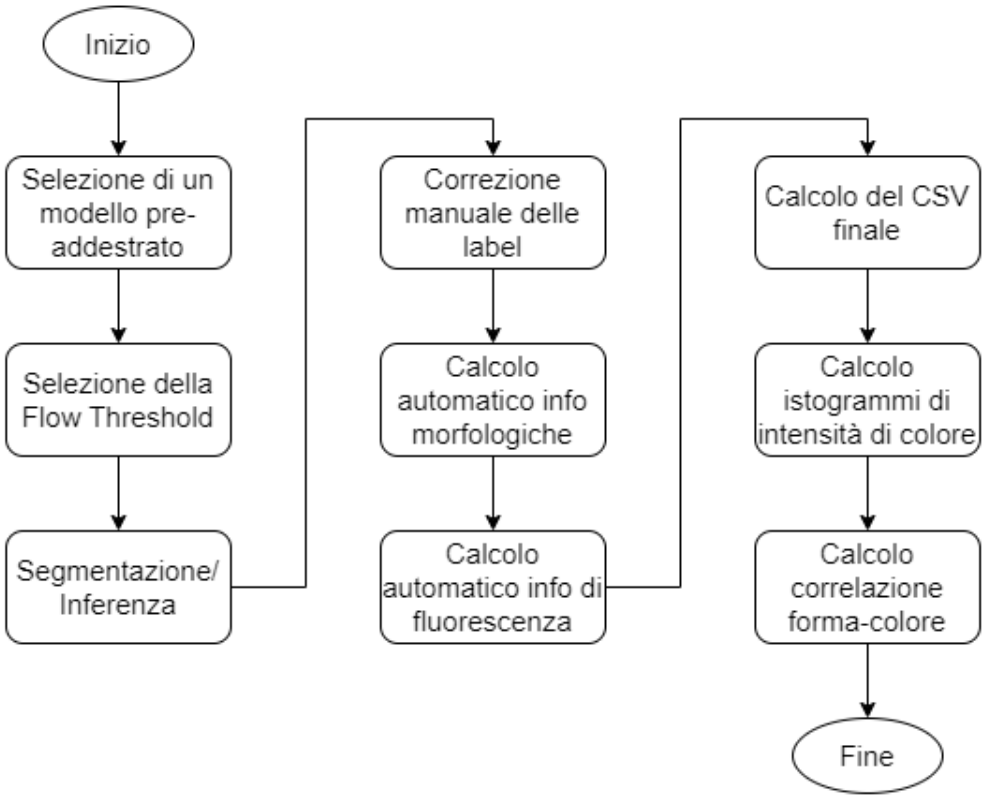
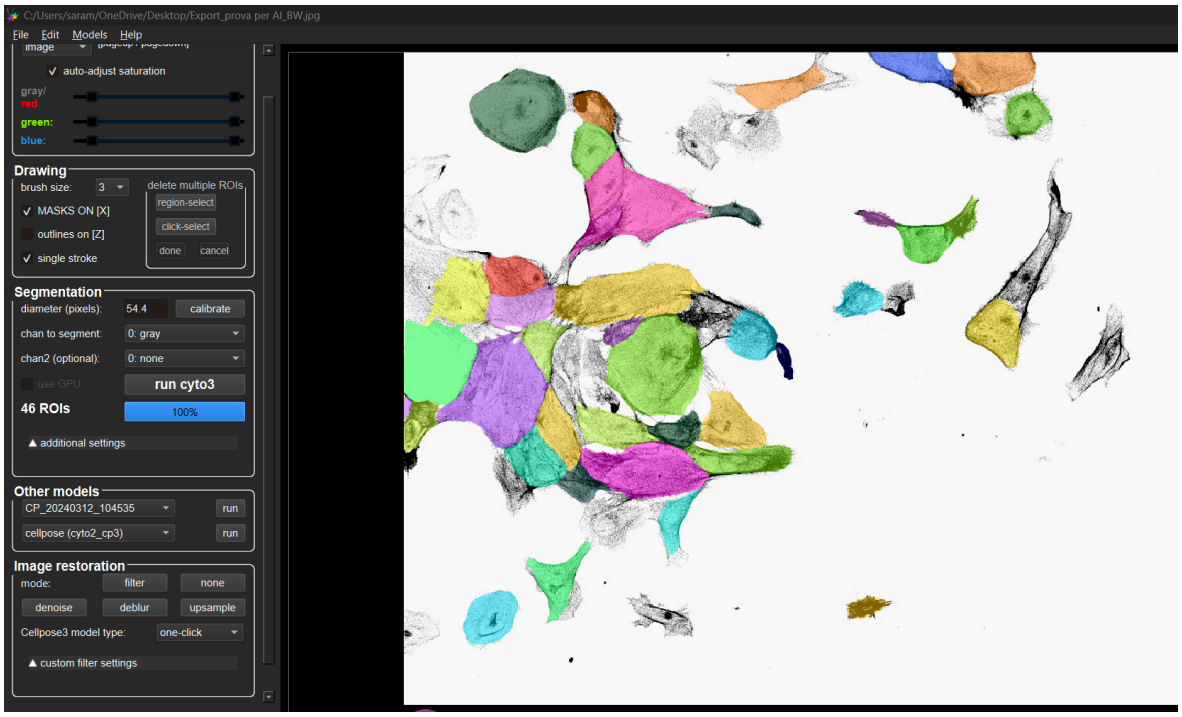


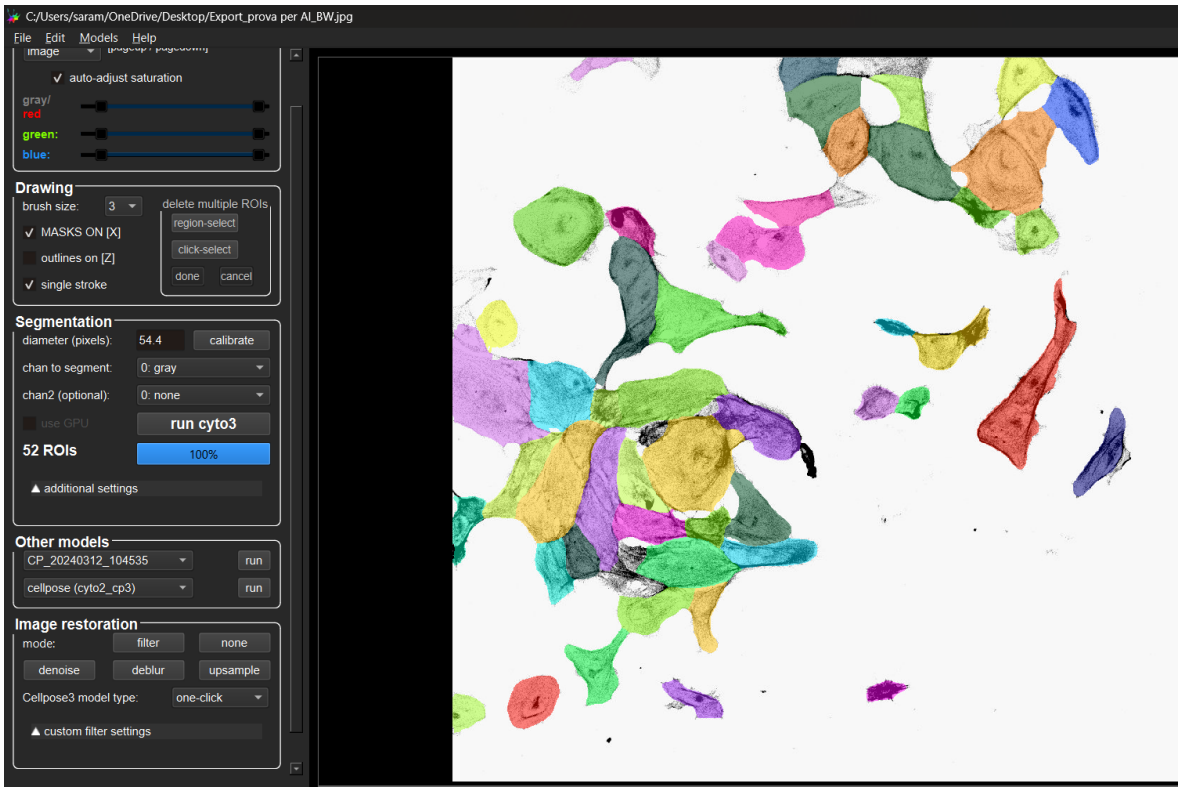
Figura 58: Schema a blocchi del tool di supporto alla segmentazione cellulare

Gli step 1-2 prevedono l'esecuzione delle medesime operazioni eseguite durante il benchmarking del modello, al fine di individuare il migliore in termini di prestazioni rispetto al dataset di testing a disposizione. In questo caso si conosce già il miglior modello per i dati forniti dal laboratorio di Medicina, ovvero Cytoplasm2.
Stabilito questo e la flow threshold che fornisce il migliore risultato in fase di predizione, si prosegue allo step 3 dove viene eseguita la segmentazione con il modello individuato sulle immagini che gli devono essere fornite. Il risultato corrisponde alle maschere relative ad ogni immagine in ingresso, le quali possono quindi essere analizzate.
Il quarto step è opzionale: se si ritiene necessario in relazione alla maschera ottenuta dagli step precedenti, è possibile sfruttare uno strumento di correzione manuale delle label. Questo permette di eliminare, correggere o aggiungere delle regioni (cellule) nella maschera fornita allo strumento stesso. In questo modo l'operatore di laboratorio è in grado di ottenere un'etichettatura ancora più precisa e corretta impiegando però uno sforzo molto minore rispetto all'esecuzione di una segmentazione interamente manuale. In Figura 59

sono mostrate le maschere prima e dopo una correzione manuale, evidenziando quindi la facilità nell'aggiungere o eliminare cellule che il modello abbia dimenticato o individuato male.



(a)



(b)

Figura 59: Step 4: correzione manuale delle label. (a) Maschera in uscita da Cytoplasm2, (b) Maschera ottenuta dopo la correzione manuale della prima

Lo step successivo, il quinto, analizza la maschera sfruttando la funzione *regionprops*() del pacchetto skimage. Questa viene utilizzata per misurare le proprietà delle regioni connesse individuate in un'immagine etichettata. Permette di estrarre una serie di caratteristiche geometriche e statistiche delle regioni. Innanzitutto fornisce il numero di regioni totali individuate e poi, per ognuna, aggiunge altre informazioni [36].
In questo caso si estraggono le seguenti: area, perimetro, coordinate x e y del centroide, lunghezza dell'asse maggiore e lunghezza dell'asse minore. Da queste si possono ottenere il valore di "Spindle Index" e di "Spindle Flag" (indicato con Spind_up3).
La funzione implementata per il calcolo di questi due parametri è definita

$$spindle\_index,\ spindle\_flag = calc\_spindle\_index\_from\_csv(csv,\ major,\ minor)$$

dove *csv* è il file CSV dove vengono memorizzare tutte le informazioni ricavate dalla maschera, *major* è l'asse maggiore e *minor* l'asse minore..
Analogamente lo step numero 6 opera sulle immagini di fluorescenza che evidenziano l'espressione di proteine che risultano nei tre colori rosso, verde e blu. In questo caso per estrarre informazioni sull'intensità di colore e sulla luminanza dell'immagine è necessario prima replicare la segmentazione sulle immagini colorate. Questo può essere eseguito con la funzione:

$$img = draw\_contour(mask,\ rgb\_image)$$

dove *img* è l'immagine colorata (*rgb_img*) fornita in ingresso, su cui sono state replicate le regioni ottenute dalla maschera (*mask*). Da *img* si possono ottenere le informazioni che esprimono la fluorescenza, in particolare l'intensità di colore, la percentuale di colore rispetto al nero e la luminanza dell'immagine.
Ottenuti tutti questi dati sulle immagini di colture cellulari a disposizione si può ricavare il file CSV finale (step 7), il cui formato è mostrato in Figura 60.

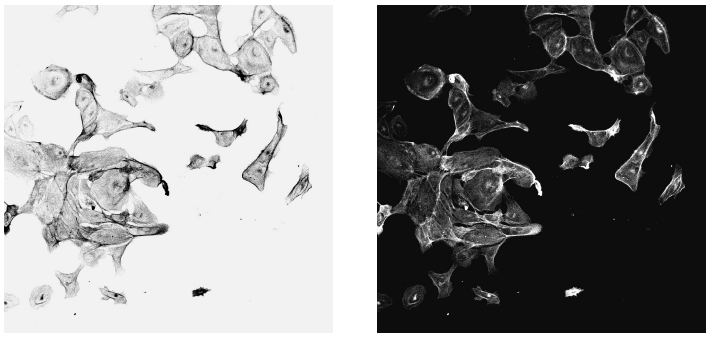
Indice	Area	Centroide (x)	Centroide (y)	Lunghezza asse maggiore	Lunghezza asse minore	Spindle Index	Spind_up 3	Intensità rosso	Intensità verde	Intensità blu	Percentuale di rosso	Percentuale di verde	Percentuale di blu	Luminanza

Figura 60: Colonne del file CSV risultante dalla completa analisi delle immagini attraverso il tool

Infine, gli step 8-9 sono utili all'analisi della fluorescenza ed alla sua correlazione con l'informazione morfologica. Questo serve all'operatore di laboratorio soprattutto per avere una panoramica sull'evoluzione della cellula in termini di forma ma anche di espressione della proteina.

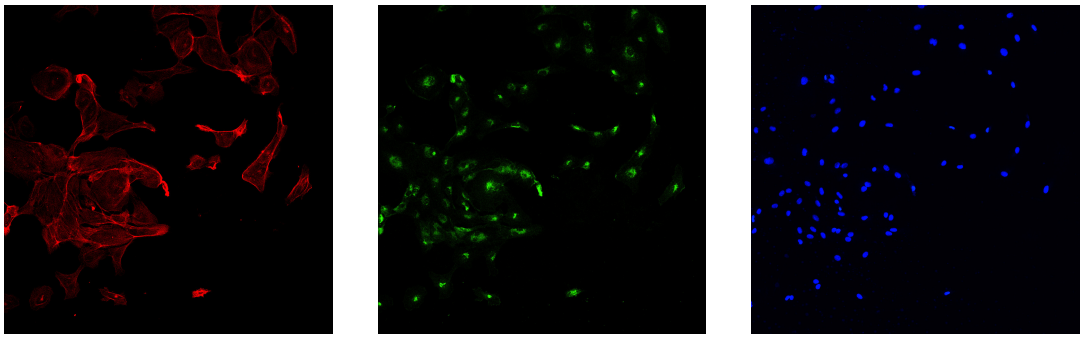
5.1.1 Analisi delle immagini di fluorescenza

Il tool viene impiegato per analizzare l'immagine di fluorescenza messa a disposizione dal laboratorio di Medicina nel dataset di testing. Di questa si hanno sia l'immagine in campo chiaro ed in campo scuro (Figura 61) che le immagini colorate, le quali esprimono la diffusione della specifica proteina applicata (Figura 62).



(a) (b)

Figura 61: Immagini in (a) campo chiaro e (b) campo scuro



(a) (b) (c)

Figura 62: Immagini di fluorescenza (a) rossa (b) verde e (c) blu

Per l'applicazione dello step 1, seguendo lo stesso ragionamento seguito per le immagini PC9 e A549, si ottiene che il modello più performante in relazione al ground truth è Cytoplasm2. Lo step 2 richiede l'analisi delle flow threshold per individuare il valore migliore in questo caso. Come per i casi precedenti, dunque, si confrontano le maschere ottenute da Cytoplasm2 con diversi valori della soglia ed il ground truth. La Tabella che riassume quest'analisi è la 11. In questo caso si individua 0.4 come valore ottimo per il parametro flow threshold.
Dunque con Cytoplasm2 e soglia impostata a 0.4 si esegue la segmentazione (step 3) e si ottengono le maschere relative alle due immagini in campo chiaro ed in campo scuro, raffigurate in Figura 63.
Si decide di lavorare con l'immagine in campo chiaro, dunque successivamente si può eseguire la correzione manuale delle label (opzionale) e poi gli step 5 e 6. Per il quinto step

Immagine Export_prova per AI_BW (campo chiaro)			
	Number of cells	**Spindle Index > 3 [%]**	**Spindle Index < 3 [%]**
Cytoplasm2 Th = 0.0	82	17.08	82.92
Cytoplasm2 Th = 0.1	8	0.0	100.0
Cytoplasm2 Th = 0.2	27	7.41	92.59
Cytoplasm2 Th = 0.3	43	11.63	88.37
Cytoplasm2 Th = 0.4	**52**	**15.38**	**84.62**
Ground truth	39	12.82	87.18
Immagine Export_prova per AI_BW (campo scuro)			
	Number of cells	**Spindle Index > 3 [%]**	**Spindle Index < 3 [%]**
Cytoplasm2 Th = 0.0	109	19.27	80.73
Cytoplasm2 Th = 0.1	10	0.0	100.0
Cytoplasm2 Th = 0.2	29	3.45	96.55
Cytoplasm2 Th = 0.3	42	4.76	95.24
Cytoplasm2 Th = 0.4	**49**	**8.16**	**91.84**
Ground truth	39	12.82	87.18

Tabella 11: Confronto ground truth - Cytoplasm2 con diversi valore di flow threshold

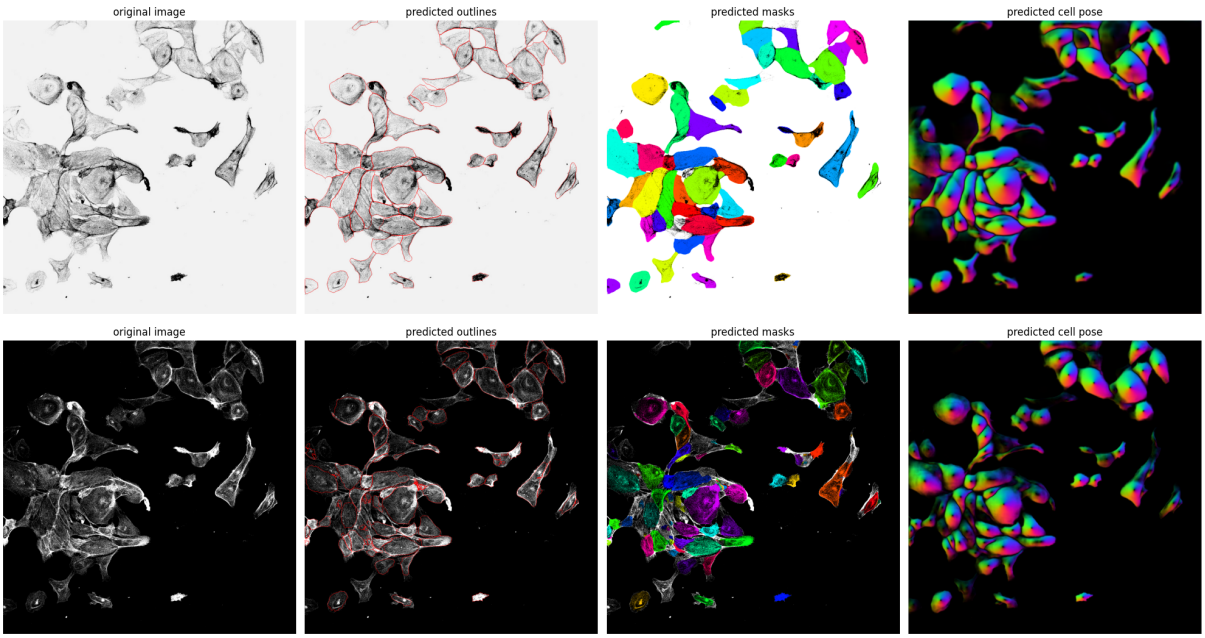


Figura 63: Maschere ottenute da Cytoplasm2 e flow threshold pari a 0.4 per le immagini in campo chiaro (in alto) ed in campo scuro (in basso)

le informazioni morfologiche riguardanti ogni regione individuata possono essere estratte con la funzione *regionprops()*. Per il sesto step, invece, è necessario trasferire le label della maschera sulle immagini che esprimono la concentrazione di proteina. In questo modo

si è in grado di identificare ogni cellula anche su di esse, permettendo di effettuare il calcolo delle informazioni di fluorescenza per ogni regione separatamente. Il risultato della replica della maschera è mostrato in Figura 64 (nell'immagine a sinistra ogni regione è colorata per evidenziarne l'estensione, mentre in quella di destra è riportato solo il numero corrispondente ad ogni regione per evidenziare la conta delle cellule).

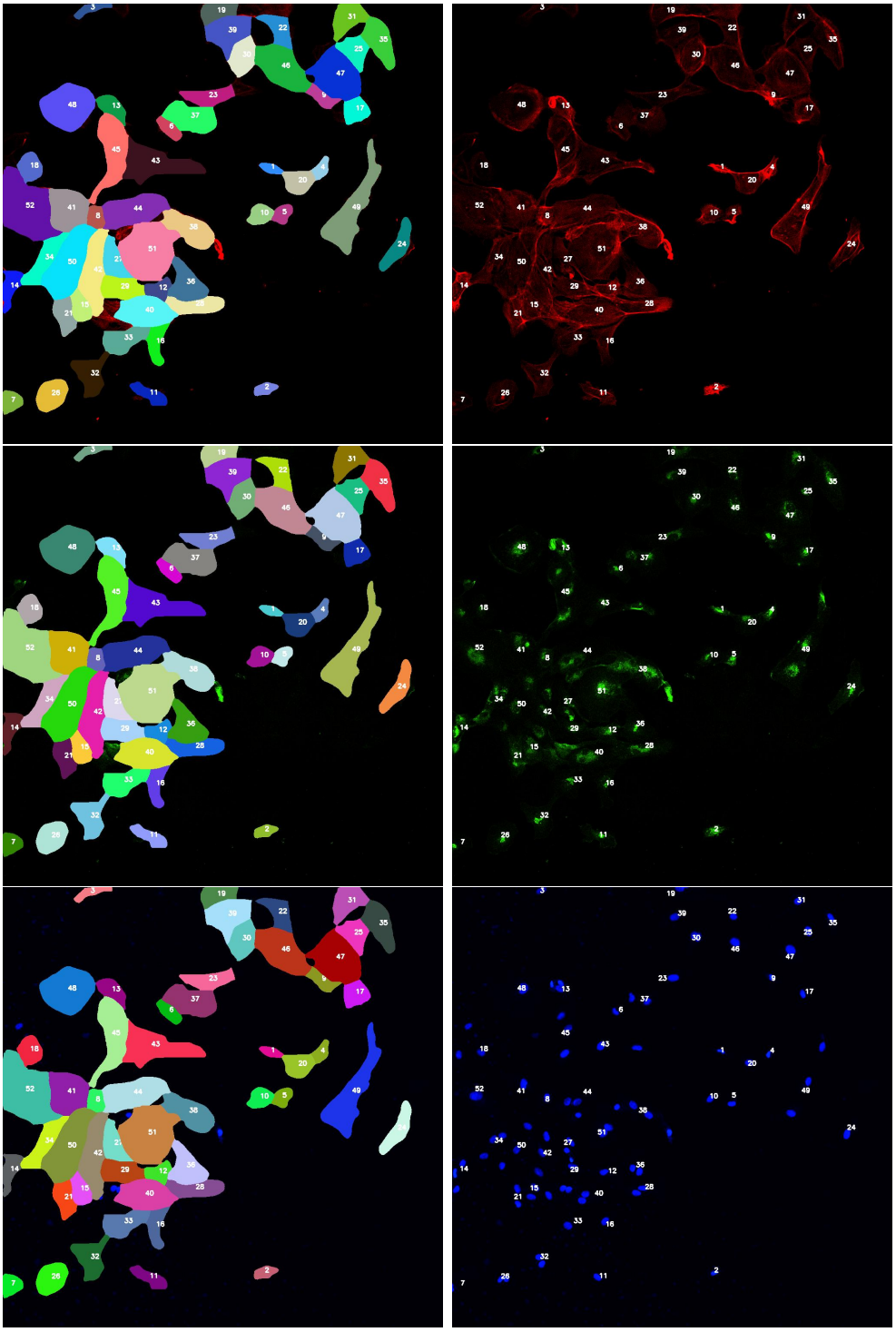


Figura 64: Applicazione della maschera alle immagini che presentano l'espressione della proteina attraverso il colore rosso (in alto), verde (al centro), blu (in basso)

Da queste è possibile quindi estrarre le informazioni di fluorescenza con le funzioni

$$intensity = color\_intensity\_calc(mask,\ rgb\_image,\ color,\ threshold)$$

$$percentage = calc\_color\_percentage(mask,\ rgb\_image,\ color,\ threshold)$$

$$luminance = calc\_luminance(mask,\ red,\ green,\ blue)$$

dove *mask* è la maschera ottenuta da Cytoplasm2, *rgb_image* (così come *red*, *green*, *blue*) è l'immagine con la fluorescenza rossa, verde o blu, *color* è pari a 0 per il colore rosso, 1 per il verde e 2 per il blu e *threshold* è una soglia che deve essere decisa sulla base dell'intensità di colore che si vuole considerare nero.
La prima funzione calcola l'intensità media del colore scelto all'interno di ogni regione: per ognuna si ottengono le coordinate dei pixel che ne determinano il contorno, si calcola la somma delle intensità su tutti i pixel del contorno (se risultano essere pixel colorati e quindi con intensità > threshold) e, infine, si divide la somma totale per il numero di pixel che costituiscono quel contorno. La soglia viene utilizzata per discriminare un pixel colorato da un pixel nero e in questo caso è impostata a 0. La formula utilizzata per il calcolo della media è:

$$region\_intensity = \frac{\sum_{i=0}^{N} pixel_i\_intensity}{N}$$

La seconda funzione calcola la percentuale di pixel colorati rispetto a quelli totali per ogni regione individuata all'interno dell'immagine. Per far questo, come prima, estrae i pixel appartenenti al contorno di ogni regione e per ognuno di essi verifica se risulta avere un valore maggiore della soglia o meno. Se il pixel ha intensità > threshold allora viene considerato colorato, altrimenti viene considerato nero. Infine la percentuale di colore rispetto al nero viene calcolata come:

$$region\_percentage = \frac{rgb\_pixels}{rgb\_pixels + black\_pixels} * 100$$

La formula è stata adattata alla discussione trovata in [37].
In questo caso la soglia impostata a 0 comporta dei risultati discordanti con le immagini da cui vengono estratti, in particolare con queste impostazioni il colore blu presenta un'intensità molto elevata (circa 90) e una percentuale pari al 100%, valori che non si riscontrano osservando l'immagine (Figura 62 in basso). Per questo motivo si è cercato di individuare la migliore soglia per questa analisi, osservando degli istogrammi per l'intensità di colore per regione, per percentuale di colore per regione e per intensità di colore

per pixel di una specifica regione. Questa operazione è stata eseguita su tutti e tre i colori e per diversi valori di soglia (0, 10, 20, 25, 30, 40, 50). In Figura 65 si riportano alcuni degli istogrammi relativi al colore blu per diversi valori di threshold.

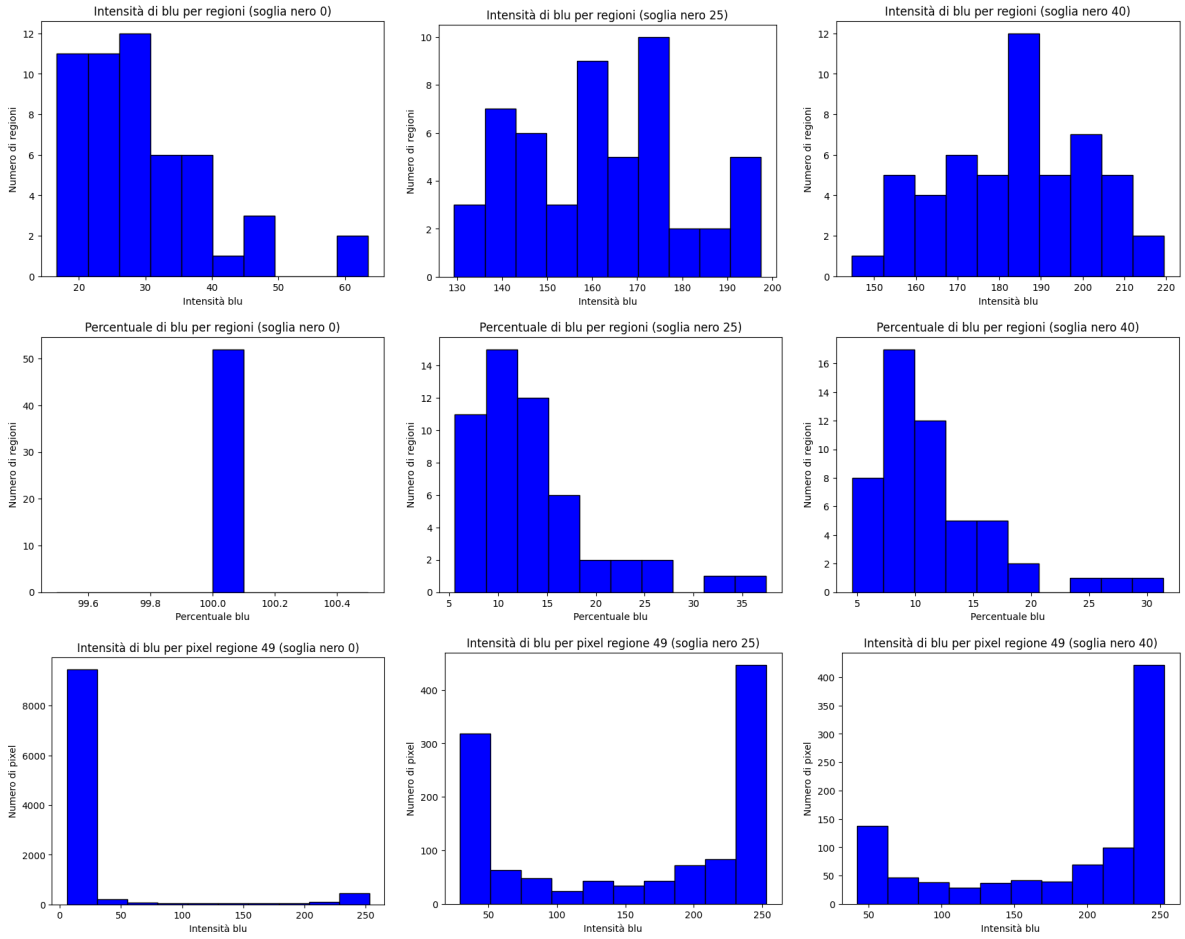


Figura 65: Istogrammi di intensità di colore blu per regione (prima riga), di percentuale per regione (seconda riga) e di intensità per pixel in una specifica regione (terza riga) per diversi valori di soglia (0, 25, 40)

Dagli istogrammi si può dedurre che con soglia impostata a 25 si riescono ad ottenere dei valori "normali" e attinenti all'immagine originale, senza influenzare invece i valori che si ottengono per i colori rosso e verde. Per ottenere il file CSV finale, dunque, si è considerata la threshold = 25 per il caso in esame.
La funzione che esegue il calcolo della luminanza, infine, richiede in ingresso tutte e tre le immagini di fluorescenza, poiché per ogni pixel è necessario estrarre l'intensità di rosso, verde e blu. Da questi valori si può ricavare la luminanza secondo la formula

$$luminance = 0.2126 * R + 0.7152 * G + 0.0722 * B$$

che può essere ottenuta dalla discussione in [38].
Come nelle due funzioni precedenti, si ricavano prima i pixel appartenenti ad ogni regione

individuata nella maschera e da quelli si calcolano le intensità di colore. La luminanza di un'intera regione viene ottenuta come media tra le luminanze di ogni suo singolo pixel.

$$region\_luminance = \frac{\sum_{i=1}^{N} pixel_i\_luminance}{N}$$

Infine, una volta ottenuto il file CSV rappresentante le immagini di fluorescenza, è possibile calcolare la correlazione tra le informazioni di tipo morfologico (spindle index) e quelle di fluorescenza (intensità di colore). Per l'analisi della correlazione si è deciso di calcolare sia l'indice di Pearson che quello di Spearman seguiti dalla rappresentazione tramite uno scatter plot.
Il coefficiente di correlazione di Pearson si calcola come

$$pearson_{coeff} = \frac{covarianza(X, Y)}{(stdv(X) * stdv(Y))}$$

ed esprime la forza della relazione lineare tra due campioni di dati. Esso restituisce un valore compreso tra -1 (correlazione negativa) e 1 (correlazione positiva), dove 0 indica assenza di correlazione. In Python per il suo calcolo si utilizza la funzione $pearsonr()$.
Il coefficiente di Spearman, invece, si ottiene come

$$spearman_{coeff} = \frac{covarianza(rank(X), rank(Y)}{(stdv(rank(X)) * stdv(rank(Y))}$$

ed esprime la correlazione tra due variabili legate da una relazione non lineare, dove esse possono avere una distribuzione non gaussiana. Anche in questo caso può valere un qualsiasi valore tra -1 e 1 con lo stesso significato del coefficiente precedente. In Python viene implementato dalla funzione $spearmanr()$ [39].
Si ottengono quindi gli scatter plot rappresentanti la correlazione tra queste informazioni, mostrati in Figura 66.

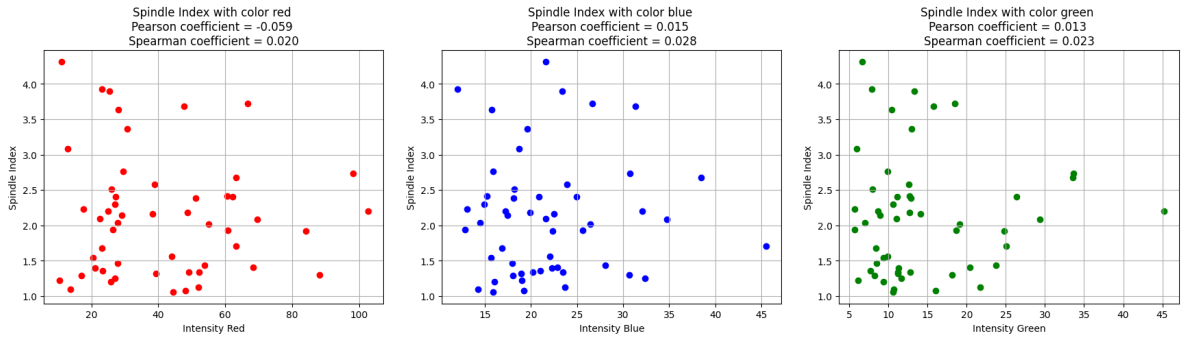


Figura 66: Rappresentazione tramite scatter plot della correlazione tra l'informazione di morfologia e di fluorescenza

Il file CSV risultante da questa analisi è presentato in Figura 67.

Indice	Area	Centroide (x)	Centroide (y)	Lunghezza asse maggiore	Lunghezza asse minore	Spindle Index	Spind_up3	Intensità rosso	Intensità verde	Intensità blu	Percentuale di rosso	Percentuale di verde	Percentuale di blu	Luminanza
1	898	383,04	623,69	56,17	20,55	2,73	0	171,09	87,23	158,56	88,53	55,23	23,16	73,30
2	1144	895,53	609,99	57,20	25,96	2,20	0	170,02	94,49	141,76	89,77	67,83	24,82	79,34
3	1305	12,67	204,06	94,45	21,89	4,31	1	41,88	47,27	135,57	36,71	19,00	26,51	55,45
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
51	13869	574,20	338,38	142,36	126,40	1,12	0	67,09	63,67	161,16	72,45	19,91	7,22	25,21
52	14354	473,66	53,92	181,17	108,17	1,67	0	46,79	64,66	129,33	43,84	11,01	9,27	24,65

Figura 67: File CSV risultante dall'analisi delle immagini a fluorescenza

5.2 Confronto tra Learning Centralizzato e Distribuito

Si esegue un confronto tra le funzioni di loss per il modello centralizzato e per quello distribuito allo scopo di determinare se le prestazioni dei modelli addestrati con approcci differenti risultano equiparabili.
I grafici della funzione di loss accostati sono riportati in Figura 68.

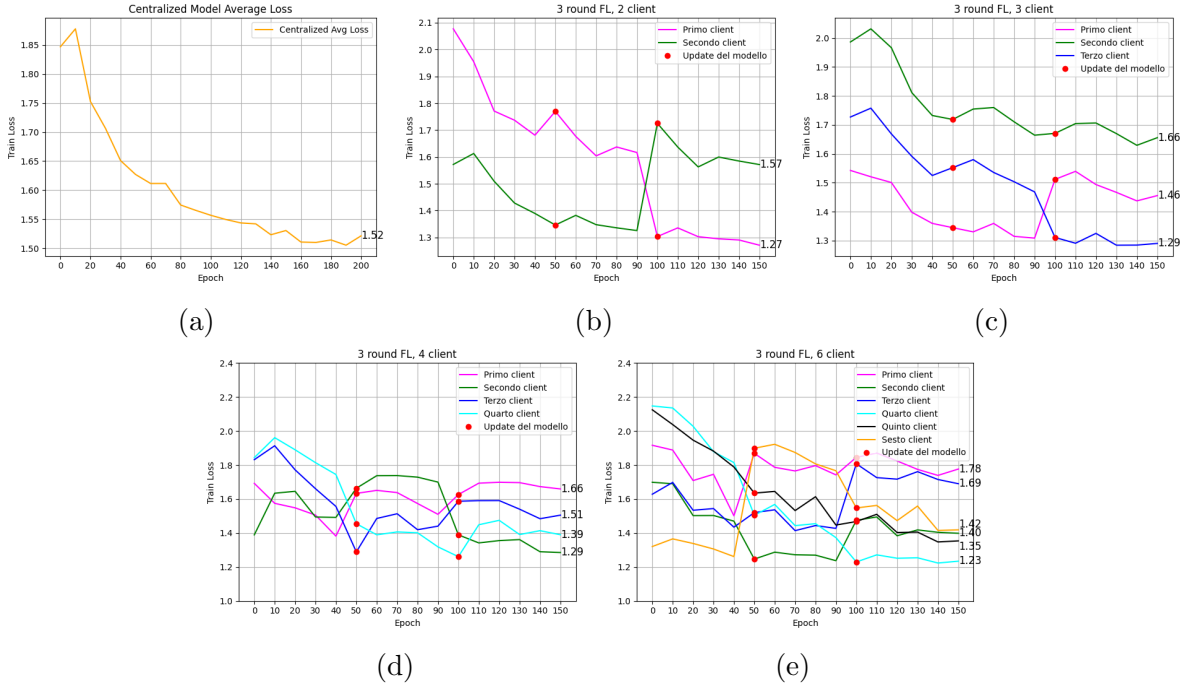


Figura 68: Grafici della funzione di loss per i modelli (a) centralized, (b) federated a 2 client, (c) federated a 3 client, (d) federated a 4 client, (e) federated a 6 client

Come prima cosa si può osservare il valore finale di loss delle diverse curve, dove per il modello federated si esegue una media tra i valori finali dei diversi client considerati.
Si ottiene dunque:

- Per il centralizzato

$$loss_{centralized} = 1.52$$

- Per il distribuito, 2 client

$$loss_{fed\_2client} = \frac{1.57 + 1.27}{2} = 1.42$$

- Per il distribuito, 3 client

$$loss_{fed\_3client} = \frac{1.66 + 1.46 + 1.29}{3} = 1.47$$

- Per il distribuito, 4 client

$$loss_{fed\_4client} = \frac{1.66 + 1.51 + 1.39 + 1.29}{4} = 1.46$$

- Per il distribuito, 6 client

$$loss_{fed\_6client} = \frac{1.78 + 1.69 + 1.42 + 1.40 + 1.35 + 1.23}{6} = 1.48$$

Questo permetterebbe già di dire che i due modelli hanno performance assimilabili.
Per effettuare un confronto più preciso delle loss finali (centralizzato e distribuito) che si ottengono dopo la fase di aggregazione dei pesi, si può ricorrere all'utilizzo della funzione per il calcolo della loss messa a disposizione da Cellpose. Tale funzione, chiamata *_loss_fn_seg*, calcola la loss come combinazione delle due funzioni Mean Squared Error (MSE) e Binary Cross Entropy (BCE) che calcolano l'errore sulla base delle label predette rispetto a quelle reali. Come già illustrato in precedenza, la loss totale si ottiene come $LOSS = MSE + BCE$ e la funzione di Cellpose che la restituisce è mostrata sotto.

```
def _loss_fn_seg(lbl, y, device):
    criterion = nn.MSELoss(reduction="mean")
    criterion2 = nn.BCEWithLogitsLoss(reduction="mean")
    veci = torch.from_numpy(lbl[:, 1:]).to(device)
    loss = criterion(y[:, :2], veci)
    loss2 = criterion2(y[:, -1], torch.from_numpy(lbl[:, 0] > 0.5).to(
                        device).float())
    loss = loss + loss2
    return loss
```

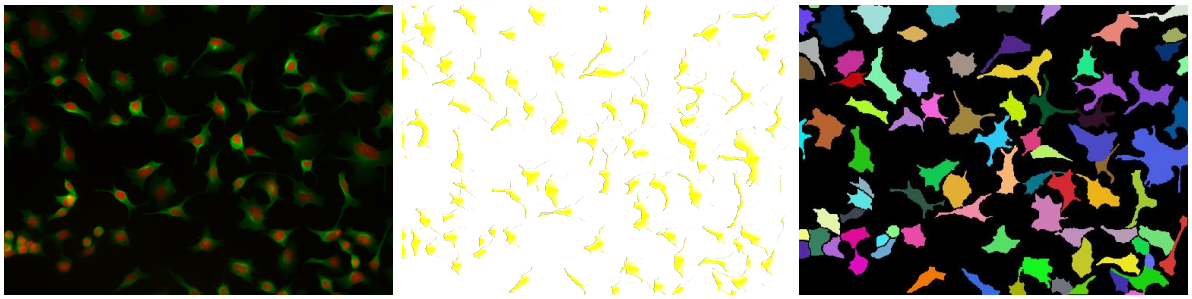
Codice 1: Funzione per il calcolo della loss in Cellpose

I parametri che vengono coinvolti al suo interno sono:

- *lbl*: le etichette reali ottenibili dal dataset di test. Deve essere un array di dimensioni [HxWx3] costituito da, rispettivamente, probabilità cellulare, flowsX e flowsY;

- y: valori predetti ottenibili in output dal modello. Deve essere un tensore di dimensioni [HxWx3] costituito rispettivamente da flowsX, flowsY e probabilità cellulare.

Per ottenere la loss globale dei modelli in esame si sfrutta il dataset di test messo a disposizione da Cellpose in [40], costituito da $n = 67$ immagini (formato PNG), ognuna associata alla corrispondente immagine delle flows (formato TIFF) ed alla rispettiva maschera (formato PNG). Un esempio di immagine, flows e maschera appartenente al dataset di test utilizzato è riportato in Figura 69.
Ogni immagine appartenente al dataset viene fornita al modello da studiare per poterne ottenere i valori da esso predetti. Successivamente, insieme ai valori reali contenuti nel



(a) (b) (c)

Figura 69: Immagini del dataset di test, corrispondenti a (a) immagine originale, (b) andamento del gradiente (flows), (c) maschera

dataset, si esegue il calcolo della loss attraverso l'utilizzo della funzione. Quest'ultima restituisce un valore di loss per ogni immagine fornitagli in ingresso, di conseguenza per ottenere il valore di loss globale è necessario calcolare una media tra tutti i 67 valori. Questo procedimento viene compiuto tramite una funzione implementata e definita

$$loss = calc\_loss(model\_path, image\_path, mask\_path, flow\_path)$$

dove $model\_path$ è il percorso dove si trovano i pesi del modello che si vuole studiare in termini di loss, $image\_path$ è il percorso dell'immagine di test, $mask\_path$ quello della maschera di test e $flow\_path$ il percorso dell'immagine rappresentante il gradiente dell'immagine di test. All'interno di questa funzione, come prima cosa, si esegue l'importazione del modello con cui viene effettuata la predizione sull'immagine fornita in ingresso. Da questa si estraggono poi le informazioni di $flowX$, $flowY$, $cellprob$ per poter definire y. Successivamente si estraggono le stesse informazioni dalle immagini del dataset, con cui è possibile definire lbl. Ottenuti il tensore e l'array necessari, si possono dare in input alla funzione $\_loss\_fn\_seg()$ che esegue il calcolo della funzione di loss per quell'immagine.
Con ogni valore di loss restituito per ognuna delle 67 immagini si calcola il valore medio che andrà a rappresentare il valore di loss globale per quello specifico modello sotto analisi. In particolare quindi si ha:

$$loss_{global} = \frac{\sum_{i=1}^{67} loss_i}{67}$$

Questo ragionamento viene applicato a tutti i modelli a disposizione (modello centralizzato e distribuito a 2, 3, 4 e 6 client) per ottenere i valori di loss globale da confrontare.
I valori utili al confronto delle performance dei due modelli sono presentati in Tabella 12. Dal confronto tra i valori ottenuti si può dedurre che le prestazioni dei modelli sono confrontabili, nonostante uno sia addestrato in modo centralizzato e l'altro in modo distribuito.

Numero di client	Loss modello centralizzato	Loss modello distribuito
2	1.493	1.502
3	1.493	1.496
4	1.493	1.498
6	1.493	1.501

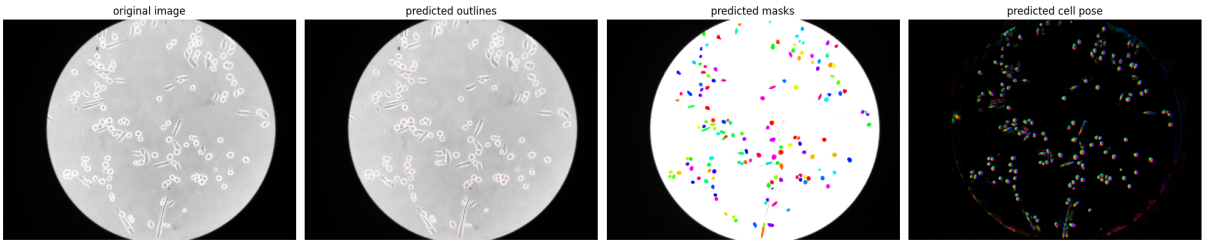
Tabella 12: Tabella di confronto delle loss globali

Un ulteriore metodo che permette di confrontare le performance dei modelli, consiste nel comparare il risultato che si ottiene in seguito ad una predizione su una stessa immagine ad essi sconosciuta. Per fare questo, dunque, viene fornita l'immagine "PC9 P1 20X 48h_1" al centralized e ai federated model per ottenere in uscita le rispettive maschere predette. I risultati possono essere osservati in Figura 70.
Dall'immagine si può osservare che le maschere ottenute dai diversi modelli possono considerarsi confrontabili. Esse infatti differiscono solamente per alcune predizioni di cellule di forma più oblunga, ma per il resto delle cellule individuate, le performance possono considerarsi buone per tutti i modelli considerati, se confrontate al ground truth (come già visto nell'implementazione del tool per la segmentazione cellulare). Questa conclusione può essere osservata anche dalla Tabella 13 che evidenzia la somiglianza tra le caratteristiche estratte dalle diverse maschere rispetto al ground truth.

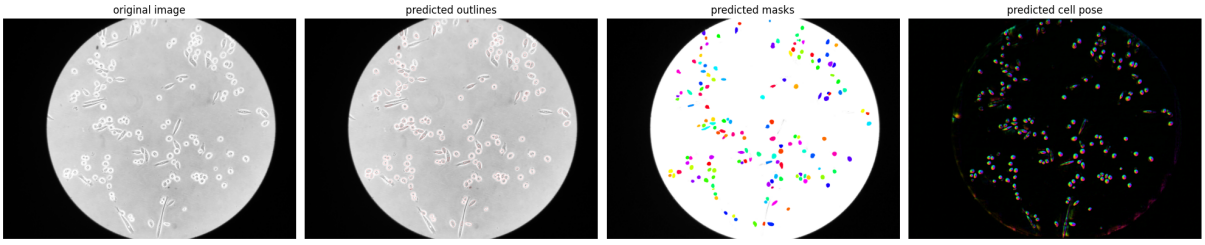
Immagine PC9 P1 20X 48h 1			
	Number of cells	**Spindle Index** > 3 [%]	**Spindle Index** < 3 [%]
Ground truth	155	12.26	87.74
Centralized Model	149	2.01	97.99
Federated Model (2 clients)	151	1.32	98.68
Federated Model (3 clients)	145	1.38	98.62
Federated Model (4 clients)	160	3.75	96.25
Federated Model (6 clients)	146	2.05	97.95

Tabella 13: Confronto tra le maschere ottenute con i modelli Centralized e Federated e la ground truth

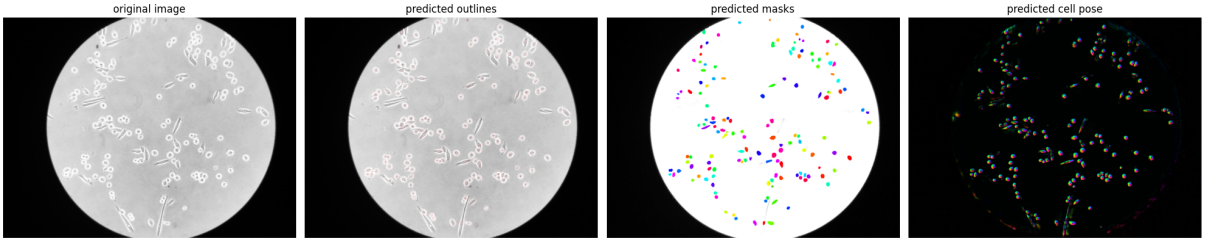
Questa ulteriore conferma permette di concludere che le performance ottenibili con l'addestramento di tipo centralizzato e con l'addestramento distribuito sono equiparabili, rendendo così possibile scegliere l'uno rispetto all'altro in caso di particolari necessità.



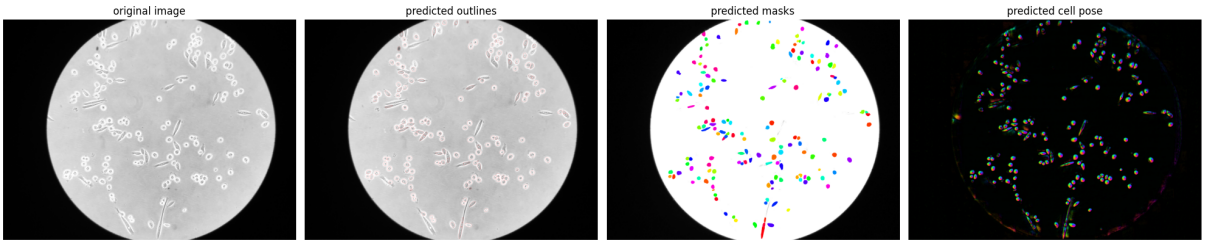
(a)



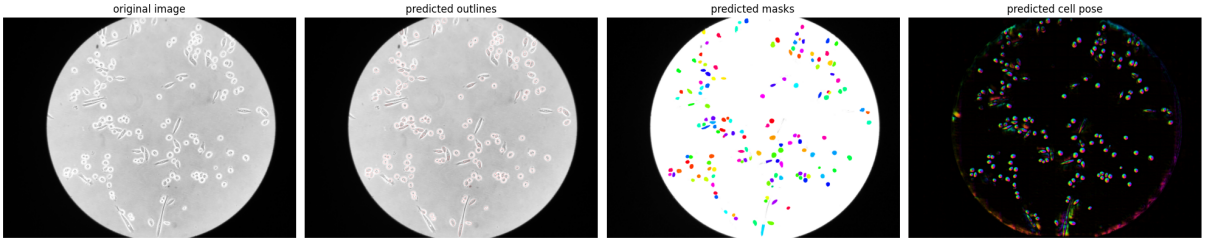
(b)



(c)



(d)



(e)

Figura 70: Maschera ottenuta dal (a) Centralized Model, (b) Federated Model con 2 client, (c) Federated Model con 3 client, (d) Federated Model con 4 client, (e) Federated Model con 6 client

Conclusione

In questa tesi, è stato affrontato il problema della segmentazione di immagini di colture cellulari, ai fini di individuare una tecnica che permettesse di rendere la procedura di labeling manuale delle immagini più automatica ed oggettiva, andando così a diminuire l'onere umano richiesto per il task di segmentazione. Inoltre, si è cercato di superare le limitazioni poste dall'approccio centralizzato. In particolare, una delle principali difficoltà riscontrate è la complessità nel condividere i dati tra i diversi laboratori, operazione necessaria per addestrare efficacemente i modelli. In aggiunta, l'addestramento richiede una grande quantità di immagini etichettate per il fine-tuning dei modelli di segmentazione cellulare. L'approccio di Federated Learning offre vantaggi significativi in questo contesto, come una condivisione semplificata dei dati, la possibilità di apprendimento continuo e una maggiore tutela della privacy, poiché i dati rimangono localizzati presso i singoli laboratori.
Il lavoro svolto ha consentito di realizzare gli obiettivi presentati. In particolare, si è giunti ai seguenti risultati:

- La realizzazione di un tool di supporto alla segmentazione cellulare, il quale impiega un modello di rete neurale che meglio performa sulle immagini a disposizione. Il modello è stato selezionato come risultato di un confronto tra quelli implementati in ZeroCostDL4Mic; individuato Cellpose come migliore, sono state confrontate le sue reti pre-addestrate, allo scopo di individuare quella che fornisce la maschera più coerente al ground truth. Il tool permette, dunque, di compiere il task di segmentazione su immagini di colture cellulari acquisite da microscopio, richiedendo minore sforzo umano rispetto all'intera segmentazione manuale delle immagini. Inoltre, permette di analizzare le maschere che si ottengono in uscita dalla rete, fornendo ulteriori informazioni sulle cellule individuate all'interno delle immagini, utili agli operatori di laboratorio per raccogliere dati riguardo alle colture analizzate;

- La risoluzione delle criticità poste dall'utilizzo dell'addestramento centralizzato tramite l'impiego di addestramento distribuito, il quale permette di realizzare il task di segmentazione cellulare senza la necessità di condividere le immagini tra diversi laboratori, e dunque i dati sensibili dei pazienti. Inoltre l'utilizzo del FL riduce notevolmente il carico di lavoro necessario per l'intero addestramento del modello selezionato, poiché lo distribuisce sui diversi nodi connessi. Di conseguenza, si dimostra il fatto che i due diversi approcci considerati (centralizzato e distribuito) risultano avere prestazioni paragonabili, rendendo quindi possibile e sensata la sostituzione di uno rispetto all'altro quando si vogliono ottenere i vantaggi portati dal Federated Learning.

In particolare, l'analisi delle performance dei modelli addestrati con tecnica distribuita permette di dedurre che l'incremento del numero di client non ha influenza sulle prestazioni del modello. Bisogna però tenere conto di uno svantaggio del FL che consiste in un training meno "stabile" e che richiede più tempo rispetto alla sua controparte centralizzata.
Per quanto riguarda gli sviluppi futuri, come prima cosa si potrebbe cercare di accumulare un numero di immagini soddisfacente per eseguire il fine tuning del modello Cellpose alle immagini di colture cellulari che si vogliono analizzare nel dettaglio. In questo modo, ri-addestrando il modello con un dataset dedicato, si riuscirebbero ad ottenere prestazioni ancora migliori nel task di segmentazione cellulare, il che potrebbe significare in una rimozione totale della necessità di intervento dell'uomo. Nell'ambito dell'addestramento distribuito, invece, poiché l'analisi qui portata a termine è stata limitata dall'utilizzo dei dataset non molto estesi messi a disposizione da Cellpose (Cytoplasm2), le simulazioni di FL sono state eseguite fino ad un massimo di 6 client e di 3 round. Uno possibile sviluppo di questo studio potrebbe prevedere l'impiego di un dataset più grande, così da consentire l'aumento dei parametri e dando quindi la possibilità di studiare l'evoluzione delle prestazioni del modello all'aumento del numero di round e del numero di client.

Riferimenti bibliografici

[1] Ongsulee, Pariwat. "Artificial intelligence, machine learning and deep learning." 2017 15th international conference on ICT and knowledge engineering (ICT&KE). IEEE, 2017.

[2] Janiesch, Christian, Patrick Zschech, and Kai Heinrich. "Machine learning and deep learning." Electronic Markets 31.3 (2021): 685-695.

[3] Sah, Shagan. "Machine learning: a review of learning types." (2020).

[4] Montesinos López, Osval Antonio, Abelardo Montesinos López, and Jose Crossa. "Fundamentals of artificial neural networks and deep learning." Multivariate statistical machine learning methods for genomic prediction. Cham: Springer International Publishing, 2022. 379-425.

[5] Shinde, Pramila P., and Seema Shah. "A review of machine learning and deep learning applications." 2018 Fourth international conference on computing communication control and automation (ICCUBEA). IEEE, 2018.

[6] Li, Zewen, et al. "A survey of convolutional neural networks: analysis, applications, and prospects." IEEE transactions on neural networks and learning systems 33.12 (2021): 6999-7019.

[7] O'shea, Keiron, and Ryan Nash. "An introduction to convolutional neural networks." arXiv preprint arXiv:1511.08458 (2015).

[8] Guo, Yanming, et al. "A review of semantic segmentation using deep neural networks." International journal of multimedia information retrieval 7 (2018): 87-93.

[9] What is federated learning? (s.d.). Flower. https://flower.ai/docs/framework/tutorial-series-what-is-federated-learning.html (23/08/2024)

[10] AbdulRahman, Sawsan, et al. "A survey on federated learning: The journey from centralized to distributed on-site learning and beyond." IEEE Internet of Things Journal 8.7 (2020): 5476-5497.

[11] Wen, Jie, et al. "A survey on federated learning: challenges and applications." International Journal of Machine Learning and Cybernetics 14.2 (2023): 513-535.

[12] Xing, Fuyong, et al. "Deep learning in microscopy image analysis: A survey." IEEE transactions on neural networks and learning systems 29.10 (2017): 4550-4568.

[13] Rivenson, Yair, et al. "Deep learning microscopy." Optica 4.11 (2017): 1437-1443.

[14] Chalfoun, Joe, et al. "FogBank: a single cell segmentation across multiple cell lines and image modalities." Bmc Bioinformatics 15 (2014): 1-12.

[15] Liu, Ping, et al. "Software Tools for 2D Cell Segmentation." Cells 13.4 (2024): 352.

[16] Carpenter, Anne E., et al. "CellProfiler: image analysis software for identifying and quantifying cell phenotypes." Genome biology 7 (2006): 1-11.

[17] von Chamier, Lucas, et al. "Democratising deep learning for microscopy with ZeroCostDL4Mic." Nature communications 12.1 (2021): 2276.

[18] GitHub - HenriquesLab/ZeroCostDL4Mic: ZeroCostDL4Mic: A Google Colab based no-cost toolbox to explore Deep-Learning in Microscopy. (s.d.). GitHub. https://github.com/HenriquesLab/ZeroCostDL4Mic (28/08/2024)

[19] Home. (s.d.). GitHub. https://github.com/HenriquesLab/ZeroCostDL4Mic/wiki (28/07/2024)

[20] Flower: A friendly federated learning framework. (s.d.). Flower: A Friendly Federated Learning Framework. https://flower.ai/ (02/09/2024)

[21] Beutel, Daniel J., et al. "Flower: A friendly federated learning framework." (2022).

[22] Human-Centric federated learning. (s.d.). ELLIS Alicante. https://ellisalicante.org/federated-learning (05/09/2024)

[23] Koo, Vincent, et al. "Novel in vitro assays for the characterization of EMT in tumourigenesis." Analytical Cellular Pathology 32.1-2 (2010): 67-76.

[24] Falk, Thorsten, et al. "U-Net: deep learning for cell counting, detection, and morphometry." Nature methods 16.1 (2019): 67-70.

[25] Skimage.measure — skimage 0.24.0 documentation. (s.d.). scikit-image: Image processing in Python - scikit-image. https://scikit-image.org/docs/stable /api/skimage.measure.html#skimage.measure.label (04/10/2024)

[26] Schmidt, Uwe, et al. "Cell detection with star-convex polygons." Medical Image Computing and Computer Assisted Intervention–MICCAI 2018: 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part II 11. Springer International Publishing, 2018.

[27] Mandal, Soham, and Virginie Uhlmann. "Splinedist: Automated cell segmentation with spline curves." 2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI). IEEE, 2021.

[28] Lalit, Manan, Pavel Tomancak, and Florian Jug. "Embedding-based instance segmentation in microscopy." Medical Imaging with Deep Learning. PMLR, 2021.

[29] Stringer, Carsen, et al. "Cellpose: a generalist algorithm for cellular segmentation." Nature methods 18.1 (2021): 100-106.

[30] Lee, Kwanyoung, Hyungjo Byun, and Hyunjung Shim. "Cell Segmentation in Multi-modality High-Resolution Microscopy Images with Cellpose." Competitions in Neural Information Processing Systems. PMLR, 2023.

[31] Edlund, Christoffer, et al. "LIVECell—A large-scale dataset for label-free live cell segmentation." Nature methods 18.9 (2021): 1038-1045.

[32] Google colab. (s.d.). Google Colab. https://colab.research.google.com/github /adap/flower/blob/main/doc/source/tutorial-series-get-started-with-flower-pytorch.ipynb (12/09/2024)

[33] Mean squared error (MSE) loss function. (s.d.). CodingNomads Learn to code, anywhere. https://codingnomads.com/mean-squared-error-loss-function (18/09/2024)

[34] Binary cross entropy: Where to use log loss in model monitoring. (s.d.). Arize AI. https://arize.com/blog-course/binary-cross-entropy-log-loss/ (18/09/2024)

[35] McMahan, Brendan, et al. "Communication-efficient learning of deep networks from decentralized data." *Artificial intelligence and statistics*. PMLR, 2017.

[36] Skimage.measure — skimage 0.24.0 documentation. (s.d.-b). scikit-image: Image processing in Python - scikit-image. https://scikit-image.org/docs/stable/api/skimage.measure.html#skimage.measure.regionprops (04/10/2024)

[37] Quora. (n.d.). Software and Applications: How can I calculate the amount of pixels and percentages of a specific color within an image https://www.quora.com /Software-and-Applications-How-can-I-calculate-the-amount-of-pixels-and-percentages-of-a-specific-color-within-an-image (04/10/2024)

[38] Formula to determine perceived brightness of RGB color. (s.d.). Stack Overflow. https://stackoverflow.com/questions/596216/formula-to-determine perceived-brightness-of-rgb-color (04/10/2024)

[39] Come calcolare la correlazione tra variabili in Python - NetAi. (s.d.). NetAi. https://netai.it/come-calcolare-la-correlazione-tra-variabili-in-python/ (14/09/2024)

[40] Cellpose. (s.d.). cellpose. https://www.cellpose.org/ (29/09/2024)