DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Università degli Studi di Padova

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA MAGISTRALE IN
BIOINGEGNERIA DELLA RIABILITAZIONE

D.M 270/2004

SVILUPPO DI UNA RETE NEURALE IN SUPPORTO ALLA DIAGNOSI DELLA MALATTIA DI HIRSCHSPRUNG

Relatore: Prof.ssa Maria Francesca Uccheddu

Laureanda:
Alessia Magoni

ANNO ACCADEMICO 2022 – 2023

13 luglio 2023

INDICE

INDICE FIGURE

INTRODUZIONE

La malattia di Hirschsprung (HD) è una patologia congenita dell'intestino distale causata dall'assenza di cellule gangliari nei plessi mioenterico di Auerbach e sottomucoso di Meissner che hanno la funzione di generare i movimenti dell'intestino. Questa malattia si manifesta clinicamente in età molto precoce, generalmente nei primi giorni di vita, con assenza di peristalsi nel tratto agangliare e conseguente ritardo nell'emissione del meconio.
Per una diagnosi e un trattamento corretti è necessaria la collaborazione di chirurghi e patologi esperti.

Dall'esame istologico dell'intestino del paziente è possibile individuare tre aree distinte:

- Zona patologica agangliare: quest'area è definita dall'assenza di cellule gangliari e dalla presenza di nervi ipertrofici. In questa regione, mancano le cellule gangliari responsabili dei movimenti dell'intestino, mentre i nervi possono risultare ipertrofici, ovvero con un aumento della loro dimensione.
- Zona di transizione: quest'area è caratterizzata dalla presenza simultanea di nervi ipertrofici e cellule gangliari. Si tratta di una regione in cui la transizione tra la zona patologica agangliare e l'area prossimale sana avviene gradualmente, mostrando una combinazione di caratteristiche di entrambe le zone.
- Zona prossimale sana: quest'area rappresenta la porzione sana dell'intestino prossimale, dove sono presenti cellule gangliari e nervi normali. In questa regione, le cellule gangliari svolgono la funzione di generare i movimenti intestinali in modo regolare e i nervi presentano un aspetto normale senza ipertrofia.

La valutazione accurata della densità delle cellule gangliari e l'ipertrofia dei nervi nelle sezioni istologiche sono la chiave per pianificare un intervento chirurgico corretto, necessario per rimuovere l'intestino non funzionante.
Tale analisi può essere impegnativa per il patologo, dispendiosa in termini di tempo e soggetta a errori, in particolare nei centri con un basso volume di pazienti.

Lo scopo di questa ricerca consiste nello sviluppo di un algoritmo capace di effettuare una segmentazione accurata delle immagini delle diverse zone precedentemente

menzionate. Tale algoritmo sarà implementato come uno strumento di supporto per i patologi nella diagnosi della malattia di Hirschsprung.

L'obiettivo è quello di accorciare i tempi nell'analisi delle immagini in quanto l'intelligenza artificiale si occuperebbe dell'individuazione della presenza (o assenza) dei gangli, lasciando all'anatomopatologo solo l'analisi delle immagini dubbie relative alla zona di transizione.

Tuttavia, una limitazione significativa per poter addestrare un algoritmo di intelligenza artificiale è la necessità di avere un dataset di partenza molto ampio, cosa difficile da ottenere per le immagini mediche, in particolare se si tratta di una malattia rara come quella di Hirschsprung. L'aumento dei dati, tramite *Data Augmentation*, è quindi fondamentale per la realizzazione del nostro scopo.

Al giorno d'oggi, ci sono diversi algoritmi di Machine Learning disponibili per la segmentazione automatica delle immagini, in particolare le reti neurali convoluzionali (CNN) sono risultate essere tra le più efficaci. Il modello scelto per questa tesi è la rete U-net, così chiamata per la sua forma grafica a "U". Questa rete utilizza una strategia di "compressione" dell'immagine in ingresso, estrae le caratteristiche rilevanti (features) e successivamente esegue una "decompressione" per ottenere una segmentazione finale pixel per pixel. L'architettura a forma di "U" consente alla rete di apprendere informazioni dettagliate sia a livello locale che globale, contribuendo così alla precisione e alla qualità della segmentazione dell'immagine istologica.

L'obiettivo principale è addestrare la rete neurale affinché sia in grado di riconoscere e classificare correttamente i dati di ingresso, che nel nostro caso sono rappresentati dalle immagini istologiche. La rete neurale sarà addestrata per associare queste immagini a classi predefinite, consentendo di identificare e distinguere automaticamente le caratteristiche di interesse presenti nelle immagini.

1. BACKGROUND

In questo capitolo, viene presentata la problematica clinica relativa alla malattia di Hirschsprung, inclusa la diagnosi e i metodi di trattamento associati.

Successivamente, viene fornito uno stato dell'arte sull'analisi delle immagini mediche, con particolare accentuazione sulle reti neurali. Verrà descritto il modello proposto per l'analisi delle immagini istologiche, basato su una rete neurale convoluzionale (CNN) con architettura U-Net.

Infine, nell'ultima sezione del capitolo, verranno affrontati i limiti e le difficoltà attuali che possono essere riscontrati nella pratica clinica per quanto riguarda la malattia di Hirschsprung.

1.1 BACKGROUND CLINICO

1.1.1 SINTOMI CLINICI

Per garantire la corretta miscelazione e propulsione dei contenuti durante la digestione, l'assorbimento e l'escrezione, l'intestino umano presenta un ricco repertorio di movimenti coordinati del suo apparato muscolare.

È il sistema nervoso enterico che controlla il tratto intestinale, compreso il pancreas e la cistifellea, tramite i motoneuroni enterici che vanno ad agire sulla muscolatura liscia, i vasi sanguigni e l'attività secretoria. In particolare, è presente il plesso mienterico (di Auerbach) nella tonaca muscolare e il plesso sottomucoso (di Meissner) nella tonaca sottomucosa. Il plesso di Auerbach è costituito da un intreccio di fibre nervose e da cellule gangliari e l'input è sia parasimpatico che simpatico (sono presenti i gangli dell'innervazione parasimpatica, ma solo le fibre simpatiche raggiungono anche il plesso); il plesso di Meissner invece ha solo fibre parasimpatiche.

La malattia di Hirschsprung (HSCR) è il disturbo più comune della motilità intestinale congenita ed è caratterizzata da una mancanza di cellule nervose gangliari (aganglionosi) in una lunghezza variabile dell'intestino distale. La funzione di queste cellule è quella di provocare il movimento dell'intestino (peristalsi), che nel colon permette alle feci di essere evacuate. [1, 2]

La causa di questa malattia è l'arresto nello sviluppo e nella migrazione delle cellule nervose gangliari. Queste cellule migrano dal tratto iniziale dell'apparato digerente (esofago) e raggiungono progressivamente gli altri tratti dell'intestino, fino alla parte più distante, il sigma-retto. Più precoce è l'arresto di migrazione durante la fase di sviluppo embrionale e più lunga sarà la porzione d'intestino priva di cellule nervose. [3]

Tra le conseguenze, oltre all'assenza di cellule gangliari nello strato muscolare e nella sottomucosa dell'intestino distale, troviamo spasmi persistenti dell'intestino distale, ipertrofia compensatoria ed espansione dell'intestino prossimale con sintomi quali costipazione, distensione addominale e ostruzione intestinale.

Nella più comune, la "classica" forma di HSCR, l'aganglionosi è limitata alla regione rectosigmoide ed è indicata come malattia del "segmento corto". Questa variante rappresenta oltre l'80% dei casi. Nei restanti casi, l'aganglionosi del colon è più estesa e

può coinvolgere l'intestino tenue distale. L'aganglionosi enterica totale è sia rara che associata ad alta morbilità e mortalità. [1]

Questa malattia si manifesta clinicamente in età molto precoce, generalmente nei primi giorni di vita, e influenza seriamente la qualità della vita e la crescita dei bambini. [3]

L'incidenza dell'HSCR è stimata a 1 su 5000 nati vivi ed è stato riconosciuto che i maschi sono più colpiti delle femmine con un rapporto maschio-femmina di 4:1 [2].

Di tutti i casi di HSCR l'80-90% producono i sintomi clinici e sono diagnosticati nel periodo neonatale. Il passaggio ritardato del meconio è il sintomo cardinale in neonati con HSCR; infatti, oltre il 90% dei pazienti affetti non riesce a passare il meconio nelle prime 24 ore di vita. La presentazione usuale di HSCR nel periodo neonatale è con costipazione, distensione addominale, ostruzione intestinale e vomito bilare.

Alcuni bambini non si ostruiscono nel periodo neonatale ma il problema si presenta più avanti nell'infanzia o persino nell'adolescenza o nell'età adulta con stitichezza severa, distensione addominale cronica e ritardo nella crescita.

Circa un terzo dei bambini con HSCR si presenta con diarrea. La diarrea nell'HSCR è sempre un sintomo di enterocolite, che è la causa principale di morbilità e mortalità. I sintomi classici dell'enterocolite associata a Hirschsprung (HAEC) includono distensione addominale, febbre e diarrea. Tuttavia, esiste un ampio spettro clinico con il quale possono presentarsi neonati affetti da HAEC e altri segnali o sintomi possono includere vomito, sanguinamento rettale, letargia, feci molli e ostruzione. [7] L'HAEC può verificarsi nel periodo preoperatorio o postoperatorio e in qualsiasi momento dal periodo neonatale fino all'età adulta e può essere indipendente dalla gestione medica o dalla procedura chirurgica eseguita. [5]

1.1.2 DIAGNOSI

La diagnosi della malattia di Hirschsprung è generalmente basata su:

- Storia clinica
- Studi radiologici
- Esame istologico

Inizialmente si eseguono esami del sangue e indagini strumentali radiologiche (clisma opaco) per escludere altre patologie che possono manifestarsi nello stesso modo dell'HSCR (ad esempio l'ipotiroidismo congenito o la fibrosi cistica).

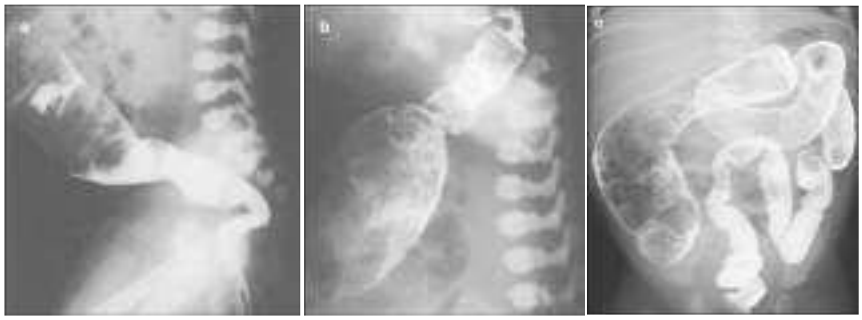


Figura 1: Clisteri di contrasto nella malattia di Hirschsprung a segmento corto (figure a e b) e lungo (figura c)

Una volta escluse altre malattie, per fare la diagnosi, l'esame più sensibile e specifico è la biopsia rettale, una procedura mininvasiva fatta al letto del paziente che consiste nel prelievo di due frammenti di mucosa e sottomucosa (le cellule che ricoprono l'intestino) del retto (ultima parte dell'intestino subito sopra l'ano) mediante una pinza appositamente dedicata. L'analisi al microscopio (istologica) dei frammenti dirà se il quadro è compatibile o meno con la diagnosi di HSCR.

Dal punto di vista microscopico la malattia è caratterizzata da assenza di cellule gangliari nei plessi mioenterico e sottomucoso e presenza di fibre e tronchi nervosi non mielinizzati ipertrofici nello spazio normalmente occupato da cellule gangliari e tronchi nervosi con dimensioni normali.

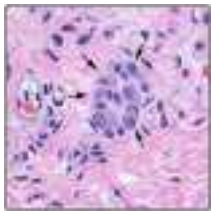


Figura 2: Ingrandimento su zona con cellule gangliari (indicate dalle frecce nere)

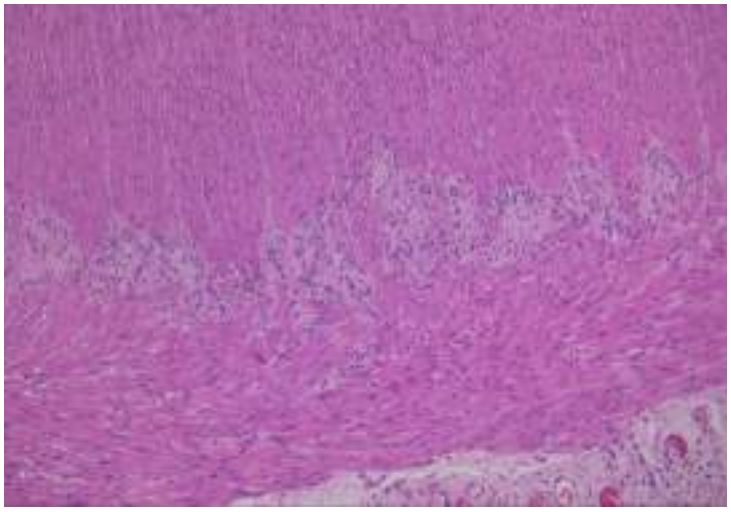


Figura 3: Immagine istologica di un tratto di intestino con tronchi normali e presenza di cellule gangliari

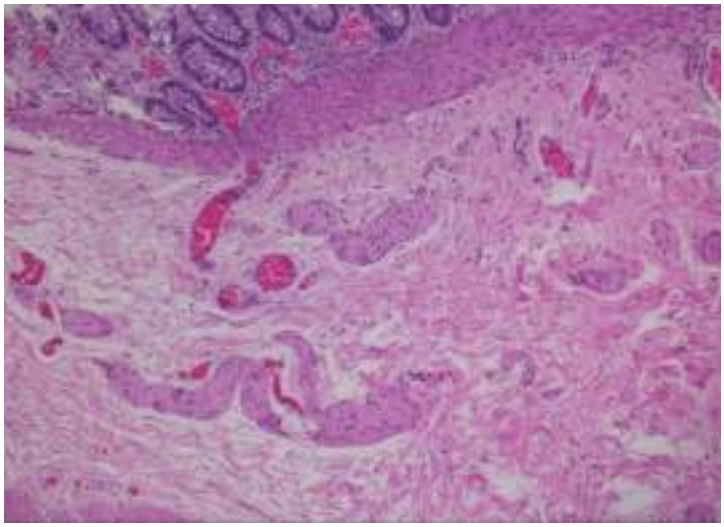


Figura 4: Immagine istologica di un tratto di intestino malato con tronchi ipertrofici e assenza di cellule gangliari

C’è inoltre una zona di transizione che rappresenta una lunghezza dell'intestino gangliare, immediatamente prossimale al segmento aganglionico, con caratteristiche neuropatologiche che sembrano correlate alla motilità. Le caratteristiche istopatologiche più spesso citate della zona di transizione includono aganglionosi circonferenziale parziale (cellule gangliari assenti intorno a parte della circonferenza intestinale), ipoganglionosi mioenterica (troppe poche cellule gangliari) e ipertrofia del nervo sottomucoso (nervi sottomucosi molto estesi). [4]

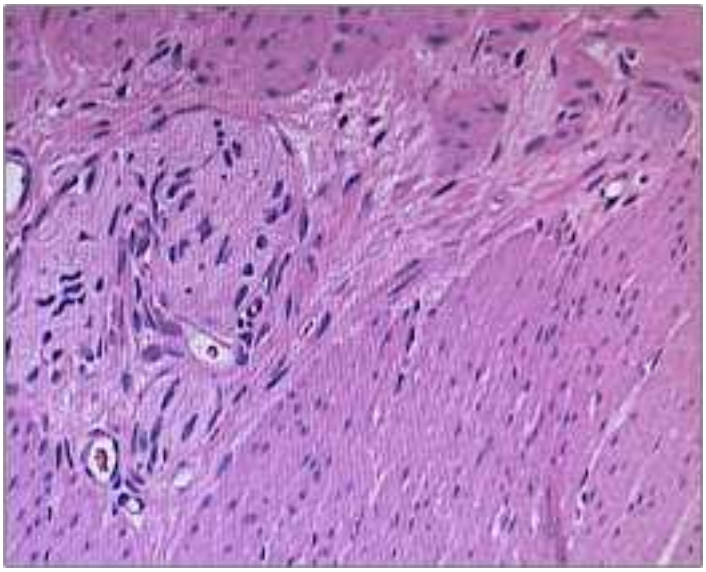


Figura 5: Immagine istologica della zona di transizione con tronchi ipertrofici e rare cellule gangliari

1.1.3 TRATTAMENTO

Il trattamento della malattia di Hirschsprung è chirurgico e l'intervento deve preferenzialmente essere effettuato il prima possibile, ovvero nei primi giorni-settimane di vita.

L'obiettivo della gestione chirurgica della malattia di Hirschsprung è quello di rimuovere il colon aganglionico e fare un'anastomosi sopra la linea dentata per ristabilire la continuità intestinale. [6]

Per tutta la durata dell'intervento il paziente è sottoposto ad anestesia totale e viene svegliato subito dopo.

Durante l'operazione vengono effettuate delle biopsie e in tempo reale l'anatomopatologo al microscopio verifica la presenza o no dei gangli.

La resezione di routine di almeno 5-10 cm di intestino gangliare prossimale al segmento aganglionico può ridurre l'incidenza della zona di transizione.

Tutto il pezzo resecato viene poi inviato in Anatomia Patologica per l'esame istologico definitivo che permette di verificare l'eventuale presenza di infiammazioni o enterocolite. Il numero di immagini istologiche da analizzare varia in base alla lunghezza del pezzo di intestino tagliato; l'utilizzo dell'intelligenza artificiale potrebbe aiutare a diminuire il tempo di analisi.

La tradizionale operazione radicale aperta ha una durata dell'intervento lunga, traumi gravi, molte complicanze postoperatorie e un'elevata mortalità. Lascia inoltre un'evidente cicatrice addominale e c'è un lungo tempo di esposizione degli organi addominali che può facilmente causare più sanguinamento durante l'operazione e ostruzioni intestinali adesive dopo l'operazione che possono ritardare l'inizio a mangiare e prolungare la degenza. [4]

Con l'evoluzione della tecnologia medica la laparoscopia è diventata un importante metodo di diagnosi e trattamento per la chirurgia mininvasiva. [6] Con l'aiuto di un laparoscopio, i pazienti beneficiano della stessa sicurezza ed efficacia delle procedure laparotomiche, ma ottengono anche migliori risultati estetici. Oltre alle cicatrici minimizzate, i rapporti hanno mostrato un ritorno più rapido della funzione intestinale, un recupero postoperatorio più breve, un migliore controllo del dolore, un minor rischio

di contaminazione addominale e di adesione nelle procedure laparoscopiche rispetto a quelle laparotomiche.

Anche i laparoscopi forniscono un chiaro campo visivo operativo per le osservazioni. L'intestino trascinato può essere visibile per garantire che non ci sia sanguinamento attivo o torsione lungo il suo asse longitudinale. Inoltre, la determinazione istologica del livello di zona di transizione diventa più facile e più conveniente con l'assistenza di un laparoscopio. [9]

Ci sono molti approcci chirurgici alla HSRC, tra cui l'approccio transaddominale (TAB) e quello transanale pull-through endorettale (TERPT). [8]

Transanal endorectal pull-through (TEPT) è considerato il metodo di trattamento preferibile per la malattia di Hirschsprung dal momento che è meno invasivo e ha meno morbilità rispetto al pull-through transaddominale. [6]

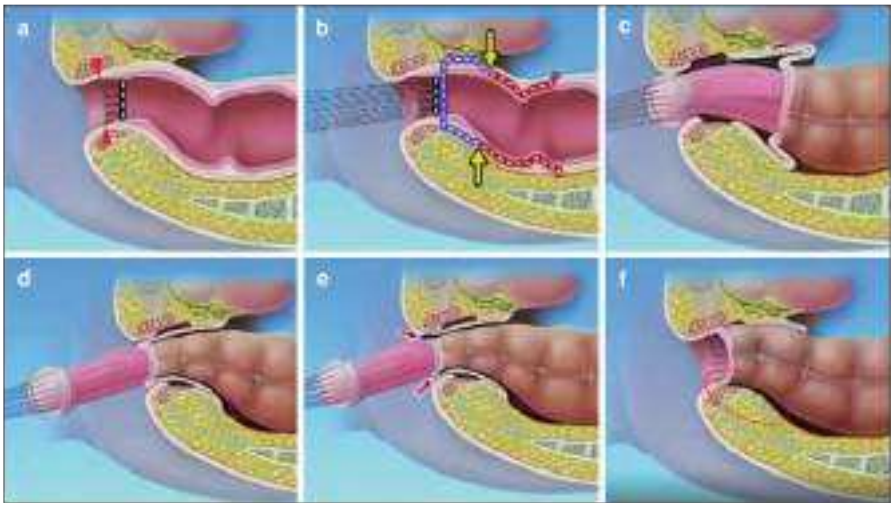


Figura 6: Relazioni anatomiche nella zona di transizione anale

Nella Figura 6 si può notare la linea dentata (frecce rosse) che si trova nella parte inferiore dei seni anali e la linea anorettale ARL (linea verticale tratteggiata) che è in cima alle colonne anali di Morgagni (a).

Il chirurgo nell'operazione transanale pull-through inizia la dissezione rettale quasi a tutto spessore trasversalmente procedendo cranicamente per 10-15 mm nel piano dello strato muscolare rettale (linea tratteggiata viola, b). Le frecce gialle indicano dove il piano di

separazione è cambiato in sottomucoso (b) e continua prossimalmente tra la mucosa rettale e gli strati muscolari rettali (linea tratteggiata marrone, b), ripiegandosi all'interno dal colon prossimale al colon distale (c) e infine all'ano (d). In questo momento l'orlo del muscolo rettale è diviso circonferenzialmente (frecce rosse; e), l'orlo rettale posteriore è completamente asportato (cerchio rosso punteggiato; f) e l'orlo rettale anteriore viene asportato dove si è formata la dissezione laparoscopica (freccia rossa vuota). [10]

Con questa chirurgia minimamente invasiva si elimina il rischio di complicazioni come aderenze addominali e lesioni del nervo pelvico. I vantaggi di TERPT includono un buon effetto cosmetico e un breve tempo di ospedalizzazione e la sua sicurezza è stata dimostrata da molti studi. Inoltre, TERPT ha una bassa incidenza di incontinenza/sporcizia e costipazione post-operatoria. [8]

Solo in casi particolari, come in presenza di perforazioni intestinali o gravi infezioni (enterocoliti/megacolon tossico) o quando l'assenza delle cellule nervose interessa tutto il colon o l'intestino tenue, può essere necessaria la realizzazione di un ano artificiale (colostomia o ileostomia) temporaneo. Questo permette al neonato di superare la situazione critica immediata, crescere, per poi realizzare in un secondo tempo e in sicurezza gli interventi di asportazione o di adattamento.

Dopo massimo quarantotto ore di digiuno, il neonato inizia una rialimentazione progressiva, se possibile al seno. La terapia antibiotica, per via endovenosa, dura pochi giorni ed è accompagnata inizialmente da una terapia analgesica per controllare il dolore post-operatorio. La dimissione, nei casi non complicati, avviene in media sette giorni dopo l'intervento.

1.1.4 ESITI FUNZIONALI

Dopo l'intervento definitivo, la prognosi è buona: la mortalità della malattia di Hirschsprung è solitamente inferiore al 3%, ma può raggiungere il 10% nei casi di enterocolite e nei casi in cui il tratto agangliare raggiunga l'ileo. La complicanza principale dell'intervento chirurgico può essere la costipazione post-operatoria persistente (7%-13% dei casi).

La quantificazione della lunghezza delle tre zone durante l'analisi istopatologica fornisce importanti informazioni cliniche. Questi dati possono aiutare i medici a valutare l'estensione della malattia, determinare la necessità di ulteriori trattamenti e monitorare la risposta del paziente all'intervento chirurgico o ad altre terapie.

L'esito funzionale a lungo termine in seguito all'intervento viene determinato utilizzando il punteggio della funzione intestinale (BFS) sulla base di sette variabili che sono:

- capacità di trattenere le defecazioni
- sente/riporta il bisogno di defecare
- frequenza delle defecazioni
- sporco
- incidenti fecali
- stitichezza
- problemi sociali

In questo sistema di punteggio, l'incontinenza fecale è classificata in sporco e incidenti. [6]

La qualità di vita del bambino sarà buona, qualora il tratto malato non sia eccessivamente lungo. I neonati curati per la malattia di Hirschsprung crescono bene, con peso e altezza talora solo di poco inferiori agli standard di riferimento.

1.2 BACKGROUND TECNOLOGICO

1.2.1 ANALISI DI IMMAGINI MEDICHE

L'analisi delle immagini mediche viene eseguita analizzando le immagini ottenute dai sistemi di imaging medico per risolvere problemi clinici. Lo scopo è estrarre informazioni efficaci e migliorare il livello di diagnosi clinica. [19]
Attualmente l'analisi delle immagini mediche include:

- Classificazione:
  - delle immagini/esami: in genere si hanno una o più immagini (un esame) come input e una singola variabile diagnostica come output (ad esempio, malattia presente o meno).
  - dell'oggetto/lesioni: solitamente si concentra sulla classificazione di una piccola parte (precedentemente identificata) dell'immagine medica in due o più classi.

- Rilevazione:
  - di organi, regioni e punti di riferimento: la localizzazione di oggetti anatomici (nello spazio o nel tempo), come organi o punti di riferimento, è un'importante fase di preelaborazione nelle operazioni di segmentazione o nel flusso di lavoro clinico per la pianificazione e l'intervento della terapia.
  - di oggetti o lesioni: è una parte fondamentale della diagnosi ed è una delle più laboriose per i clinici. Esiste una lunga tradizione di ricerca sui sistemi di rilevamento assistiti da computer progettati per rilevare automaticamente le lesioni, migliorando l'accuratezza del rilevamento o riducendo il tempo di lettura degli esperti umani.

- Segmentazione:
  - di organi e sottostrutture: consente l'analisi quantitativa dei parametri clinici relativi al volume e alla forma delle regioni di interesse. Il compito consiste nell'identificare i pixel o voxel che formano il contorno, volume o superfice dell'oggetto desiderato.

    - delle lesioni: combina le sfide del rilevamento di oggetti e della segmentazione di organi e sottostrutture nell'applicazione di algoritmi di deep learning. Il contesto globale e locale è tipicamente necessario per eseguire una segmentazione accurata.

- Registrazione: viene calcolata una trasformazione di coordinate da un'immagine medica all'altra. Spesso ciò viene eseguito in un quadro iterativo in cui si assume un tipo specifico di trasformazione (non) parametrica e si ottimizza una metrica predeterminata.

- Altri compiti nell'imaging medico:
    - Recupero di immagini basato sul contenuto: offre la possibilità di identificare casi clinici simili, comprendere malattie rare e, in definitiva, migliorare la cura del paziente.
    - Generazione e miglioramento delle immagini: è stata proposta una varietà di metodi che utilizzano deep learning che vanno dalla rimozione di elementi di ostruzione nelle immagini, alla normalizzazione delle immagini, al miglioramento della qualità dell'immagine, al completamento dei dati e alla scoperta di modelli.
    - Combinazione di dati di immagini con referti: ha portato a due percorsi di ricerca: sfruttare i referti per migliorare l'accuratezza della classificazione delle immagini e generare referti di testo da immagini.

Le principali aree di applicazione sono:

- Cervello
- Occhio
- Petto
- Patologia digitale e microscopia
- Seno
- Cardiaca
- Addome
- Muscoloscheletrica [15]

1.2.1.1 Segmentazione di immagini mediche

La segmentazione è il processo di partizione di un'immagine in regioni significative. Viene utilizzata per ottenere una rappresentazione più compatta, per estrarre degli oggetti o come strumento per l'analisi delle immagini e permette di partizionare le immagini digitali in insiemi di pixel.

La fase di segmentazione è importante perché consente di separare l'oggetto di studio dallo sfondo. Si tratta di un processo che associa un'etichetta o una categoria a ogni pixel di un'immagine. Lo scopo della segmentazione è semplificare e/o cambiare la rappresentazione delle immagini in qualcosa che è più significativo e facile da analizzare. Più precisamente, la segmentazione è il processo con il quale si classificano i pixel dell'immagine che hanno caratteristiche comuni, pertanto ciascun pixel in una regione è simile agli altri della stessa regione per una qualche proprietà o caratteristica (colore, intensità o texture).

Per calcolare la qualità della segmentazione, viene utilizzato un approccio che calcola il numero di pixel correttamente classificati sull'immagine generata rispetto all'immagine originale: maggiore è la qualità della segmentazione, migliore è l'accuratezza dell'elaborazione delle immagini. [17]

Segmentazione manuale

I tradizionali metodi di segmentazione manuale sono basati principalmente su metodi come l'estrazione dei confini, la segmentazione basata sulla soglia e la segmentazione basata sulla regione.

Tuttavia, può risultare difficile in ambito medico a causa dei seguenti motivi:

- le immagini possono essere molto rumorose
- la tessitura è complessa
- ci possono essere degli artefatti rilevanti
- gli oggetti di interesse possono avere forma complessa e poco prevedibile
- le variazioni clinicamente rilevanti possono essere difficili da rilevare
- occorre una validazione che garantisca affidabilità clinica

Segmentazione automatica

Per la segmentazione delle immagini mediche, come accennato precedentemente, a causa della complessità della loro elaborazione e alla mancanza di contorni chiari dei nuclei cellulari, si ha la necessità di utilizzare approcci più moderni.

Negli ultimi anni è stata ampiamente utilizzata la segmentazione automatica basata su metodi di *deep learning (DL)*, in cui una rete neurale può apprendere automaticamente le caratteristiche dell'immagine. [13]

L'uso di una rete neurale permette di migliorare la qualità della segmentazione. Con l'aumento del campione di allenamento, la qualità della segmentazione aumenta, in contrasto con gli algoritmi di segmentazione standard senza l'uso di reti neurali. [17]

Quando si tratta di elaborare in modo efficiente grandi quantità di dati ed estrarre le caratteristiche dell'immagine più rilevanti, uno dei principali vantaggi dell'utilizzo dei metodi DL è la loro capacità di apprendere caratteristiche complesse direttamente dai dati originali.

Anche se il metodo DL può fornire migliori prestazioni nell'elaborazione di immagini mediche, ha anche alcune limitazioni, che possono limitare la sua applicazione in alcuni campi della medicina clinica:

- una rete DL richiede un ampio dataset e un'elevata potenza di calcolo. Se il dataset è troppo piccolo e la potenza di calcolo è insufficiente, sarà necessario più tempo per formare la rete
- la maggior parte delle reti DL, come la rete neurale convoluzionale (CNN), hanno bisogno di tag e taggare manualmente immagini mediche è un compito difficile.

Queste limitazioni sono sempre più superate da miglioramenti nella potenza di calcolo, strutture di archiviazione dati, archiviazione digitale di immagini mediche e architetture di rete profonde.

La rete *U-net* è uno dei più importanti framework di segmentazione automatica per una rete neurale convoluzionale (CNN). È ampiamente utilizzato nel dominio dell'analisi delle immagini mediche per la segmentazione delle lesioni, la segmentazione anatomica e la classificazione. Il vantaggio di questo framework di rete è che non solo può segmentare accuratamente il target delle caratteristiche desiderate ed elaborare e valutare

in modo efficace le immagini mediche, ma anche aiutare a migliorare l'accuratezza nella diagnosi delle immagini mediche.

1.2.2 RETI NEURALI

Le reti neurali, note anche come reti neurali artificiali (o ANN, artificial neural network) o reti neurali simulate (o SNN, simulated neural network) sono un sottoinsieme del Machine Learning e sono l'elemento centrale degli algoritmi di Deep Learning (DL).
Tendenzialmente, si parla di Deep Learning e reti neurali in modo interscambiabile, il che può creare confusione. Di conseguenza, vale la pena notare che il termine "profondo" (deep) in Deep Learning si riferisce esclusivamente alla profondità dei livelli in una rete neurale. Una rete neurale che consiste in più di tre livelli, che sarebbero comprensivi degli input e dell'output, può essere considerata un algoritmo di Deep Learning, mentre una rete neurale che ha solo due o tre livelli è soltanto una rete neurale di base.
Il loro nome e la loro struttura sono ispirati al cervello umano, in quanto imitano il modo in cui i neuroni biologici si inviano segnali.
Le reti neurali fanno affidamento ai dati di addestramento per imparare e migliorare la loro accuratezza nel tempo. Tuttavia, una volta ottimizzati per l'accuratezza, questi algoritmi di apprendimento sono dei potenti strumenti nella computer science e nell'intelligenza artificiale (AI), consentendo di classificare e organizzare in cluster i dati ad alta velocità: le attività di riconoscimento vocale o di riconoscimento delle immagini possono richiedere pochi minuti invece che ore, come sarebbe il caso se l'identificazione manuale fosse affidata a esperti umani.

1.2.2.1 Modello neurone artificiale

Le reti neurali artificiali sono composte da livelli di nodi che contengono un livello di input, uno o più livelli nascosti e un livello di output. Ciascun nodo, o neurone artificiale, si connette ad un altro e ha un peso e una soglia associati. Se l'output di qualsiasi singolo nodo è al di sopra del valore di soglia specificato, tale nodo viene attivato, inviando i dati al successivo livello della rete. In caso contrario, non viene passato alcun dato al livello successivo della rete.
Si possono quindi identificare tre elementi di base del modello neurale:

- Un insieme di sinapsi, o collegamenti, ognuno dei quali è caratterizzato da un peso o forza propria. In particolare, un segnale X_j all'ingresso della sinapsi j collegato

al neurone k viene moltiplicato per il peso sinaptico w_{kj}. A differenza del peso di una sinapsi nel cervello, il peso sinaptico di un neurone artificiale può trovarsi in un intervallo che include valori negativi e positivi.

- Una sommatoria per sommare i segnali in ingresso, ponderati dalle rispettive forze sinaptiche del neurone.
- La funzione di attivazione che viene anche definita come una funzione di squashing, in quanto schiaccia (limita) l'intervallo di ampiezza ammissibile del segnale di uscita a un certo valore finito.

Può esserci anche un bias b_k che ha l'effetto di aumentare o abbassare l'input netto della funzione di attivazione, a seconda che sia positivo o negativo, rispettivamente.

In termini matematici, possiamo descrivere il neurone k scrivendo la coppia di equazioni:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j$$

$$y_k = \varphi(u_k + b_k)$$

Dove:

- $x_1, x_2, \ldots, x_m$ sono i segnali in ingresso;
- $w_{k1}, w_{k2}, \ldots, w_{km}$ sono i rispettivi pesi sinaptici del neurone k;
- u_k è l'uscita combinatrice lineare dovuta ai segnali in ingresso;
- b_k è il bias;
- $\phi(\cdot)$ è la funzione di attivazione;
- y_k è il segnale in uscita del neurone. [16]

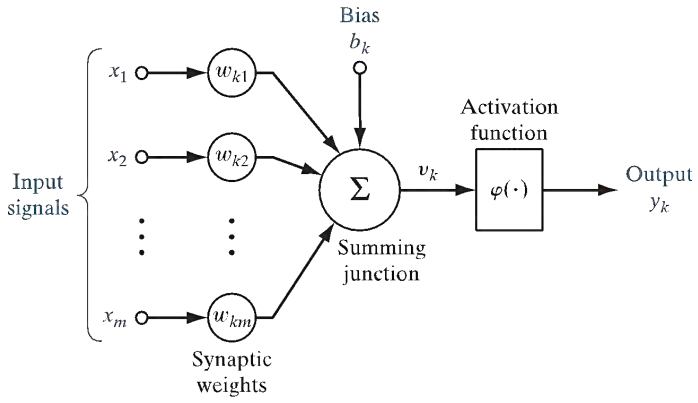


Figura 7: Modello neurone artificiale

Nel neurone binario a soglia come funzione di uscita si utilizza la funzione di Heaviside, illustrata in Figura 8, corrispondente ad un gradino con soglia θ. L'uscita di un neurone binario a soglia, pertanto, può essere espressa come:

$$y = +1 \quad \text{Se} \sum_{i=1}^{N} w_i x_i > \theta,$$
$$y = 0 \quad \text{Altrimenti.}$$

Il neurone binario a soglia dunque codifica l'informazione in ampiezza quantizzata su due valori di uscita: (0) e (1).

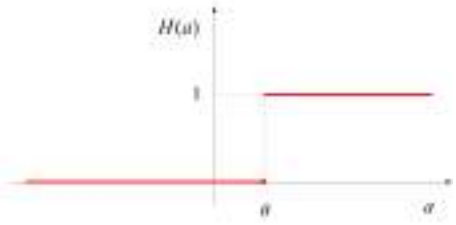


Figura 8: Funzione di Heaviside, utilizzata come funzione di uscita nel neurone binario a soglia

1.2.2.2 Addestramento rete neurale

Fissata la struttura (numero di livelli, neuroni, connessioni), l'addestramento di una rete neurale consiste nel determinare il valore dei pesi *w* che consentono di ottenere la miglior corrispondenza tra output prodotto dalla rete e l'output desiderato.

Gli algoritmi di apprendimento si dividono in generale in due macrocategorie:

- *Algoritmi ad apprendimento supervisionato:* in questo caso ad ogni esempio del dataset è associato un certo target che l'algoritmo dovrà poi predire nel testing e nell'applicazione. Il termine "supervisionato" deriva dal fatto che l'atto di etichettare i vettori input viene eseguito da un operatore umano che faccia da "insegnante".
- *Algoritmi ad apprendimento non supervisionato:* a questi algoritmi è richiesto di trovare da solo delle proprietà utili dalla struttura del dataset fornito. Uno dei compiti in cui il training è non supervisionato è il clustering, in cui l'algoritmo deve dividere il dataset in base alle proprietà degli esempi.

I due tipi di apprendimento non sono formalmente definiti, molti algoritmi di apprendimento sono in grado di risolvere entrambi i tipi di problemi.

Nel 1957 venne sviluppato il primo modello di neurone artificiale in grado di apprendere, il Perceptron. La regola di apprendimento, nota come *Delta Rule*, è semplice ma efficace e consiste nel modificare i pesi in modo proporzionale all'ingresso e all'errore (δ) commesso dal neurone, pari alla differenza tra l'uscita reale (y) e quella desiderata (y_d). La costante di proporzionalità è detta *learning rate η*.

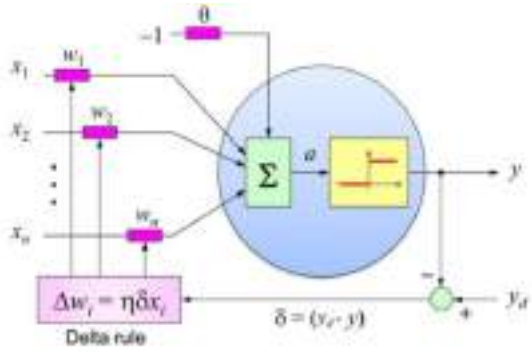


Figura 9: Il Perceptron di Rosenblatt

Nel 1986 si passò ad un potente algoritmo di apprendimento supervisionato, noto come Backpropagation, che permette ad una rete neurale di imparare a classificare dei pattern di ingresso attraverso un insieme di esempi, detto *training set*.
Le reti neurali addestrabili con Backpropagation sono di tipo stratificato, come quella illustrata in Figura 10.

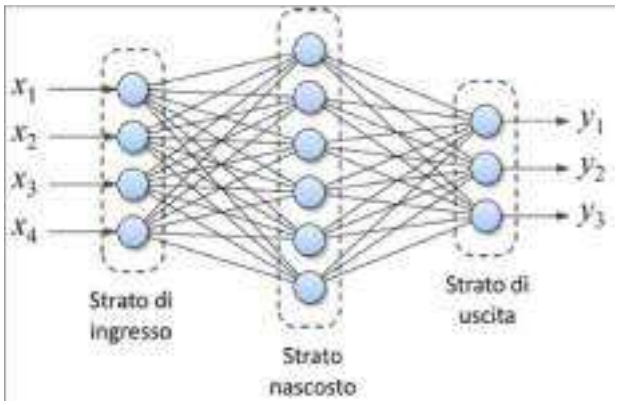


Figura 10: Esempio di rete a tre strati addestrabile con l'algoritmo di Backpropagation

Ogni neurone di uno strato è connesso con ogni neurone dello strato successivo, ma non esistono connessioni tra neuroni dello stesso strato, nè tra neuroni appartenenti a strati non adiacenti. Il primo strato è quello di ingresso (input layer), che riceve i dati da elaborare. L'ultimo strato è quello di uscita (output layer), che produce i risultati dell'elaborazione. Gli strati intermedi vengono detti strati nascosti (hidden layer) in quanto non sono visibili dall'esterno in una visione black-box della rete. In questo tipo di rete, il modello di neurone utilizzato in tutti gli strati è molto simile al neurone binario a soglia e differisce unicamente per la funzione di uscita.

Le reti neurali più comunemente utilizzate operano con livelli continui e la funzione di attivazione ϕ (·) è una funzione non-lineare ma continua e differenziabile (quasi ovunque):

- non-lineare per poter eseguire un mapping complesso dell'informazione di input.
- continua e differenziabile per la retro-propagazione dell'errore.

Le funzioni di uscita oggi più utilizzate sono la sigmoide, la tangente iperbolica e la lineare rettificata (ReLU), illustrate in Figura 11.

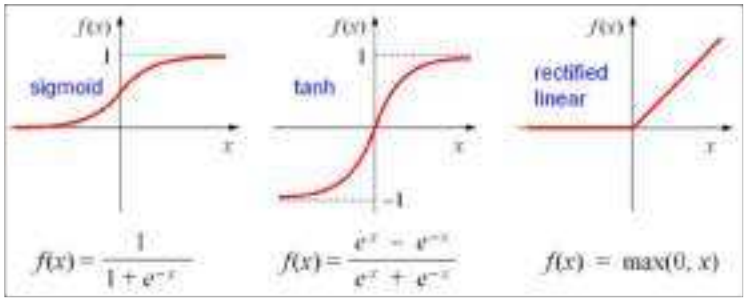


Figura 11: Funzioni di uscita comunemente utilizzate nelle reti multistrato addestrate con Backpropagation

Il funzionamento della rete è suddiviso in due fasi, dette di addestramento e di inferenza. Nella fase di addestramento vengono presentati gli esempi del training set dai quali la rete deve imparare e di cui è nota la classe.

Si parla di *forward propagation* in quanto vengono propagati gli input verso gli output. Ciascun esempio consiste in una coppia di dati vettoriali: l'ingresso da classificare (x) e l'uscita desiderata da associare (y_d). Per ogni esempio viene calcolata l'uscita della rete (y) e tali valori sono utilizzati per calcolare una funzione di errore, detta anche *Loss*

function, di cui se ne parlerà approfonditamente in seguito, utile per modificare i pesi della rete.
Prima dell'addestramento i pesi sono tipicamente inizializzati con valori random. Durante l'addestramento vengono modificati retro-propagando l'errore (backpropagation) in modo da minimizzare l'errore.
L'aspetto più interessante dell'apprendimento supervisionato è che la rete riesce a generalizzare ciò che ha appreso, classificando correttamente nuovi dati mai visti in fase di training.
Terminato l'apprendimento, nella fase di inferenza la rete deve stabilire se dei nuovi dati appartengano o meno alla classe.

1.2.2.3 Features

Quando il Machine Learning viene applicato alle immagini, i dati in input sono costituiti da caratteristiche, *feature*, derivate dalle immagini stesse.
Le feature posso essere di vario tipo e descritte in vari modi:

- valore di intensità del singolo pixel o dei pixel di una regione
- media e varianza dell'intensità dei pixel dell'intera immagine o di una regione
- autovettori ed autovalori calcolati in una regione
- istogramma dell'intera immagine o di una regione
- istogramma delle orientazioni di una regione

Le feature devono essere individuate nell'immagine e descritte:

- *Feature extraction (o feature detection):* ricavare da un'immagine informazioni che possono essere usate da un algoritmo di machine learning.
- *Feature vector:* vettore di descrittori di una feature

$$x = [x_1, x_2, ..., x_n]^T$$

La scelta delle feature e la loro descrizione ricoprono un ruolo fondamentale: le feature variano tra classi diverse ed all'interno della stessa classe. Le classi sono separabili se la differenza tra le feature che appartengono alla stessa classe è maggiore della differenza tra feature che appartengono a classi diverse.

1.2.2.4 Stochastic Gradient Descent (SGD)

Come già esposto, l'obiettivo fondamentale nell'addestramento supervisionato di una rete neurale è quello di determinare i parametri del modello che consentano di minimizzare la *Loss Function*.

La Gradient Descent è la tecnica di ottimizzazione più utilizzata in quanto aggiorna iterativamente i parametri imparabili della rete in modo da minimizzare l'errore. [13]

La *Loss Function* più usata è la somma dei quadrati degli errori, che quantifica quanto l'output prodotto dalla rete neurale si discosta da quello desiderato:

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i}(y^{(i)} - \Phi(z^{(i)}))^2$$

dove

- *J(w)* è la funzione da minimizzare;
- $y^{(i)}$ è il valore desiderato in uscita, o l'uscita reale;
- $\Phi(z^{(i)})$ è la funzione in uscita dall'ottimizzatore (Adam, Adeline, ecc.).

Per la fase di training, bisogna definire il numero di epoche ed il numero di iterazioni per epoca:

- *Epoch:* con epoca si intende la presentazione alla rete di tutti i feature vector del training set; ad ogni epoca, gli *n* feature vector del training set sono ordinati in modo casuale e poi suddivisi, considerandoli sequenzialmente, in gruppi di uguale dimensione $size_{mb}$, denominati mini-batch.
- *Iteration:* con iterazione si intende la presentazione (una volta) dei feature vectors costituenti un mini-batch e il conseguente aggiornamento dei pesi. Il numero di iterazioni per epoca è legato alla dimensione del mini-batch:

$$numero\ iterazioni = n\ /\ size_{mb}$$

La *Loss Function J(w)* può essere ridotta modificando i pesi *w* in direzione opposta al suo gradiente: il gradiente indica la direzione di maggior crescita di una funzione (di più variabili) e muovendoci in direzione opposta riduciamo (al massimo) l'errore.

J(w) è nota solo per alcuni valori assegnati ai pesi *w* e potrebbe avere anche minimi locali.

Lo spostamento è determinato dal Learning Rate η che è lo step che si intende fare ad ogni iterazione, ovvero di quanto spostarsi lungo l'antigradiente ad ogni iterazione, ed è il parametro più interessante e delicato in fase di addestramento di una rete:

- un valore troppo basso di η può comportare di raggiungere il minimo globale troppo lentamente
- un valore troppo alto di η può comportare di non raggiungere mai il minimo globale

Una possibile soluzione è variare il Learning Rate η nelle varie iterazioni: il valore di η è alto nelle prime iterazioni e decresce man mano nelle iterazioni successive.

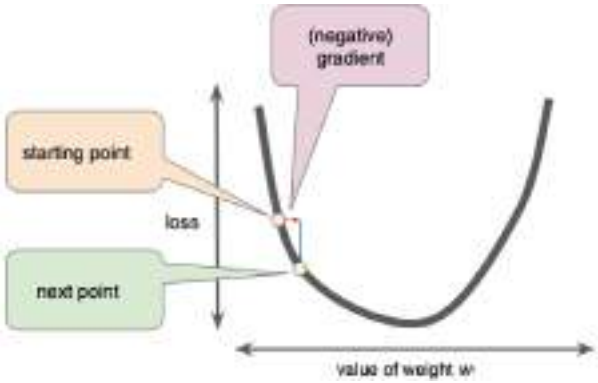


Figura 12: Ottimizzazione iterativa mediante discesa del gradiente della funzione costo

Il termine stocastico deriva dal fatto che l'utilizzo di un set ridotto di dati di addestramento produce una stima rumorosa del gradiente altrimenti ottenuto utilizzando tutti gli esempi [17].

Questa semplice procedura di solito trova una buona serie di pesi in modo sorprendentemente rapido se confrontata con tecniche di ottimizzazione molto più elaborate. [18]

Diversi algoritmi di ottimizzazione derivano dalla SGD, tra cui uno dei principali e che verrà utilizzato nel nostro modello di apprendimento è il metodo Adam. È un algoritmo di ottimizzazione che tenta di migliorare l'addestramento della rete utilizzando learning rate in grado di adattarsi automaticamente e singolarmente, parametro per parametro, in base alla *Loss function* da minimizzare.

1.2.2.5 Overfitting

L'overfitting rappresenta un problema comune nell'addestramento supervisionato di modelli complessi. Si tratta della situazione in cui la rete "memorizza" i dati di addestramento piuttosto che apprendere da essi alcune caratteristiche generali che consentono di predire in modo accurato input mai visionati. Viene dunque a mancare la capacità di generalizzazione del modello.

Di seguito vengono proposte alcune strategie volte a ridurre la situazione di overfitting:

- *Regolarizzazione:* un esempio è il *weight decay*, decadimento di peso, che aggiunge un termine di penalità alla *Loss function* in modo che i pesi della rete convergano a valori assoluti più piccoli. [17]
- *Batch normalization:* opera sottraendo la media e dividendo per la deviazione standard di ciascun mini-batch di dati in modo da normalizzarlo.
- *Dropout:* rimuove casualmente le unità nascoste dalle reti neurali durante l'allenamento; le unità disattivate dunque non contribuiscono alla fase di propagazione in avanti e non partecipano alla backpropagation, consentendo di ridurre il co-adattamento tra neuroni.
- *Data augmentation:* una tecnica che consiste nel processare i dati a disposizione elaborandoli mediante trasformazioni cosiddette label-preserving per poter aumentare il dataset. [20] Le più comuni consistono nel tagliare, ruotare, riflettere o aggiungere rumore gaussiano alle immagini.

1.2.2.6 Validazione

Per stabilire la capacità di generalizzazione di un modello è necessario valutarne la performance su dati non inclusi nel set di addestramento e dunque mai visionati dalla rete. Prima di iniziare l'allenamento è quindi necessario rimuovere dal dataset alcuni dati che andranno a costituire il *test set*. Questi dati, al termine dell'allenamento, permetteranno di individuare i casi di overfitting e di testare la capacità di produrre risposte sensate su nuovi input che il sistema non ha mai visto durante l'addestramento.

In caso di dataset di allenamento con numerosità elevata di dati, un approccio comune è quello di suddividerlo in due gruppi:

- *Training set*, tipicamente l'80%, usato per l'addestrare la rete.
- *Validation set*, il restante 20%, utilizzato per valutare la perdita ed eventuali metriche del modello al termine di ogni epoca; il modello non viene addestrato su questi dati.

Quando i dati di addestramento hanno numerosità ridotta (non si riescono a formare i tre set con abbastanza variabilità), una tecnica comune è la *K-fold cross-validation*: consiste nel suddividere l'intero dataset a disposizione in K sottoinsiemi di uguale dimensione ed eseguire K diversi cicli di training, in ognuno dei quali si esclude un diverso sottoinsieme tra i K e lo si usa per la validazione. Le prestazioni del modello vengono misurate facendo la media degli errori quadrati sul validation set per ognuna delle K ripetizioni.
Questa tecnica permette di testare modelli simili tra loro su insiemi di dati diversi e validare quindi un modello generale addestrato su tutti i dati a disposizione. Se K = N, dove N indica il numero di esempi nel training set, il metodo viene detto *leave-one-out-cross-validation*: nel K-esimo ciclo di addestramento, vengono utilizzati tutti i dati ad eccezione del K-esimo, che viene invece usato per la validazione [35libro, 18].

1.2.3 CONVOLUTIONAL NEURAL NETWORKS

Dal 2000 ad oggi sono stati ideati nuovi modelli di rete neurale che hanno permesso di risolvere problemi prima considerati intrattabili con queste tecniche. In ordine temporale i modelli più rilevanti proposti in letteratura sono le reti ricorrenti, le reti convoluzionali e le reti generative.

Una delle reti neurali profonde più popolari e che viene utilizzata in questa tesi è la rete neurale convoluzionale (CNN). Prende questo nome dal funzionamento lineare matematico tra matrici chiamato convoluzione. [14]

La CNN è un costrutto matematico tipicamente composto da tre tipi di strati (o elementi costitutivi): convoluzione, raggruppamento e strati completamente connessi. I primi due livelli, convoluzione e raggruppamento, eseguono l'estrazione delle caratteristiche, mentre il terzo, un livello completamente connesso (*fully connected*), mappa le caratteristiche estratte nell'output finale. [19]

L'architettura della CNN comprende diversi elementi:

- Strato convoluzionale
- Funzione di attivazione non lineare
- Strato di pooling
- Strato completamente connesso
- Funzione di attivazione dell’ultimo livello

La CNN ha una prestazione eccellente nei problemi di apprendimento automatico, in particolare le applicazioni che trattano i dati delle immagini.

L'aspetto più vantaggioso delle CNN è la riduzione del numero di parametri. [14]

Un *parametro* indica una variabile che viene imparata automaticamente durante il processo di formazione. Un *iperparametro* si riferisce a una variabile che deve essere impostata prima dell'inizio del processo di allenamento.

Gli strati convoluzionali e completamente connessi hanno parametri, mentre quelli di pooling e non linearità non li hanno.

	Parameters	Hyperparameters
Convolution layer	Kernels	Kernel size, number of kernels, stride, padding, activation function
Pooling layer	None	Pooling method, filter size, stride, padding
Fully connected layer	Weights	Number of weights, activation function
Others		Model architecture, optimizer, learning rate, loss function, mini-batch size, epochs, regularization, weight initialization, dataset splitting

Figura 13: Elenco di parametri e iperparametri in una rete neurale convoluzionale (CNN)

Un'architettura tipica di una CNN consiste in ripetizioni di una pila di diversi strati di convoluzione e di uno strato di pooling, seguiti da uno o più strati completamente connessi (FC).

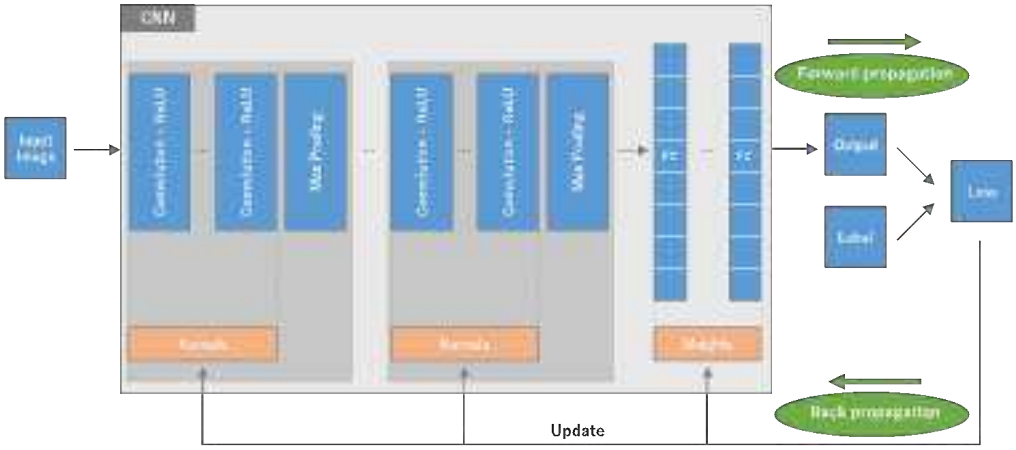


Figura 14: Una panoramica di una rete neurale convoluzionale (CNN) architettura e il processo di formazione

L'ottimizzazione dei parametri della rete avviene con le stesse modalità descritte in precedenza: il segnale viene propagato in avanti (*Forward propagation*) nella rete fino allo strato finale, dove il gradiente della *Loss function* viene calcolato e retro-propagato (*Back propagation*) per consentire l'aggiornamento dei pesi tramite un algoritmo di discesa del gradiente [21].

1.2.3.1 Strato convoluzionale

Uno strato convoluzionale è una componente fondamentale dell'architettura CNN perché consente di eseguire l'estrazione delle caratteristiche. Consiste tipicamente in una

combinazione di operazioni lineari e non lineari, ad esempio operazione di convoluzione e funzione di attivazione.

La convoluzione è un tipo specializzato di operazione lineare usata per l'estrazione delle caratteristiche, dove un piccolo array di numeri, chiamato *kernel*, viene applicato all'input, che è un insieme di numeri chiamato *tensore*. Viene calcolato in ciascuna posizione del tensore un *element-wise product* tra ciascun elemento del kernel e il tensore di ingresso e poi sommato per ottenere il valore di uscita nella posizione corrispondente del tensore di uscita, che prende il nome di *feature map*.

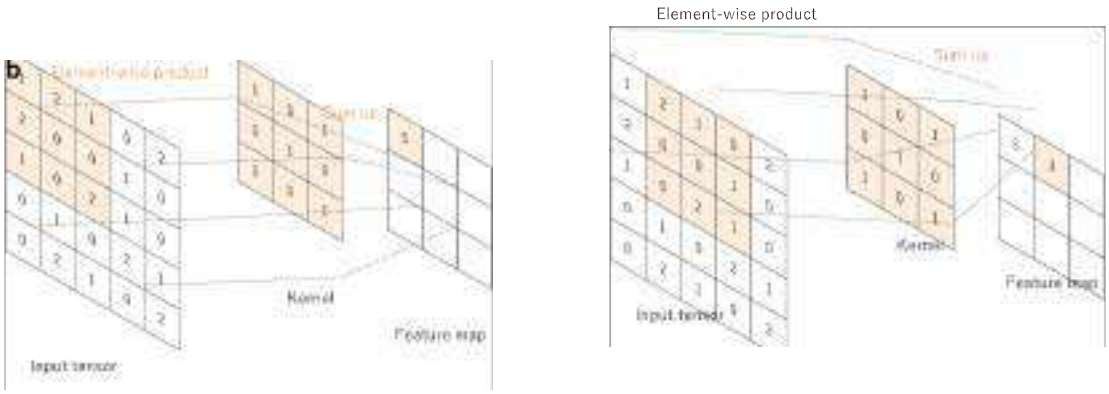


Figura 15: Esempio di convoluzione di operazioni con una tensione di 3x3, senza padding, e un passo 1

Questa procedura viene ripetuta applicando più kernel per formare un numero arbitrario di *future map*, che rappresentano diverse caratteristiche dei tensori di input; dunque, diversi kernel possono essere considerati come estrattori di caratteristiche differenti.

Due iperparametri chiave che definiscono l'operazione di convoluzione sono la dimensione, che tipicamente è 3x3, ma a volte 5x5 o 7x7, e il numero dei kernel, che è arbitrario e determina la profondità delle *feature map* di output.

Uno degli svantaggi della fase di convoluzione è la perdita di informazioni che potrebbero esserci sul bordo dell'immagine: infatti l'operazione di convoluzione descritta sopra non permette al centro di ogni kernel di sovrapporre l'elemento più esterno del tensore di input, con conseguente riduzione dell'altezza e della larghezza della feature map di output rispetto al tensore di input.

Il padding, tipicamente *zero padding*, è una tecnica per affrontare questo problema: righe e colonne di zeri vengono aggiunte su ogni lato del tensore di input in modo che il centro di un kernel possa adattarsi sull'elemento più esterno e mantenere la stessa dimensione

sul piano durante l'operazione di convoluzione. Senza zero padding, ogni mappa delle caratteristiche successive si ridurrebbe dopo l'operazione di convoluzione.

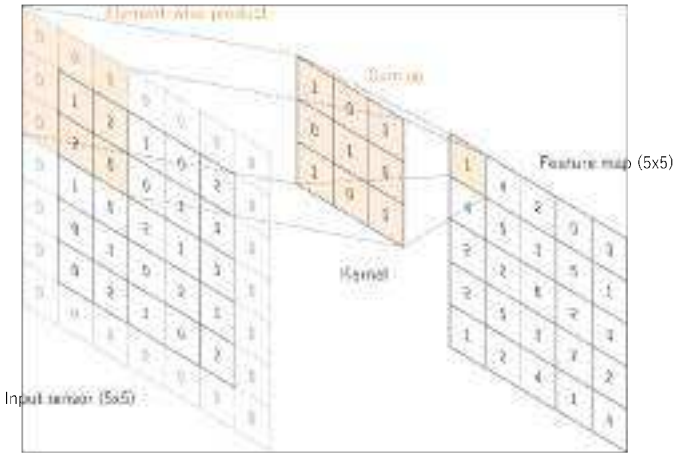


Figura 16: Operazione di convoluzione con zero padding

La distanza tra due posizioni successive del kernel è detta *stride,* passo, e definisce l'operazione di convoluzione. La scelta più comune è 1; tuttavia un passo più grande di 1 è talvolta utilizzato al fine di ottenere downsampling delle *feature map*. Una tecnologia alternativa per eseguire il downsampling è l'operazione di pooling, che verrà descritta di seguito.

Il processo di formazione di un modello CNN per quanto riguarda il livello di convoluzione è quello di identificare i kernel che funzionano meglio per un dato compito sulla base di un dato training dataset. I kernel sono gli unici parametri imparati automaticamente durante il processo di addestramento nello strato di convoluzione; d'altra parte, le dimensioni dei kernel, il numero di kernel, padding e stride sono iperparametri che devono essere impostati prima dell'inizio del processo di allenamento. [19]

1.2.3.2 Funzione di attivazione non lineare

Le uscite di un'operazione lineare quale la convoluzione sono in seguito passate con una funzione di attivazione non lineare per regolare o tagliare l'output generato. Sebbene funzioni non lineari lisce, come la funzione sigmoide o iperbolica tangente (Tanh), siano state usate in precedenza perché sono rappresentazioni matematiche di un comportamento

biologico dei neuroni, la funzione di attivazione non lineare più comune utilizzata attualmente è l'unità lineare rettificata (ReLU), che calcola semplicemente la funzione:

$$f(x) = max(0, x)$$

La funzione sostituisce ogni pixel con valore negativo con il valore 0.

Questo livello viene applicato per saturare o limitare l'output generato. [19]

1.2.3.3 Strato di pooling

Uno strato di pooling fornisce una tipica operazione di downsampling che riduce le dimensioni sul piano delle feature map per introdurre una invarianza di traduzione a piccoli spostamenti e distorsioni, e diminuire il numero di parametri di apprendimento successivi. Si noti che non esiste un parametro imparabile in nessuno degli strati di pooling, mentre la dimensione del filtro, il passo e il padding sono iperparametri simili alle operazioni di convoluzione. [19]

Max pooling

La forma più popolare dell'operazione di pooling è il max pooling, che estrae patch dalle feature maps di input, produce il valore massimo in ogni patch e scarta tutti gli altri valori. Nella pratica è comunemente usato un max pooling con un filtro di dimensione 2x2 con un passo di 2. Questo riduce di un fattore 2 la dimensione sul piano delle feature map. A differenza dell'altezza e della larghezza, la dimensione della profondità delle feature map rimane invariata. [19]

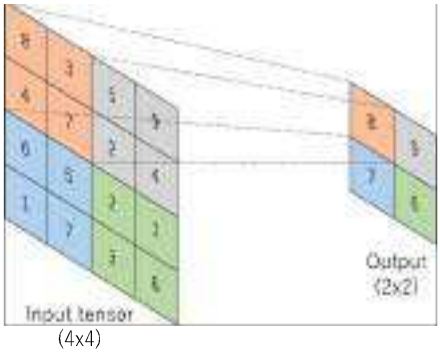


Figura 17: Esempio di operazione di max pooling

1.2.3.4 Strato completamente connesso

Lo strato completamente connesso è simile al modo in cui i neuroni sono disposti in una rete neurale tradizionale. Pertanto, ogni nodo in un livello completamente connesso è collegato direttamente ad ogni nodo sia nel livello precedente che nel livello successivo.
Le prestazioni della rete dipendono anche dal numero di livelli all'interno della rete, d'altra parte come il numero di livelli aumenta, aumenta anche il tempo necessario per allenare e testare la rete. [14]
Le feature map di output degli strati di convoluzione o pooling finali sono tipicamente appiattite, cioè trasformate in un array unidimensionale (1D) di numeri (o vettori) e collegate a uno o più strati completamente connessi, noti anche come strati densi, in cui ogni input è collegato a ogni output da un peso apprendibile.
Il livello finale completamente connesso ha tipicamente lo stesso numero di nodi di uscita del numero di classi. Ogni strato completamente connesso è seguito da una funzione non lineare, ad esempio ReLU, come descritto sopra. [19]

1.2.3.5 Funzione di attivazione dell'ultimo livello

La funzione di attivazione applicata all'ultimo livello completamente connesso è di solito diversa dalle altre. Deve essere selezionata una funzione di attivazione appropriata in base al compito che deve eseguire. [19]
Ad esempio, quella applicata all'attività di classificazione multiclasse è una funzione Softmax che classifica ogni pixel separatamente, produce N canali di probabilità per ogni N numero di classi. La classe con la probabilità più alta in ogni pixel corrisponde alla segmentazione prevista.
Di seguito è riportato un elenco di funzioni di attivazione dell'ultimo livello comunemente applicate per vari compiti.

Task	Last layer activation function
Binary classification	Sigmoid
Multiclass single-class classification	Softmax
Multiclass multiclass classification	Sigmoid
Regression to continuous values	Identity

Figura 18: Funzioni di attivazione dell'ultimo livello

1.2.4 U-NET

U-Net è uno dei più importanti framework di segmentazione per una rete neurale convoluzionale (CNN) e, al giorno d'oggi, è la prima scelta per risolvere la segmentazione delle immagini mediche. [3,1]

Il vantaggio di questo framework di rete è che non solo può segmentare accuratamente il target delle caratteristiche desiderate ed elaborare e valutare in modo efficace le immagini mediche, ma aiuta anche a migliorare l'accuratezza nella diagnosi delle immagini mediche.

La rete U-Net è adatta a tutti i tipi di problemi di segmentazione biomedica in quanto consente la segmentazione di immagini arbitrariamente grandi. [19]

Inoltre, è stata posta l'attenzione su questa rete a causa della sua capacità di ottenere brillanti risultati anche partendo da un dataset esiguo.

Questa rete può essere applicata in diversi sistemi di imaging e analisi di immagini mediche, tra cui tomografia computerizzata (CT), risonanza magnetica (MRI), ultrasuoni, raggi X, tomografia a coerenza ottica (OCT) e tomografia computerizzata a emissione di positroni (PET). [19]

Questa architettura si basa su una sequenza di strati convoluzionali e la sua caratteristica è che nella parte in cui c'è un aumento di dimensione, ci sono un gran numero di canali di funzionalità che consentono alla rete di distribuire informazioni contestuali ai livelli di risoluzione superiore.

Questa tipologia di rete non utilizza strati completamente connessi, solo convoluzionali. [1]

È costituita da un percorso di codifica/contrazione, per catturare il contesto nell'immagine, seguito da un percorso di decodifica/espansione simmetrica per ottenere la classificazione di ogni singolo pixel.

L'architettura della rete è illustrata in Figura 19.

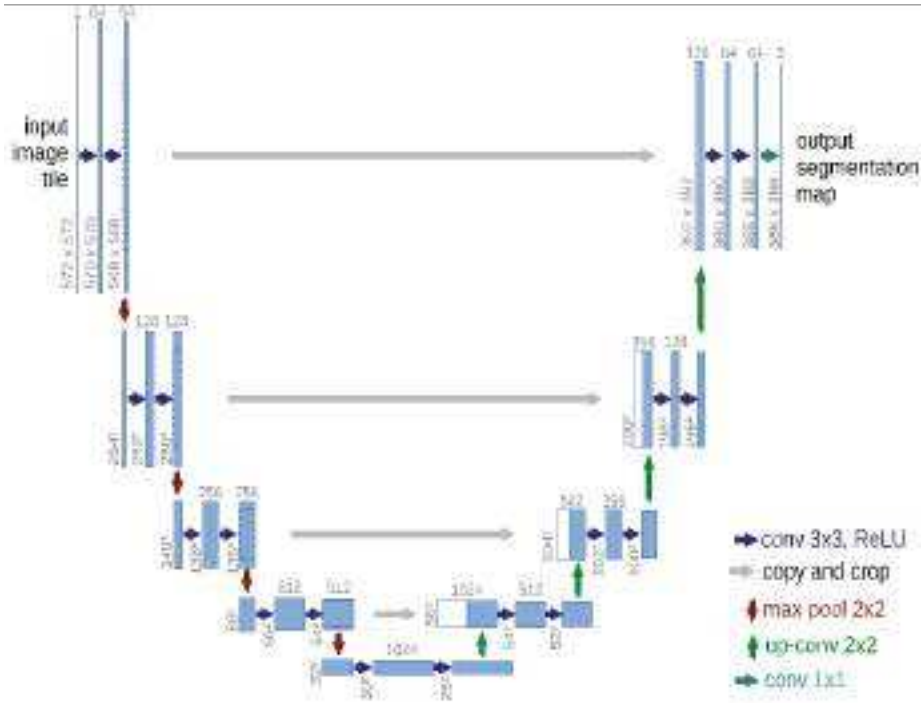


Figura 19: Architettura U-Net

Il percorso di contrazione, *encoder path*, è nel lato sinistro mentre il percorso espansivo, *decoder path*, è il lato destro.

In particolare, si ha la ripetizione di due blocchi:

- *Encoder block*: caratterizzato dall'applicazione ripetuta di due convoluzioni, ciascuna seguita da una funzione di attivazione ReLU, e da un'operazione di Max Pooling 2x2 con passo 2 per il downsampling. Ad ogni passo di downsampling si raddoppia il numero di canali di funzionalità, ma si dimezza la dimensione dell'input.
- *Decoder block*: consiste in un up-sampling (operazione inversa del Max Pooling) della *feature map* seguito da una convoluzione 2x2, una concatenazione con la corrispondente *feature map* ritagliata dal percorso contraente e due convoluzioni 3x3, ciascuna seguita da una funzione di attivazione ReLU. All'ultimo livello viene applicata una convoluzione 1x1 seguita da una funzione di attivazione (scelta in base al compito da svolgere) per mappare ogni *feature vector* al numero desiderato di classi.

L' *Encoder block* viene ripetuto cinque volte nel percorso di contrazione, anche se la quinta volta non è seguito dal Max Pooling e quindi in totale si hanno:

- 10 livelli convoluzionali 2D con filtri 3x3, ognuno con attivazione ReLU
- 4 livelli Max Pooling 2D con filtri 2x2 e valore di stride pari a 2

Il *Decoder block* viene ripetuto quattro volte per un totale di:

- 4 livelli che prendono il nome di conv2Dtranspose o "up-convolution" e che consistono in un up-sampling seguito da una convoluzione 2D con filtri 2x2
- 8 livelli convoluzionali 2D con filtri 3x3, ognuno con attivazione ReLU

1.2.5 STRUMENTI SOFTWARE

I principali linguaggi di programmazione solitamente utilizzati per implementare algoritmi di Machine Learning sono tre: Python, C++, Matlab. Tutti questi linguaggi permettono, attraverso l'uso di apposite librerie, di effettuare i calcoli di interesse per le CNN (convoluzione 2D, derivate ecc).

1.2.5.1 Python

Il motivo per cui è stato scelto Python come linguaggio di programmazione è la semplicità. Python è, come C++, un linguaggio ad oggetti con la possibilità di creare classi, tuttavia è nato con un'impostazione user-friendly che lo rende molto didattico e più semplice di C++. Python non raggiunge le prestazioni di C++, in quanto questi, grazie ai puntatori, può essere ottimizzato meglio lavorando a più basso livello. È pur vero che, grazie a librerie ottimizzate (scritte in C++ anche per Python), la differenza di prestazioni oscilla tra il 5 e il 10%. Matlab, purtroppo, non possiede le librerie degli altri due linguaggi, rendendo perciò tutto il procedimento decisamente più complesso.

Per realizzare e implementare modelli robusti ed efficienti è necessaria una potenza di calcolo davvero importante. Per le prime fasi di test può essere sufficiente lavorare sulla propria macchina, ma con il crescere delle dimensioni del dataset la potenza necessaria per il training dei modelli può aumentare a dismisura, per questo motivo sono presenti sul mercato diversi tool, tra cui Google Colab.

Google Colab è uno strumento gratuito presente nella suite Google che consente di scrivere codice Python direttamente dal proprio browser.

Si può creare un documento Google Colab direttamente da Google Drive e, proprio come un qualunque documento in G Suite, può essere condiviso con altri utenti che hanno la possibilità di modificarlo e lasciare commenti direttamente nel notebook.

Le macchine virtuali messe a disposizione in Google Colab ospitano un ambiente configurato che consente di concentrarsi sin da subito sui progetti di Data Science: sono presenti numerose librerie Python, tra cui moltissime di Data Science come Keras e Tensorflow.

1.2.5.2 Tensorflow

La libreria più interessante, appartenente sia a Python che a C++, si chiama Tensorflow. Tensorflow è una libreria open source per l'apprendimento automatico che permette di velocizzare significativamente i calcoli necessari ad un algoritmo di Machine Learning e contiene una serie di funzioni che sono in grado di sfruttare i tensori.
Il nome è infatti composto dai termini Tensor e Flow:

- *Tensor* è un array multidimensionale ossia una matrice di tre o più dimensioni; in algebra lineare è detta tensore.
- *Flow* indica un flusso di operazioni.

Tensorflow possiede una API chiamata Keras che viene utilizzata per la creazione di modelli.

1.2.5.3 Keras

Keras è una API di Tensorflow molto utilizzata in Machine Learning per creare modelli di CNN con poche righe di codice. Keras possiede, inoltre, le funzioni necessarie a creare e salvare modelli di CNN per poter essere utilizzati successivamente in altre applicazioni (file del tipo modello.h5)

1.3 LIMITI TECNOLOGICI

1.3.1 DATASET RISTRETTO

I modelli di deep learning dipendono principalmente dalla disponibilità di enormi set di dati. Senza l'esistenza di molte immagini nel dataset, diversi modelli di deep learning non sarebbero in grado di apprendere e produrre modelli accurati. Sfortunatamente diversi campi, come l'elaborazione di immagini mediche, non hanno accesso a grandi quantità di prove.

L'aumento dei dati è molto importante per superare la limitazione dei campioni di dati. [18] Un metodo semplice ma efficace è rappresentato dalla *data augmentation*, una tecnica che consiste nel processare i dati a disposizione elaborandoli mediante trasformazioni cosiddette *label-preserving*, come ad esempio taglio, rotazione, riflessione o aggiunta di rumore gaussiano alle immagini [20]

La malattia di Hirschsprung è relativamente rara e rappresenta una sfida significativa per lo sviluppo di algoritmi diagnostici poiché un set di dati di 10.000 o più vetrini semplicemente potrebbe non essere fattibile per un dato istituto e si rivelerebbe piuttosto impegnativo anche con diversi ospedali che lavorano in collaborazione.

A titolo dimostrativo, il numero totale di nati vivi negli Stati Uniti per il 2018 è stato di circa 3,8 milioni. Supponendo un'incidenza media di HSCR di circa 1:8000, ci si aspetterebbe meno di 500 casi all'anno. Ciò significa che ottenere un set di dati grande come quelli generalmente utilizzati (oltre 15.000 casi) richiederebbe la raccolta di tutti i casi HSCR negli Stati Uniti per oltre 30 anni.

Queste difficoltà intrinseche possono anche spiegare perché l'applicazione dell'AI e dell'apprendimento automatico nella malattia di Hirschsprung rimane un territorio relativamente inesplorato. [1]

1.3.2 TEMPISTICHE CLINICHE

Le tempistiche a cui dobbiamo fare riferimento quando si tratta la malattia di Hirschsprung variano a seconda delle fasi:

- *Fase 1: diagnosi*

La diagnosi della malattia di Hirshsprung avviene tramite biopsia rettale e l'esito si ha dopo 2-3 settimane.

- *Fase 2: intervento*

  Durante l'intervento l'anatomopatologo analizza in tempo reale al microscopio la presenza o meno delle cellule gangliari del segmento che deve essere resecato e il tutto dura 2-3 ore.

- *Fase 3: analisi istopatologica post-intervento*

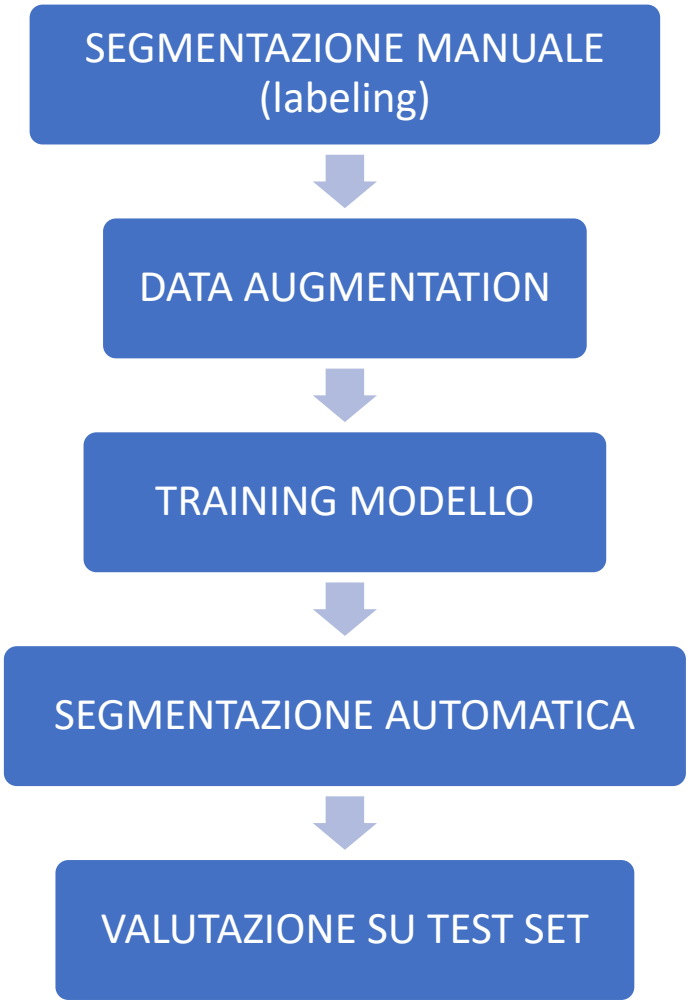
  Il tratto di intestino resecato durante l'intervento viene inviato in anatomia patologica e viene fatta l'analisi istopatologica di tutto il pezzo per quantificare la lunghezza delle tre zone: zona sana, zona di transizione e zona patologica. Il numero delle immagini dipende dalla lunghezza del pezzo, in genere varia tra le 30-50 immagini e i risultati si hanno dopo circa 20 giorni.

Per accorciare questi tempi, in particolare nell'ultima fase post-operatoria, potrebbe essere utile l'utilizzo dell'intelligenza artificiale in quanto essa si occuperebbe dell'individuazione della presenza (o assenza) delle cellule gangliari, lasciando all'anatomopatologo solo l'analisi delle immagini dubbie della zona di transizione.

2 METODOLOGIA PROPOSTA

Questo capitolo si concentra sulla metodologia utilizzata per l'addestramento della rete neurale. Inizialmente, viene eseguita la segmentazione manuale per creare il dataset composto dalle immagini e dalle relative maschere. Successivamente, si procede con la fase di data augmentation al fine di aumentare le dimensioni del dataset. Infine, la rete neurale viene addestrata per eseguire la segmentazione automatica.

Nel capitolo successivo, sarà affrontato il passaggio finale che riguarda il test della capacità del modello di effettuare una segmentazione automatica e la valutazione della sua capacità di generalizzazione sul test set.



2.1 SEGMENTAZIONE MANUALE

Per poter addestrare la rete è necessario avere un dataset contenente due cartelle: una con le immagini istologiche e una con le rispettive maschere.
L'anatomopatologo ci ha fornito 108 immagini istologiche con le seguenti caratteristiche:

- Formato: jpg o tiff
- Dimensioni: 2048x1536
- Colorazione: ematossilina eosina

Queste immagini sono state acquisite da una telecamera che ha fotografato i vetrini con un ingrandimento 10x.
In seguito, le immagini sono state caricate sulla piattaforma Apeer per poter fare una segmentazione manuale.
Apeer consente di caricare l'immagine, sia in formato jpg che czi, fare delle annotazioni creando diverse classi ed infine esportare la corrispettiva maschera.
Le immagini istologiche di un tratto di intestino sano presentano cellule gangliari e tronchi normali, mentre in immagini di intestino malato si ha assenza di cellule gangliari e tronchi ipertrofici.
Per questo lavoro di tesi non sono state utilizzate le immagini istologiche della zona di transizione.
Sono state dunque create 4 classi:

- Tronchi ipertrofici
- Tronchi normali
- Cellule gangliari
- Background

L'anatomopatologo ha analizzato ogni immagine evidenziando manualmente le varie classi.

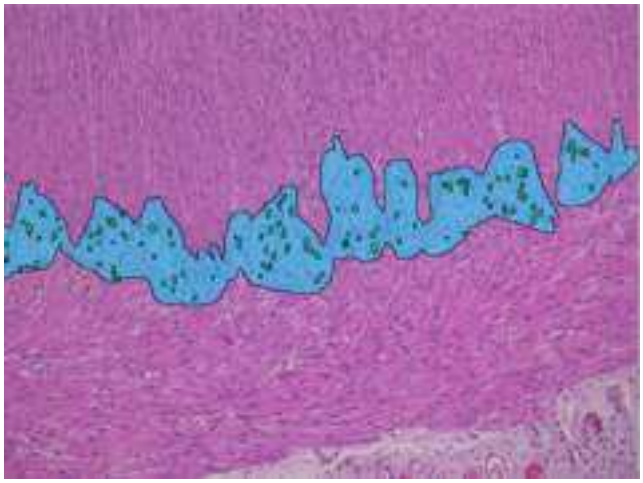


Figura 20: Tratto di intestino sano, evidenziate cellule gangliari (in verde) e tronchi nervosi normali (azzurro)

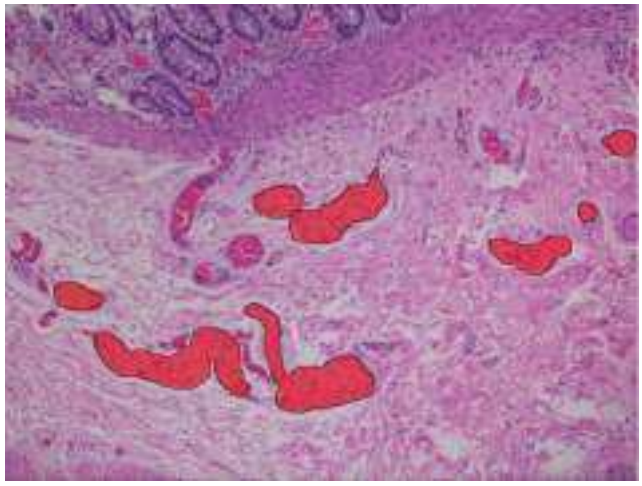


Figura 21: Tratto di intestino malato, evidenziati tronchi nervosi ipertrofici (rosso)

Dopo questo passaggio abbiamo provveduto ad esportare le annotazioni fatte su ogni immagine come maschera multiclasse includendo tutte le classi.

Per ogni immagine istologica viene dunque creato un file tiff, le cui annotazioni di ogni classe selezionata sono mappate allo stesso valore.

In alcune immagini istologiche in basso a destra è presente un'etichetta che indica la *scale bar*, che non serve ai fini dell'addestramento della rete neurale. Per questo motivo abbiamo pensato di tagliare la parte inferiore delle immagini istologiche e di conseguenza anche delle maschere.

Si ottengono dunque due dataset, uno per le immagini istologiche e uno per le maschere, con le stesse dimensioni: 2048x1280.

2.2 DATA AUGMENTATION

I modelli di deep learning dipendono principalmente dalla disponibilità di enormi set di dati. Senza l'esistenza di molte immagini in dataset, diversi modelli di deep learning non sarebbero in grado di apprendere e produrre modelli accurati. Sfortunatamente diversi campi, come l'elaborazione di immagini mediche, non hanno accesso a grandi quantità di dati.

In particolare, le reti neurali convoluzionali hanno, tipicamente, un numero di parametri apprendibili dell'ordine del milione, data la complessità dei *task* che viene loro richiesto di eseguire. Per ottenere buone performance, sarebbe dunque necessario avere un numero di esempi proporzionale.

Nel contesto della classificazione di immagini, quando il *training set* a disposizione ha una cardinalità ristretta, un metodo semplice ma efficace è rappresentato dalla *data augmentation*, una tecnica che consiste nel processare i dati a disposizione elaborandoli mediante trasformazioni cosiddette *label-preserving* [20].

La data augmentation è estremamente necessaria per i seguenti motivi:

1. Si tratta di una metodologia poco costosa se confrontata con la regolare raccolta di dati con l'annotazione dell'etichetta.
2. È estremamente accurata, in quanto i dati vengono generati da quelli originali.
3. È controllabile, il che aiuta a generare dati equilibrati.
4. Contribuisce a superare il problema dell'overfitting.
5. Aiuta a raggiungere una migliore accuratezza dei test.
6. Incrementa la capacità di generalizzazione di un modello.

Le operazioni di data augmentation più comuni consistono nel tagliare, ruotare, riflettere o aggiungere rumore gaussiano alle immagini. Le reti convoluzionali sono invarianti alla traslazione, perciò non avrebbe senso fare *data augmentation* con operazioni di traslazione.

Poiché nel progetto in esame il *training set* aveva cardinalità molto limitata, 108 immagini, per incrementarne la numerosità è stata utilizzata la tecnica di *data augmentation*.

Per ciascuna immagine, quindi, sono state ottenute altre quattro mediante operazioni di riflessione (*flip*) sia in direzione verticale che orizzontale, di rotazione di 90° e di diminuzione della saturazione.

In particolare è stato utilizzato il modulo tf.image contenente varie funzioni per l'elaborazione delle immagini e le operazioni di codifica-decodifica.

Originale

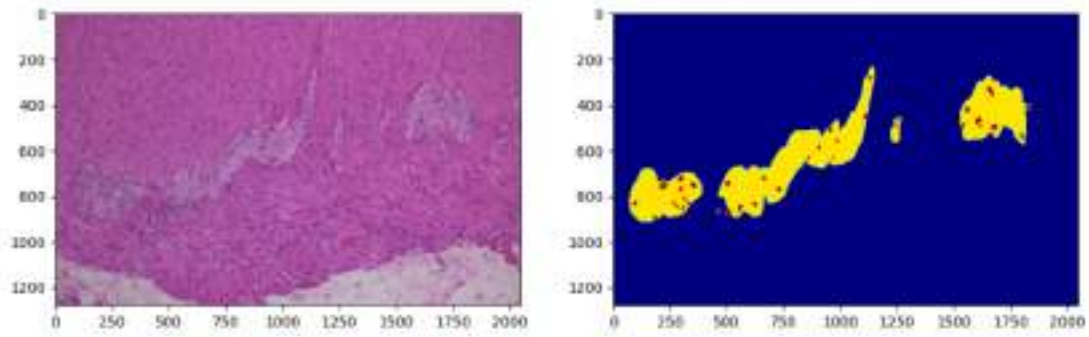


Figura 22: Immagine e maschera originale

Flipping

Riflette un'immagine attorno al suo asse verticale o orizzontale o sia verticale che orizzontale. Aiuta gli utenti a massimizzare il numero di immagini in un set di dati senza bisogno di alcuna elaborazione artificiale.

Flipping verticale: l'immagine viene ruotata sottosopra in modo che l'asse y sia in alto e l'asse x in basso

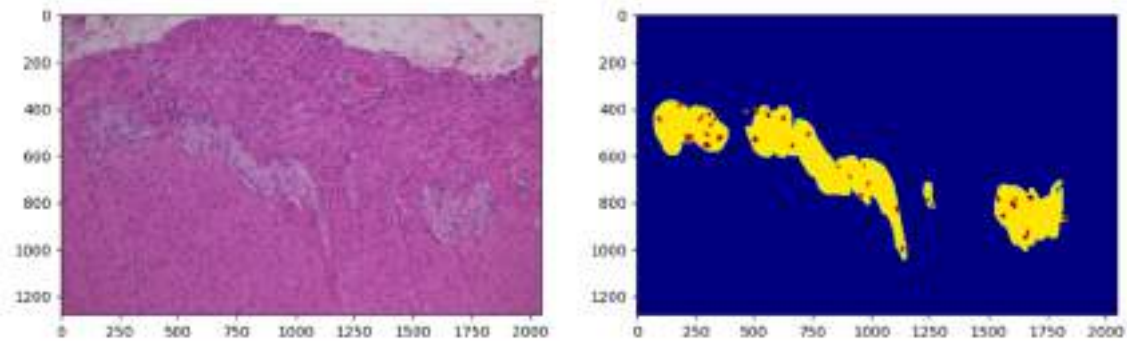


Figura 23: Flipping verticale

Flipping orizzontale: l'immagine viene ruotata orizzontalmente nei lati sinistro e destro

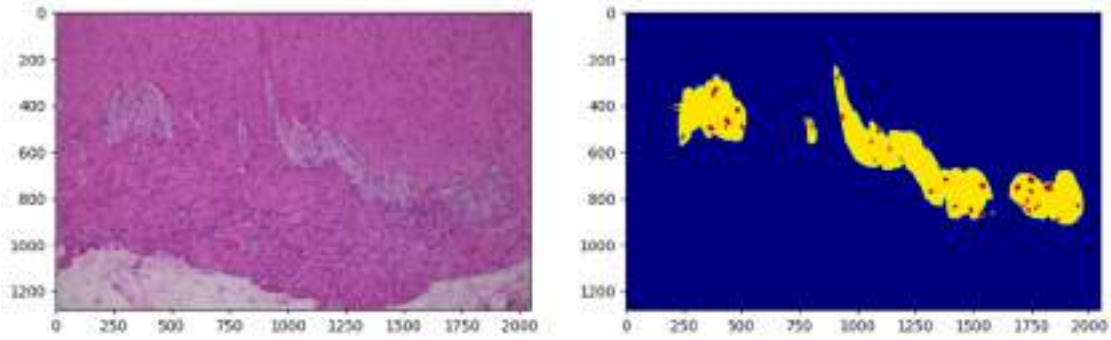


Figura 24: Flipping orizzontale

Rotazione

La rotazione è un altro tipo di aumento dei dati dell'immagine geometrica classica; il processo di rotazione viene fatto ruotando l'immagine attorno a un asse sia nella direzione destra che in quella sinistra di angoli tra 1° e 359°.

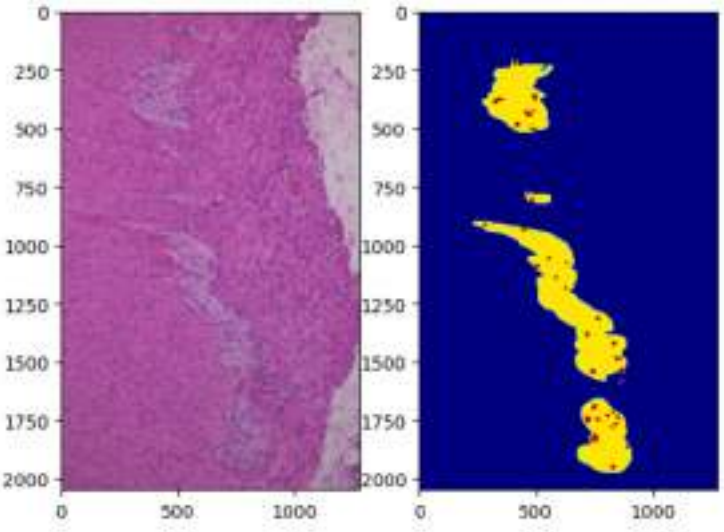


Figura 25: Rotazione di 90°

Saturazione

È stato scelto di utilizzare questo tipo di operazione perché i campioni se non vengono analizzati subito possono perdere colorazione e risultare sbiaditi.

La saturazione dell'immagine è stata diminuita a 0.5; la maschera corrispettiva rimane invariata rispetto a quella originale.

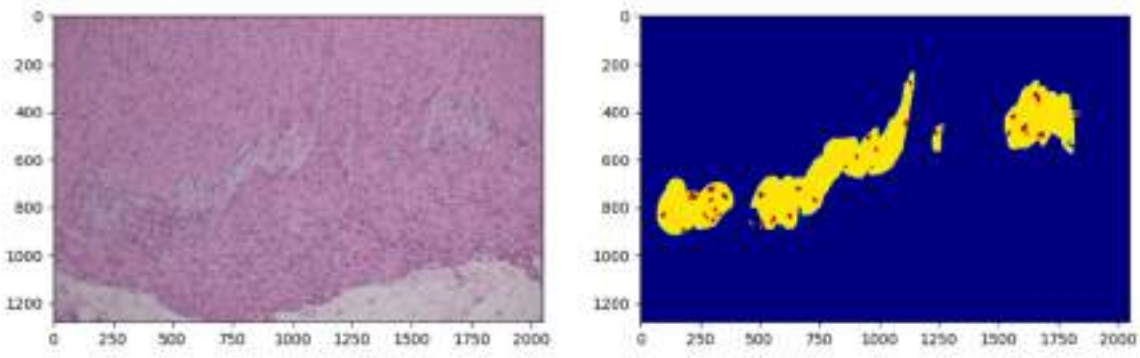


Figura 26: Saturazione

In seguito a queste operazioni il dataset è dunque aumentato e si è arrivati ad avere 540 immagini con le corrispettive maschere. Di queste ne vengono tolte 50 da utilizzare in seguito all'addestramento della rete per testare l'algoritmo e che vanno dunque a costituire il *test set*. Ne rimangono 490.

Durante l'addestramento di qualsiasi algoritmo di deep learning, si preferisce utilizzare immagini piccole perché producono risultati migliori. Se però si hanno immagini grandi, come nel nostro caso, una soluzione è dividerle in patch più piccole, permettendoci di addestrare qualsiasi algoritmo. Questo consente anche di incrementare ulteriormente i dati a disposizione.
Abbiamo un'immagine di dimensioni 2048x1280 pixel che può essere suddivisa in 40 patch quadrati di 256×256 pixel ciascuno.

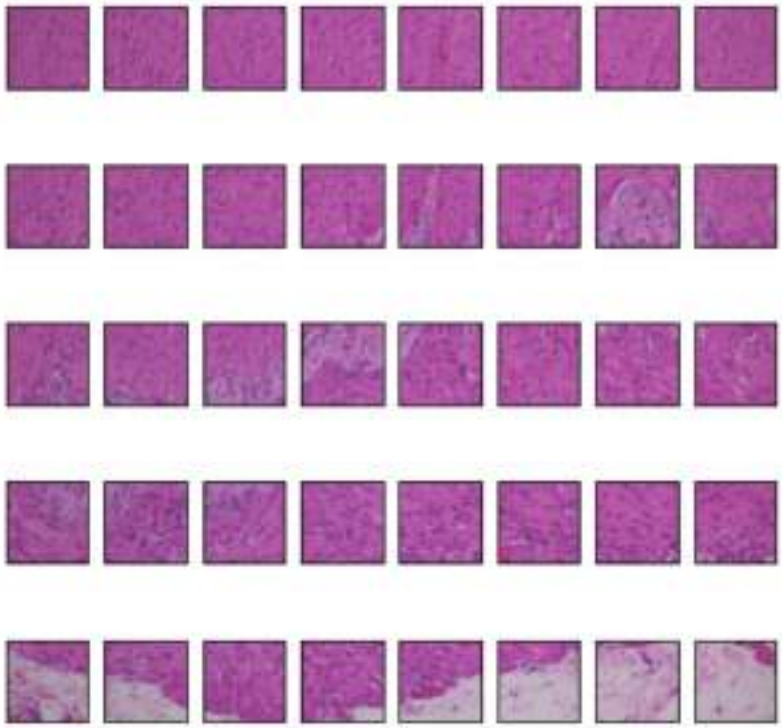


Figura 27: Immagine suddivisa in 40 patch

Ovviamente viene fatta la stessa cosa anche per le maschere.

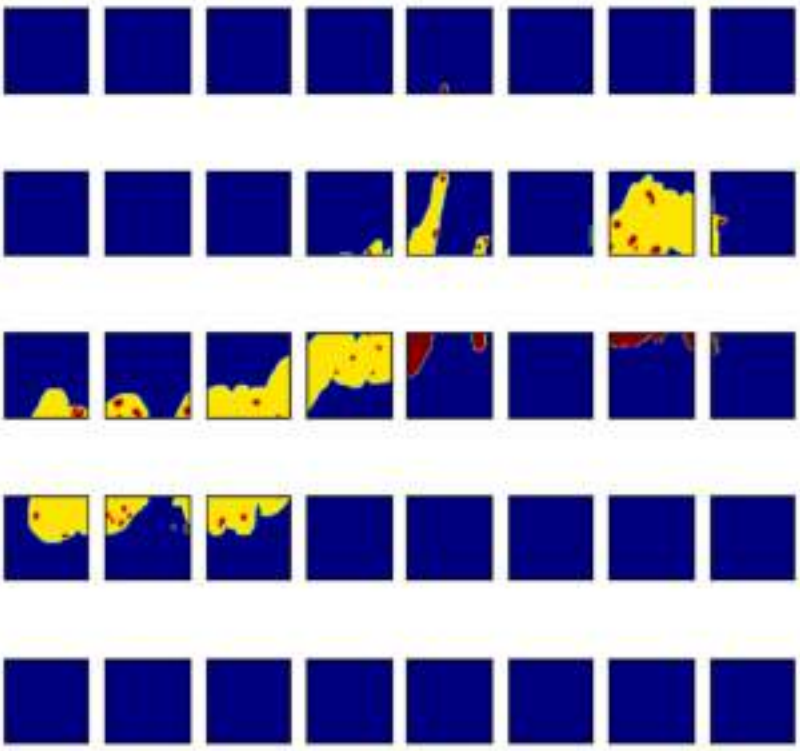


Figura 28: Maschera suddivisa in 40 patch

In questo modo il dataset aumenta moltissimo e si ottengono 19600 patch per le immagini e 19600 patch per le maschere.
Questo ampio dataset è stato utilizzato in vari modi ai fini nell'addestramento della rete, ossia sono stati creati diversi dataset contenenti solo immagini con determinate caratteristiche, così da ottenere vari modelli di rete neurale.
In particolare, è stato creato un dataset contenente solo immagini di cellule gangliari e tronchi normali e uno con soli tronchi ipertrofici. È necessario inserire nei due dataset precedenti anche delle immagini caratterizzate solo da background per far si che la rete impari a riconoscere non solo le zone target da classificare ma anche il contesto dell'immagine in cui non saranno mai presenti cellule e tronchi normali, piuttosto che tronchi ipertrofici.

Riassumendo i dataset ottenuti contengono immagini con le seguenti caratteristiche:

- *Dataset 1*: Cellule gangiari e tronchi normali
- *Dataset 2*: Tronchi ipertrofici

2.3 TRAINING MODELLO

2.3.1 SUDDIVISIONE DATASET

Prima di iniziare l'allenamento è stato necessario suddividere ciascun dataset in:

- *training set*: corrispondente all'80% dell'intero dataset, è utilizzato per addestrare la rete
- *validation set*: è il 20%, è utilizzato per valutare la perdita ed eventuali metriche del modello al termine di ogni epoca. Il modello non viene addestrato su questi dati.

Questa procedura viene fatta in modo casuale tramite la funzione "train test split".

```
X_train, X_test, y_train, y_test = train_test_split(image_dataset,
    mask_dataset_encoded, test_size = 0.2, random_state = 42)
```

Questa funzione, infatti, prende come ingressi i vettori, image_dataset e mask_dataset_encoded, e un "test size": quest'ultimo parametro rappresenta una percentuale (espressa come numero compreso tra 0 e 1) che indica la quantità di dati del dataset che verrà utilizzata per la fase di test; per ricavare quindi la parte che verrà utilizzata durante l'addestramento addestramento sarà sufficiente sottrarre il valore test size da 1 (1−test size).
Da notare come la funzione ritorni quattro elementi, e non due come ci si aspetterebbe. Durante la fase di addestramento viene elaborato il vettore X_train, per verificare se un risultato è o meno corretto, si compara il dato in uscita dalla rete con quello di y_train: se sono uguali allora il risultato è corretto, altrimenti il risultato necessita di correzioni. Stessa cosa avviene nella fase di test.

2.3.2 CREAZIONE RETE

Inizialmente vengono creati i due blocchi:

- *Encoder block*: convolutional block + maxpooling

```
def conv_block(input, num_filters):
    x = Conv2D(num_filters, 3, padding="same")(input)
    x = BatchNormalization()(x)
    x = Activation("relu")(x)

    x = Conv2D(num_filters, 3, padding="same")(x)
    x = BatchNormalization()(x)
    x = Activation("relu")(x)

    return x
```

```
def encoder_block(input, num_filters):
    x = conv_block(input, num_filters)
    p = MaxPool2D((2, 2))(x)
    return x, p
```

Il blocco convoluzione è costituito da due convoluzioni 3x3, ciascuna seguita da una BatchNormalization e da una funzione di attivazione Activation ReLU (unità lineare rettificata). Nell'operazione di convoluzione si imposta Padding = "same" per avere lo stesso output a seguito della convoluzione.

A questo blocco segue l'operazione di max pooling 2x2 con passo 2 per il downsampling e insieme formano l'encoder block.

- *Decoder block:*

```
def decoder_block(input, skip_features, num_filters):
    x = Conv2DTranspose(num_filters, (2, 2), strides=2, padding="same")(input)
    x = Concatenate()([x, skip_features])
    x = conv_block(x, num_filters)
    return x
```

È costituito da una Conv2DTranspose 2x2, una concatenazione con la corrispondente feature map ritagliata dall'encoder path e il blocco convoluzionale visto prima.

La Conv2DTranspose è un up-sampling seguito da una convoluzione

Di seguito è riportato come vengono sviluppati i due percorsi di contrazione ed espansione tramite i blocchi encoder e decoder precedentemente definiti. Il quinto blocco encoder rappresenta la base dell'architettura a U e non è seguito da Max Pooling ma, come si vede, è costituito solo dal blocco convoluzionale.

```
def build_unet(input_shape, n_classes):
    inputs = Input(input_shape)

    s1, p1 = encoder_block(inputs, 64)
    s2, p2 = encoder_block(p1, 128)
    s3, p3 = encoder_block(p2, 256)
    s4, p4 = encoder_block(p3, 512)

    b1 = conv_block(p4, 1024)

    d1 = decoder_block(b1, s4, 512)
    d2 = decoder_block(d1, s3, 256)
    d3 = decoder_block(d2, s2, 128)
    d4 = decoder_block(d3, s1, 64)

    if n_classes == 1:
      activation = 'sigmoid'
    else:
      activation = 'softmax'

    outputs = Conv2D(n_classes, 1, padding="same", activation=activation)(d4)
    print(activation)

    model = Model(inputs, outputs, name="U-Net")
    return model
```

La funzione di attivazione applicata all'output è diversa dalle precedenti. In base al compito che svolge deve essere selezionata una funzione di attivazione appropriata: per un'attività di classificazione multiclasse viene utilizzata una funzione Softmax, se invece la classificazione è binaria la funzione di attivazione è Sigmoid.

Questo modello in totale ha 27 layers:

- 18 convoluzione 3x3
- 4 max pooling
- 4 up-convolution 2x2

- 1 convoluzione 1x1

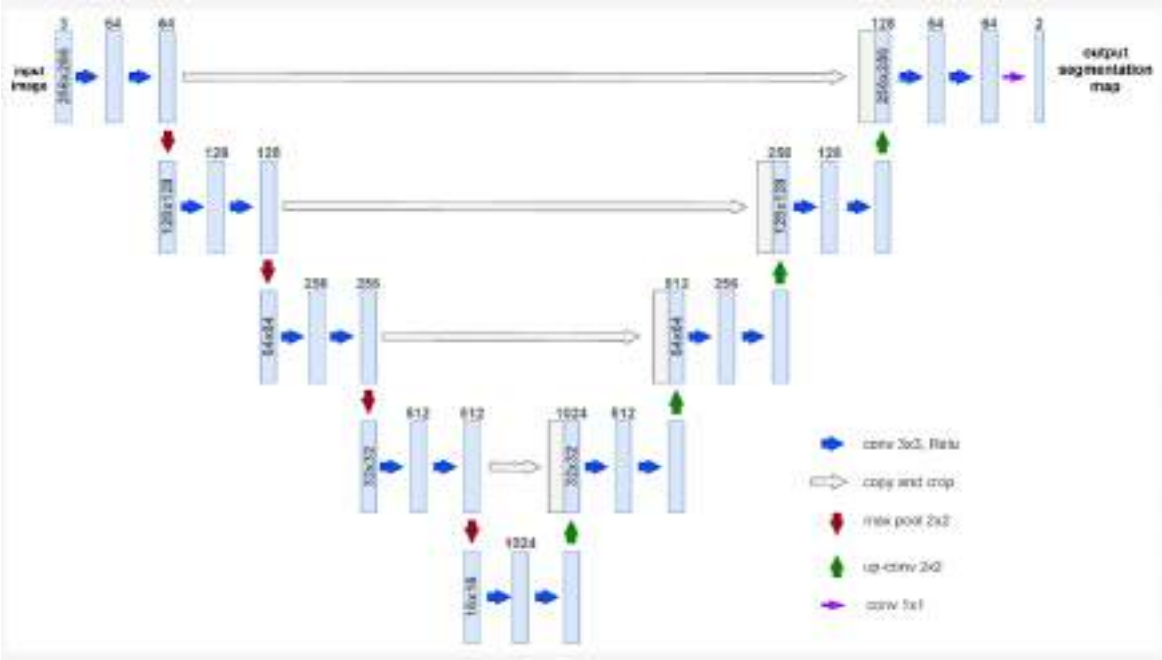


Figura 29: Architettura U-Net con in input un'immagine di dimensioni 256x256 e classe 2

L'immagine originale ha dimensione 256x256 e vengono applicati 64 filtri. In seguito al max pooling l'immagine diminuisce di dimensioni, si passa a 128x128 ma il numero dei filtri raddoppia a 128. Dopo un totale di 4 blocchi encoder si arriva alla base in cui si ha un'immagine di dimensioni 16x16 e 1024 filtri. Si applicano ancora due convoluzioni e poi si passa alla fase di decoder in cui i filtri diminuiscono ma le dimensioni aumentano. L'uscita dell'ultimo livello di espansione avrà dunque le stesse dimensioni trasversali (altezza e larghezza, 256x256) dell'immagine in ingresso alla rete ma il numero di canali sarà maggiore, pari al numero di filtri dell'ultima convoluzione (16).
Per far sì che il numero di canali corrisponda al numero di classi in esame per la segmentazione si applica uno strato convoluzionale 1x1 il cui numero di filtri è pari al numero di classi.
Il numero di classi nei dataset con immagini di tronchi normali e cellule gangliari è tre, mentre in quelli con tronchi ipertrofici è due.

2.3.3 ALLENAMENTO RETE

La prima cosa da fare è configurare il modello per l'addestramento tramite il metodo model.compile come si vede di seguito.

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

- ***Optimizer:*** stringa (nome dell'ottimizzatore) o istanza dell'ottimizzatore.
  - *Adam*: l'ottimizzazione di Adam è un metodo di discesa del gradiente stocastico che calcola learning rate individuali per diversi parametri basandosi sulla stima adattiva dei momenti del primo e del secondo ordine dei gradienti; questo metodo ha pochi requisiti di memoria, è invariante rispetto al ridimensionamento diagonale dei gradienti ed è adatto per problemi di grandi dimensioni in termini di dati/parametri.
- ***Loss:*** funzione di perdita.
  - *Categorical crossentropy*: calcola la perdita di entropia incrociata tra le maschere e le previsioni; è utilizzata per la classificazione multiclasse e restituisce un vettore con il numero 1 nella posizione del risultato più probabile.
- ***Metrics:*** elenco delle metriche che devono essere valutate dal modello durante il training e il test.
  - *Accuracy*: calcola la frequenza con cui le previsioni corrispondono alle maschere.

Di conseguenza si addestra il modello per un numero fisso di epoche (iterazioni del dataset) indicando i dati da utilizzare in fase di allenamento e quelli da usare in fase di validazione.
È importate scegliere un appropriato numero di epoche: un valore troppo ridotto potrebbe compromettere le prestazioni del modello sui dati di addestramento (underfitting), mentre un numero troppo elevato potrebbe condurre l'algoritmo a memorizzare i dati usati per l'allenamento, perdendo la capacità di generalizzazione (overfitting).
Per addestrare il modello si utilizza la funzione *fit*:

```
history = model.fit(X_train, y_train_cat,
                    batch_size = 16,
                    verbose=1,
                    epochs=50,
                    validation_data=(X_test, y_test_cat),
                    shuffle=False)
```

- X_train: dati in input
- y_train_cat: dati targhet (maschere)
- Batch_size: è il numero di campioni dopo cui viene aggiornato il gradiente.
- Verbose: modalità verbose utilizzata è 1 che corrisponde alla barra di avanzamento.
- Epochs: numero di epoche utilizzate per addestrare il modello; un'epoca è un'iterazione su tutti i dati X_train e y_train forniti
- Validation_data: dati su cui valutare la perdita ed eventuali metriche del modello al termine di ogni epoca; il modello non viene addestrato su questi dati.
- Shuffle: è un parametro di modellazione che consente di mescolare i dati di addestramento prima di ogni epoca; questo parametro deve essere impostato su *false* se i dati sono serie temporali.

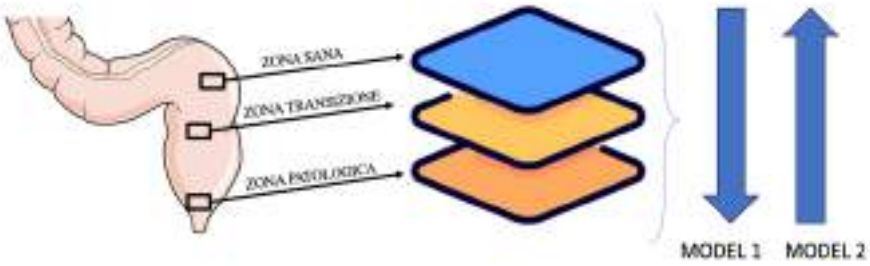
Essendoci 2 dataset diversi sono stati allenati 2 modelli diversi

- MODELLO 1: per riconoscere tronchi normali e cellule gangliari
- MODELLO 2: per riconoscere tronchi ipertrofici

2.4 APPLICAZIONE DELLA RETE NEURALE

Nella pratica clinica il tratto di intestino resecato durante l'intervento viene inviato in anatomia patologica e viene fatta l'analisi istopatologica di tutto il pezzo per quantificare la lunghezza delle varie zone in quanto forniscono importanti informazioni cliniche.
Il numero delle immagini dipende dalla lunghezza del pezzo, in genere varia tra le 30 e le 50 immagini.
Queste immagini seguono una sequenza: partendo dalla zona sana dell'intestino prossimale troveremo immagini con tronchi normali e cellule gangliari, a seguire una zona di transizione con cellule gangliari e tronchi ipertrofici e infine una zona distale con solo immagini e tronchi ipertrofici. Nelle varie zone è possibile talvolta trovare immagini in cui non ci sia nulla, solo background.

Di seguito uno schema di come l'analisi con le reti neurali possa fungere da strumento di supporto per accelerare il processo di analisi.



All'intero dataset fornito dall'anatomopatologo vengono applicati prima il modello 1 e poi il modello 2.
Applicando il modello 1 si dovrebbe ottenere la seguente segmentazione:

- Segmentazione di tronchi normali e cellule gangliari per le immagini appartenenti alla zona sana
- Segmentazione di cellule gangliari per le immagini appartenenti alla zona di transizione
- Nessuna segmentazione per le immagini appartenenti alla zona patologica
- Nessuna segmentazione per le immagini con colo background

Applicando il modello 2 invece si dovrebbe ottenere la seguente segmentazione:

- Nessuna segmentazione per le immagini appartenenti alla zona sana
- Segmentazione dei tronchi ipertrofici per le immagini appartenenti alla zona di transizione
- Segmentazione dei tronchi ipertrofici per le immagini appartenenti alla zona patologica
- Nessuna segmentazione per le immagini con colo background

Risulteranno dunque delle immagini che sono state segmentate sia dal modello 1 che dal modello 2 e che appartengono alla zona di transizione e che dovranno essere analizzate in modo più dettagliato anche dall'anatomopatolo.

3 RISULTATI

In questo capitolo, saranno presentati i risultati ottenuti dall'addestramento della rete. Dopo ogni fase di addestramento, è stata testata la capacità del modello di eseguire una segmentazione automatica, seguita da una valutazione della sua capacità di generalizzazione utilizzando il test set, che comprende immagini mai visualizzate in precedenza dalla rete.

Come descritto nel capitolo precedente, sono stati utilizzati diversi dataset per l'addestramento della rete, il che ha comportato la creazione di diversi modelli. In tutti i casi, l'accuratezza è stata la metrica utilizzata per valutare le prestazioni del modello.

3.1 SEGMENTAZIONE AUTOMATICA

Al termine dell'allenamento si applica al modello la funzione *evaluate*, una funzione che restituisce i valori delle metriche del modello. Nel nostro caso viene valutata l'*Accuracy*:

```
_, acc = model.evaluate(X_test, y_test_cat)
print("Accuracy = ", (acc * 100.0), "%")
```

l'accuratezza fornisce un'indicazione sulla vicinanza del valore predetto al valore reale o corretto.
Questo valore può essere calcolato dividendo il numero di previsioni corrette per il numero totale di previsioni effettuate.

$$Accuracy = \frac{correct\ prediction}{all\ prediction}$$

Può essere considerata una buona accuratezza quando si raggiunge un valore superiore al 90%.

Per testare la capacità del modello di effettuare una segmentazione automatica si utilizza la funzione *predict* che genera previsioni per ogni campione dato in input.

```
y_pred=model.predict(X_test)
```

Nel nostro caso l'input è X_test, ossia le immagini che appartengono al *Validation set*, il 20% del dataset di partenza su cui il modello non viene addestrato.
Questo dataset viene utilizzato anche per valutare l'*Accuracy* del modello al termine di ogni epoca.

3.1.1 MODELLO 1

Per addestrare questo modello sono state utilizzate immagini con tronchi normali e cellule gangliari, immagini con solo cellule e immagini con solo background.
Le classi in esame per questa segmentazione sono tre e corrispondono a tronchi normali, cellule gangliari e background.
In seguito all'operazione di train_test_split il dataset di partenza composto da 930 immagini viene così suddiviso:

- Training set: 744 immagini
- Validation set: 186 immagini

Il numero di epoche utilizzate per l'allenamento è 120.
Il grafico seguente riporta l'andamento dei valori di *Accuracy* del training set e del validation set al termine di ogni epoca.

Figura 30: Grafco dei valori di Training e Validation Accuracy

Anche in questo caso, osservando il grafico, è evidente che i valori raggiungono un plateau, indicando che il numero di epoche utilizzate per l'addestramento è sufficiente.

Il valore di *Accuracy* del modello calcolato con la funzione *evaluate* è:

```
Accuracy = 92.32 %
```

Di seguito vengono riportate alcune predizioni ottenute con la funzione *predict*:

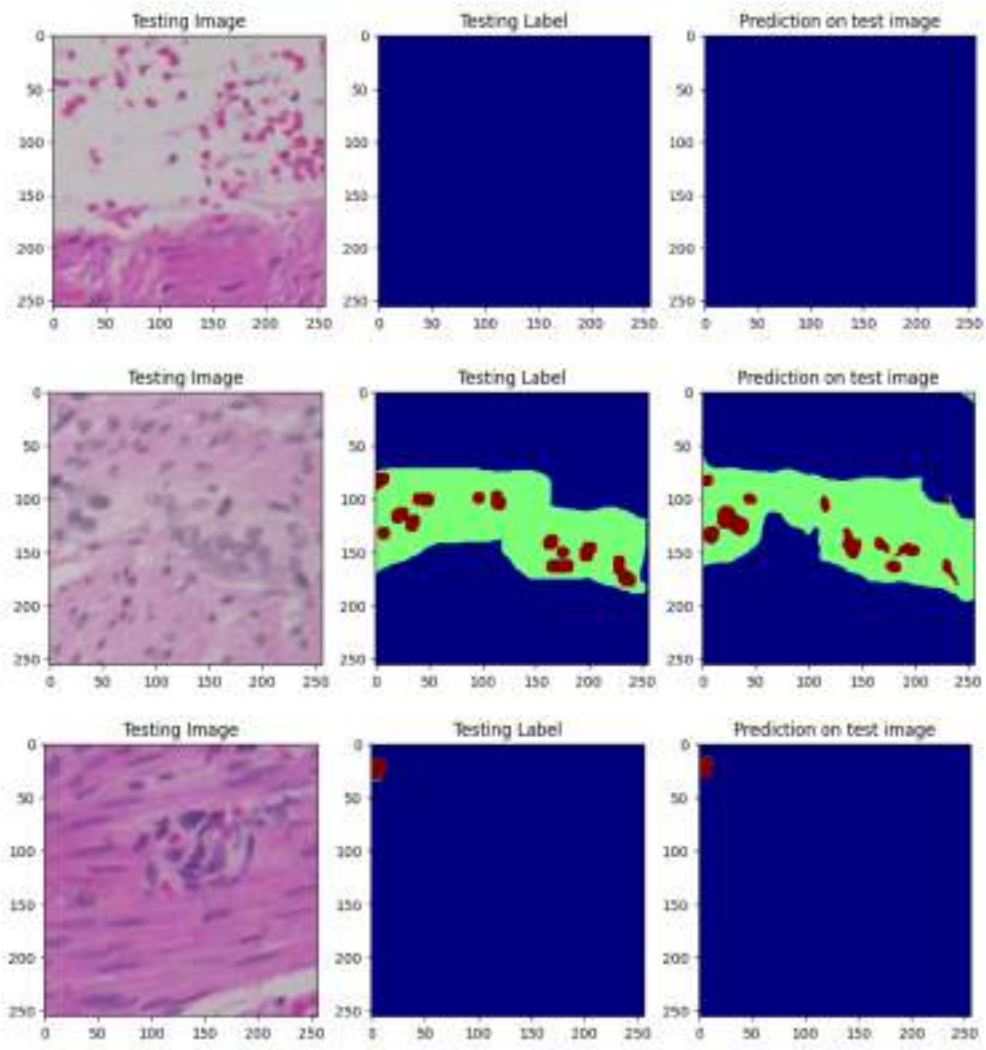


Figura 31: "Testing imag"e sono le immagini appartenenti al validation set, "Testing label" sono le corrispettive maschere segmentate manualmente dall'anatomopatologo e "Prediction on test imag" e sono le predizioni generate dal modello

In blu è classificato il background, in verde il tronco normale e in bordeaux le cellule gangliari.

È un risultato positivo il fatto che nella prima immagine non sia stata eseguita alcuna segmentazione e che nella terza immagine sia stata rilevata correttamente solo l'unica cellula gangliare presente. Come si può notare il modello fa più fatica a segmentare correttamente le cellule gangliari rispetto che ai tronchi.

Tuttavia, per il nostro scopo non è importante che le predizioni siano identiche alle maschere, ma che vengano segmentate correttamente. In altre parole, non importa che le cellule segmentate siano più piccole e di numero inferiore, ma che il modello abbia predetto correttamente la presenza delle tre classi.

3.1.2 MODELLO 2

Per addestrare questo modello sono state utilizzate solo immagini con tronchi ipertrofici.
Le classi in esame per questa segmentazione sono due e corrispondono a tronchi ipertrofici e background.
In seguito all'operazione di train_test_split il dataset di partenza composto da 1015 immagini viene così suddiviso:

- Training set: 812 immagini
- Validation set: 203 immagini

Il numero di epoche utilizzate per l'allenamento è 150.
Il grafico seguente riporta l'andamento dei valori di *Accuracy* del training set e del validation set al termine di ogni epoca.

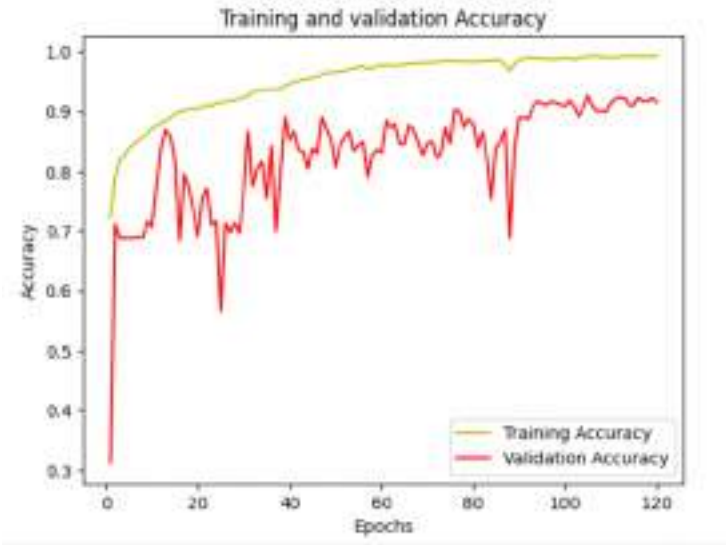


Figura 32: Grafco dei valori di Training e Validation Accuracy

Anche in questo caso osservando il grafico si nota che i valori raggiungo un plateau e quindi il numero di epoche utilizzate è sufficiente per l'addestramento.

Il valore di *Accuracy* del modello calcolato con la funzione *evaluate* è:

```
Accuracy = 91.50 %
```

Di seguito vengono riportate alcune predizioni ottenute con la funzione *predict*:

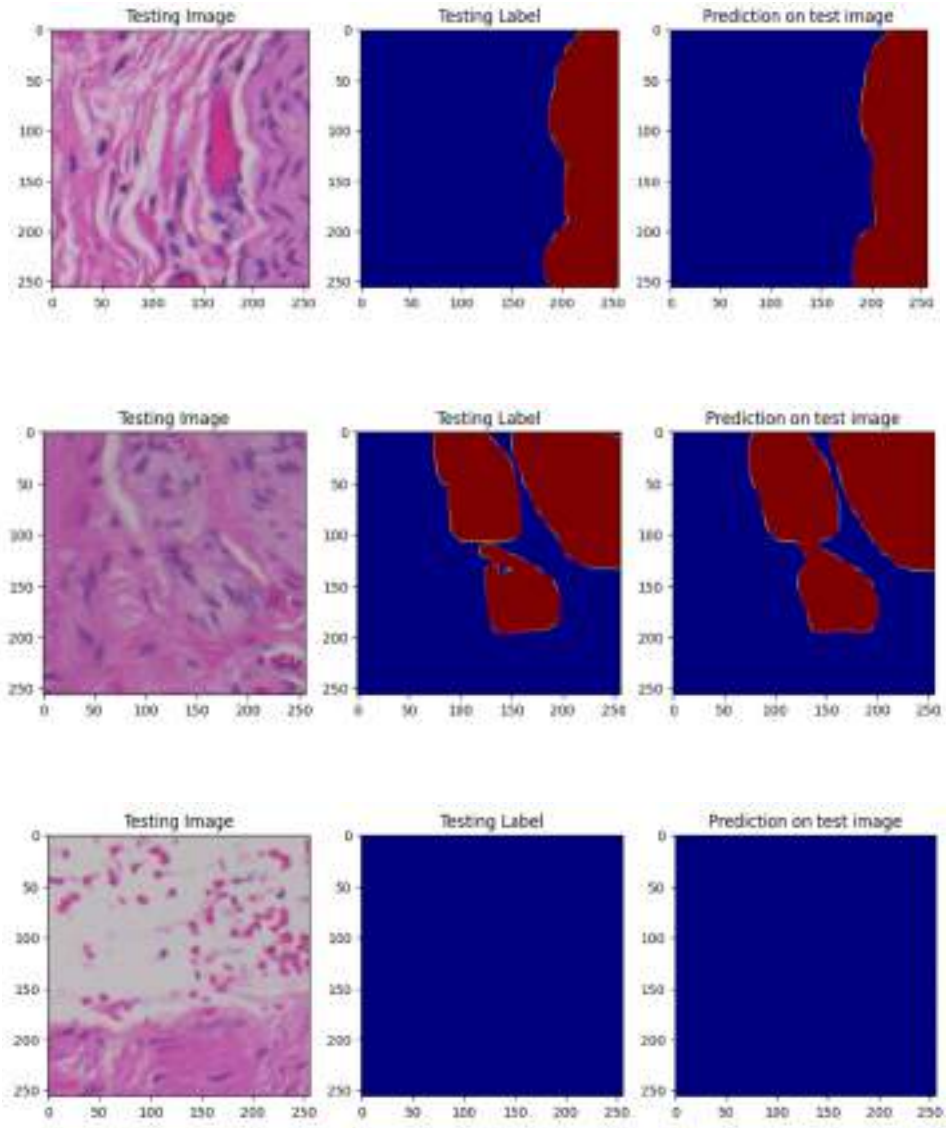


Figura 33: "Testing image" sono le immagini appartenenti al validation set, "Testing label" sono le corrispettive maschere segmentate manualmente dall'anatomopatologo e "Prediction on test image" sono le predizioni generate dal modello

In blu è classificato il background e in bordeaux i tronchi ipertrofici.

Come si può vedere la differenza fra le due maschere è lieve, la segmentazione dei soli tronchi ipertrofici risulta più semplice.
Anche per questo modello l'aspetto fondamentale è che abbia predetto correttamente la presenza delle due classi.

3.2 VALUTAZIONE SU TEST SET

Durante questa fase, l'obiettivo è simulare la pratica clinica in cui le immagini istologiche fornite dall'anatomopatologo vengono sottoposte ai modelli per generare una segmentazione per ciascuna immagine.

Per questa prova sono state utilizzate le immagini del *test set* che hanno dimensioni di 2048x1280 e che non sono mai state presentate alla rete, neanche per la validazione.
In questo dataset ci sono anche le corrispettive maschere segmentate manualmente dall'anatomopatologo che servono solo per la verifica.

Per ottenere la segmentazione di queste immagini è necessario caricare nel sistema il modello che si vuole applicare e dare come input alla funzione *predict* le immagini del *test set*. Il modello è stato addestrato su immagini con dimensioni specifiche di 256x256 pixel, per questo è necessario che le immagini in input abbiano la stessa dimensione per ottenere risultati ottimali.
È quindi necessario seguire i seguenti passaggi:

- dividere ogni immagine istologica in quaranta patch ciascuna con dimensione 256x256 (come è stato fatto nella fase di *data augmentation*)
- applicare la funzione *predict* a ogni patch
- riunire le patch predette mediante la funzione *unpatchify* generando la maschera di dimensione 2048x1280

In questo test set sono presenti solo immagini di zona sana o zona patologica.

Come descritto nel capitolo 2.4 si applica prima il MODELLO 1 e poi il MODELLO 2.

Applicando il MODELLO 1 alle immagini della zona sana esse vengono segmentate correttamente.

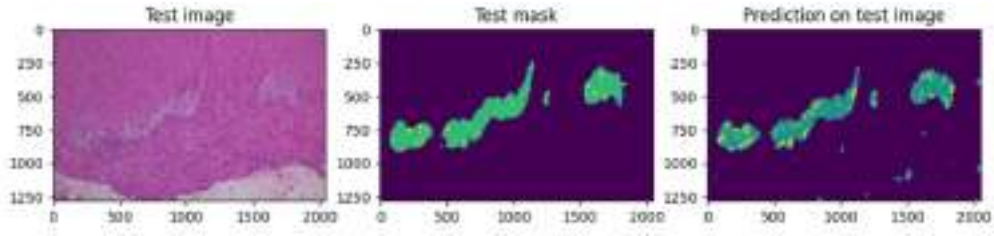


Figura 34: Immagine istologica, maschera segmentata manualmente, maschera segmentata dal modello 1

Si può notare un'eccessiva segmentazione delle cellule gangliari, questo perché il modello fatica a distinguere in modo accurato le cellule gangliari dalle altre cellule presenti nell'immagine.

Applicando il MODELLO 1 alle immagini della zona patologica ci si dovrebbe aspettare solo background in quanto il modello è stato addestrato specificamente per riconoscere tronchi normali e cellule gangliari.

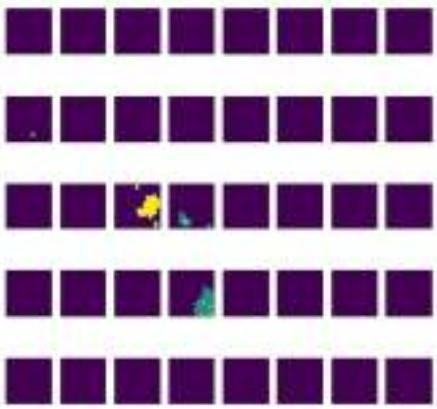


Figura 35: segmentazione di patch di una zona patologica

Dalla figura si può notare che solo 4 patch su 40 non sono state segmentate in modo corretto. L'errata classificazione di esse è causata dal fatto che il modello ha classificato come cellule gangliari alcune cellule che non lo sono e ha individuato dei possibili tronchi normali dove invece c'erano tronchi ipertrofici.

Applicando il MODELLO 2 alle immagini della zona patologica esse vengono segmentate correttamente.

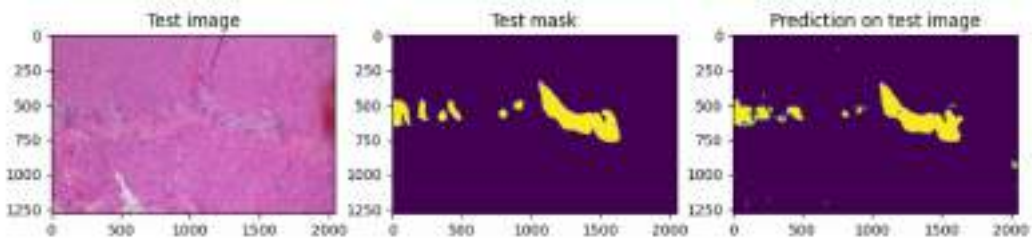


Figura 36: Immagine istologica, maschera segmentata manualmente, maschera segmentata dal modello 2

Applicando il MODELLO 2 alle immagini della zona sana ci si dovrebbe aspettare solo background in quanto il modello è stato addestrato specificamente per riconoscere tronchi ipertrofici.

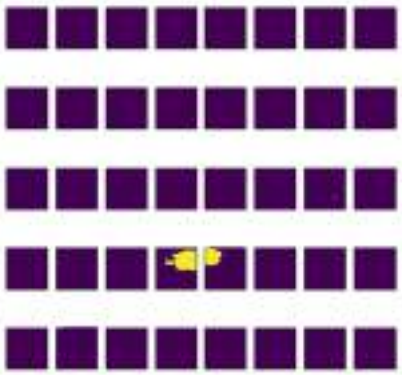


Figura 37: segmentazione di patch di una zona sana

Come si può vedere solo 3 patch su 40 sono stati classificati in modo errato in quanto il modello ha predetto un tronco ipertrofico dove in realtà c'è un tronco normale.

Pertanto, risulta evidente che a volte si riscontrano difficoltà nella corretta segmentazione delle cellule gangliari, in quanto vengono confuse con altre cellule presenti nell'immagine istologica. Allo stesso modo, può verificarsi confusione anche tra i tronchi normali e quelli ipertrofici.

CONCLUSIONI

Per una diagnosi e un trattamento accurati della malattia di Hirschsprung (HD), è fondamentale valutare attentamente la densità delle cellule gangliari e l'ipertrofia dei nervi nelle sezioni istologiche. L'obiettivo è fornire uno strumento di supporto all'anatomopatologo in quanto l'analisi può risultare impegnativa, richiedere tempo ed essere soggetta a possibili errori.

Lo scopo di questa tesi è stato quello di sviluppare un algoritmo in grado di effettuare una segmentazione accurata delle immagini relative alle diverse zone individuabili nell'analisi del tratto intestinale resecato durante l'intervento chirurgico.

Nel presente studio, è stato impiegato un modello basato sulla rete U-net per eseguire una segmentazione pixel per pixel, consentendo l'identificazione delle diverse regioni e la loro classificazione in base alle caratteristiche apprese durante il processo di addestramento.

Il focus principale non è la ricerca di una segmentazione estremamente precisa, ma piuttosto assicurarsi una corretta segmentazione delle immagini istologiche.

La disponibilità di un dataset di partenza ampio rappresenta una sfida significativa per l'addestramento di algoritmi di intelligenza artificiale, soprattutto quando si lavora con immagini mediche e malattie rare come la malattia di Hirschsprung. L'utilizzo della Data Augmentation si è rivelato fondamentale per raggiungere il nostro obiettivo.

I risultati ottenuti evidenziano una buona accuratezza ottenuta durante l'addestramento di entrambi i modelli. Tuttavia, in alcuni casi, la rete riscontra difficoltà nel segmentare correttamente le cellule gangliari, in quanto si trova a dover distinguere tra queste e le altre cellule presenti nell'immagine istologica. Inoltre, la distinzione tra tronchi ipertrofici e normali può risultare confusa in determinate circostanze.
Pertanto, potrebbe essere vantaggioso acquisire nuove immagini istologiche e applicare nuovamente la tecnica di Data Augmentation per espandere ulteriormente il dataset.

Bibliografia

[1] Simon E. Kenny, BSc, ChB(Hons), MD, FRCS(Paed), FAAP,a Paul K. H. Tam, MBBS, ChM, FRCS, FHKAM,b Mercè Garcia-Barcelo, BSc (Hons), M Phil, PhDb, *Hirschsprung's disease (2010)*

[2] Prem Puri, *Hirschsprung's Disease and Allied Disorders (2019, Springer International Publishing)*

[3] Yun-jin Wang, Yuan-bin He, Liu Chen, Yu Lin, Ming-kun Liu and Chao-ming Zhou, *Laparoscopic-assisted Soave procedure for Hirschsprung disease: 10-year experience with 106 cases (2022)*

[4] Raj P. Kapur, MD, PhD, *Histology of the Transition Zone in Hirschsprung Disease (2016)*

[5] Laura V. Veras and Ankush Gosain, *Hirschsprung-Associated Enterocolitis (2019)*

[6] Gunadi, Ivana, G., Mursalin, D.A. et al., *Functional outcomes of patients with short-segment Hirschsprung disease after transanal endorectal pull-through (2021)*

[7] Prem Puri and Hiroki Nakamura, *Epidemiology and Clinical Characteristics of Hirschsprung's Disease (2019)*

[8] Yan BL, Bi LW, Yang QY, Wu XS, Cui HL*, Transanal endorectal pull-through procedure versus transabdominal surgery for Hirschsprung disease: A systematic review and meta-analysis (2019)*

[9] Jie-xiong Feng, Ting Li, and Ning Li, *Laparoscopically Assisted Pull-Through Operation for Hirschsprung's Disease (2019)*

[10] Atsuyuki Yamataka, Masahiro Takeda & Yuta Yazaki, *Transanal Pull-Through With or Without Laparoscopic Assistance for Rectosigmoid Hirschsprung's Disease (2019)*

[11] O. Berezsky, O. Pitsun, B. Derysh, I. Pazdriy, G. Melnyk and Y. Batko, *Automatic Segmentation of Immunohistochemical Images Based on U-net Architecture (2021)*

[12] N. Siddique, S. Paheding, C. P. Elkin and V. Devabhaktuni, *U-Net and Its Variants for Medical Image Segmentation: A Review of Theory and Applications (2021)*

[13] Getao Du, Xu Cao, Jimin Liang, *Medical image segmentation based on u-net: A review (2020)*

[14] Saad Albawi , Tareq Abed Mohammed, *Understanding of a Convolutional Neural Network (2017)*

[15] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A.W.M. van der Laak, Bram van Ginneken, Clara I. Sa´nchez, *A Survey on Deep Learning in Medical Image Analysis (2017)*

[16] Haykin S., *Neural Networks and Learning Machines (2009)*

[17] Seonwoo Min, Byunghan Lee and Sungroh Yoon, *Deep learning in bioinformatics (2017)*

[18] Yann LeCun, Yoshua Bengio & Geoffrey Hinton, *Deep learning (2015)*

[19] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do & Kaori Togashi, *Convolutional neural networks: an overview and application in radiology (2018)*

[20] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*

[21] Rikiya Yamashita1, Mizuho Nishio, Richard Kinh Gian Do2, Kaori Togashi, *Convolutional neural networks: an overview and application in radiology (2018)*