Training an encoder-decoder neural network with limited data for fetal femur segmentation from echographic images

Addestramento di una rete neurale encoder-decoder con dati limitati per la segmentazione del femore fetale da immagini ecografiche

Ollari Ischimji Dmitri

26 ottobre 2023

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

1.1 Binary Semantic Segmentation

La segmantazione semantica è una tecnica di *computer vision* che permette di assegnare ad ogni pixel di un'immagine un'etichetta che ne descrive il contenuto.

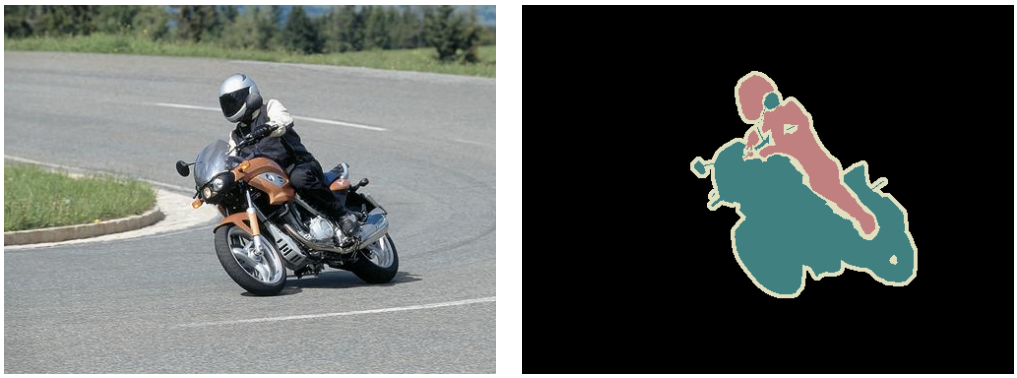

Figura 1.1: Segmentazione semantica (Fonte: [2])

Nello specifico in questa tesi si tratta una sottocategoria della semganzione semantica, ovvero la **Binary Semantic Segmentation**(segmentazione semantica binaria), questa tecnica di *computer vision* permette di assegnare ad ogni pixel di un'immagine un'etichetta che ne descrive il contenuto, ma a differenza della segmentazione semantica classica, che permette di assegnare ad ogni pixel una delle N possibili etichette, la segmentazione semantica binaria permette di assegnare ad ogni pixel una delle due etichette possibili: **oggetto** o **sfondo**.

La segmentazione è una tipologia di problema molto ricorrente in ambito medico, in quanto permette di automatizzare alcune procedure che altrimenti sarebbero eseguite manualmente, riducendo i tempi di esecuzione e i costi, permettendo di ottenere risultati più precisi e accurati limitando lo sforzo umano.

1.2 Fully Convolutional Network

Le **Fully Convolutional Network** (FCN) [6] sono una tipologia di reti neurali convoluzionali (CNN) che permettono di effettuare segmentazioni semantiche, in quanto sono in grado di gestire input di qualsiasi dimensione e di produrre mappe di segmentazione più precise grazie alla loro capacità di apprendere contesti spaziali.

Le motivazioni riguardanti l'ampio utilizzo nel settore della *computer vision* sono legate all'assenza di strati completamente connessi (lineari) che vincolano l'input alla medesima

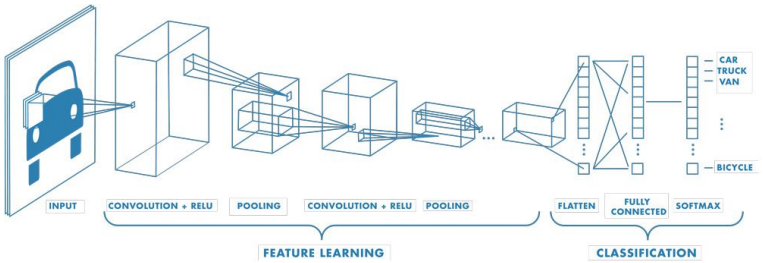

Figura 1.2: CNN (Fonte: [4])

grandezza per ogni singola immagine, permettendo di fornire in input l'intera immagine e non frammenti della stessa così da aumentare l'apprendimento spaziale della rete.

Questa maggior flessibilità comporta un'addestramento libero da limitaioni sull'input comportando una maggiore tolleranza agli errori e al rumore rendendo questa tipologia di reti particolarmente adatte a contesti poveri di dati.

1.3 U-Net

L'architettura **U-net** [7] è una particolare implementazione di FCN che permette di effettuare segmentazioni semantiche, in quanto è in grado di gestire input di qualsiasi dimensione e di produrre mappe di segmentazione più precise grazie alla sua capacità di apprendere contesti spaziali.

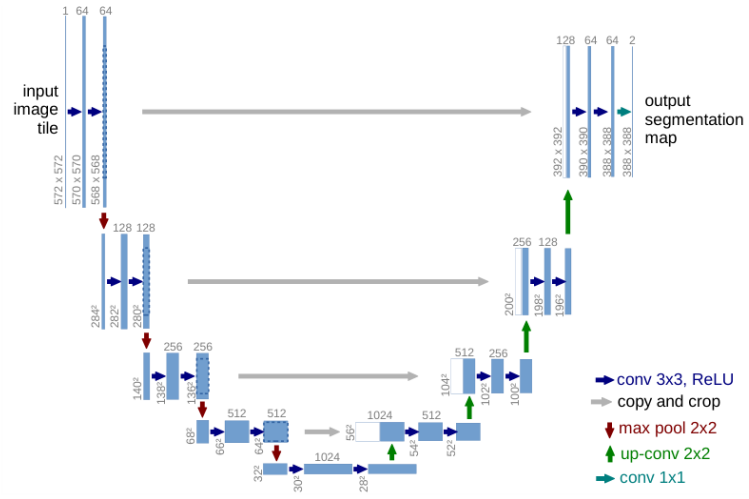

Figura 1.3: U-Net (Fonte: [7])

Capitolo 2

Lavori correlati

2.1 Segmentazione ossea

Un progetto degno di nota è **Towards whole-body CT Bone Segmentation** [5] poichè si propone di risolvere il problema della segmentazione ossea di umani e la raggiunge con ottimi risultati con accuratezza del $96\% \pm 2\%$ mediante la metrica di **Dice Score** e $94\% \pm 2\%$ con la metrica di **Intersection over Union**.

Il progetto si basa su una rete neurale convoluzionale che utilizza l'architettura U-Net [7] e raggiunge ottimi risulati con circa 4000 immmagini e 60 epoche di training lasciando l'architettura della rete invariata.

2.2 Segmentazione vasi sanguigni

L'articolo **Accurate Retinal Vessel Segmentation via Octave Convolution Neural Network** [3] propone un metodo di segmentazione automatico per la segmentazione dei vasi sanguigni retinici.

L'implementazione del metodo è basata su una rete neurale convoluzionale che utilizza l'architettura Octave UNet, modello che segue l'architettura di U-Net [7] ma utilizza l'operazione di convoluzione octave e delle convoluzioni octave trasposte.

Le convoluzini octave sono state introdotte in **Drop an Octave: Reducing Spatial Redundancy in Convolutional Neural Networks with Octave Convolution** [1] e sono una variante delle convoluzioni standard che prova a ovviare al problema di sbilanciamento delle *features* all'interno delle mappe di *features*.

Capitolo 3

Metodi

3.1 Dati

Tutti i dati provengono da analisi ecografiche effettuate presso **Azienda Ospedaliero-Universitaria di Parma**. Le immagini raccolte si rifanno al periodo compreso tra **Aprile 2022** e **Gennaio 2023** e sono state selezionando tenendo conto di alcuni parametri per standardizzare l'input come:

- Indice di massa corporea (BMI)
- Età delle madri
- Problematiche durante la gravidanza
- Problematiche dopo il parto
- Immagini scattate con la medesima angolazione rispetto al femore

Le immagini hanno subito un processo di *preprocessing* aggiuntivo per uniformare le dimensioni e la risoluzione. In particolare, sono state ridimensionate a immagini $1280px$ **di larghezza** e $876px$ **di altezza**.

Inoltre data che il problema è categorizzato come problema di segmantazione semantica binomiale, si è scelto di convertire le immagine da **RGB** a immagini in **scala di grigi** per ridurre la complessità del problema e per ridurre il quantitativo di dati necessari per l'addestramento della rete.

Sono state realizzate manualmente delle **maschere** di segmantazione per ogni immagine, in modo da avere un *ground truth* da confrontare con le predizioni del modello.

Data la scarsa quanti di dati a disposizione per l'addestramento della UNET, si è scelto di utilizzare alcune tecniche di *data augmentation* per aumentare la quantità di dati a disposizione. In particolare si è scelto di utilizzare le seguenti tecniche applicate in modo casuale per ogni coppia **immagine-maschera**

- *Flip* orizzontale e verticale
- *Rotazioni* di 35°
- *Rumore* Gaussiano

Queste tecniche di *data augmentation* migliorano notevolmente le segmentazioni ottenute mediante la rete UNET e rendono la rete più robusta a variazioni di luce e a rumore presente nelle immagini.

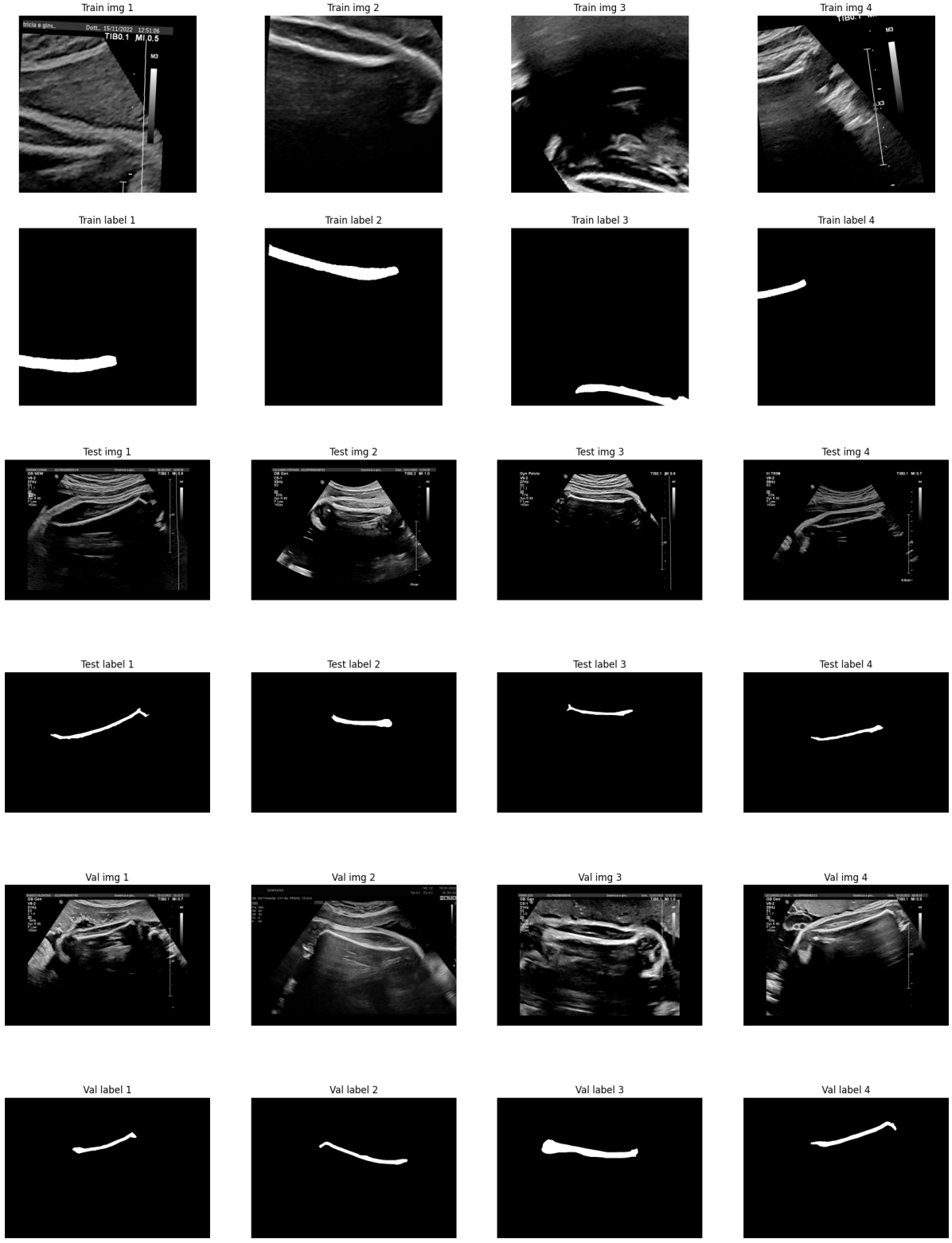

Figura 3.1: *Data augmentation*

L'ottimo risultato ottenuto è in buona parte dovuto alla *data augmentation* effettuata nella fase di addestramento del modello, la *data augmentation* ha migliorato l'adattabilità a contesti non controllati migliorando così la generalizzazione del modello.

Come si può notare dalla Figura 3.1 la *data augmentation* è stata effettuata in modo casuale per ogni coppia **immagine-maschera** ed è stata applicata solo nella fase di addestramento, lasciando cosi invariate le immagini inerenti al controllo del modello.

3.2 Etichettatura

Le immagini utilizzate per l'addestramento della rete sono state fornite da **Azienda Ospedaliero-Universitaria di Parma** e sono state etichettate manualmente mediante l'uso di un software di etichettatura chiamato **LabelMe** [9].

Il processo di etichettatura della immagini prevede per ogni immagine l'applicazione di indicatori che delimitano il perimetro del femore, potendolo così isolare dal resto dell'immagine.

3.3 Modello

Il modello dal quale si è partiti prende il nome di **U-Net**(Figura 1.3), la sua realizzazione iniziale è stata effettuata seguendo lo studio di Olaf Ronneberger, Philipp Fischer e Thomas Brox del 2015 [7].

Lo studio propone un'architettura di rete neurale convoluzionale per la segmentazione semantica di immagini biomediche, classificando ogni singolo pixel dell'immagine in una delle varie categorie del problema analizzato, la rete convoluzionale emersa da quesa analisi rimane tutto'oggi una delle più utilizzate in ambito medico per la segmentazione semantica data la sue performace e la sua versatilità.

L'implementazione iniziale ricalca il modello realizzato da Olaf Ronneberger, Philipp Fischer e Thomas Brox, utilizzando il framework **PyTorch** [8] come base per la realizzazione della rete.

L'architettura proposta da Olaf Ronneberger, Philipp Fischer e Thomas Brox è composta di 4 parti principali:

- **Encoder**: Visionabile graficamente come la parte discenente della U-Net
- **Bridge**: Visionabile graficamente come la linea di congiunzione fra la parte discendente e la parte ascendente della U-Net
- **Decoder**: Visionabile graficamente come la parte ascendente della U-Net
- **Output**: Visionabile graficamente come l'ultimo layer della U-Net

Le applicazioni che si appogiano a modelli derivati dall'architettura U-net dominano settori come la medicina e la biologia, in particolare la segmentazione di immagini biomediche, come la segmentazione di immagini ecografiche, la segmentazione di immagini TC e la segmentazione di immagini RM.

Le cause principali di tale successo possono essere ricondotte a:

- **Segmentazione dettagliata**: U-Net è in grado di produrre segmentazioni dettagliate e precise grazie alle sue innovative "skip connections" che permettono al modello di catturare sia i dettagli di basso livello che il contesto di alto livello.
- **Architettura compatta**: Nonostante la sua capacità di catturare dettagli, il modello è relativamente snello e può essere addestrato con successo anche con dataset di dimensioni moderate.
- **Adattabilità**: U-Net è stata originariamente concepita per applicazioni mediche, ma si è dimostrata estremamente versatile e può essere utilizzata con successo in una vasta gamma di contesti.

Ovviamente, come ogni modello, U-Net ha anche alcuni svantaggi. Il principale è la necessità di un dataset di addestramento ampio e accuratamente etichettato. Questo aspetto può essere un ostacolo, soprattutto in contesti in cui la disponibilità di dati è limitata. Inoltre, U-Net richiede una quantità significativa di memoria per memorizzare i pesi del modello, il che può diventare un problema quando si lavora con immagini ad alta risoluzione.

3.3.1 Convoluzione

La convoluzione è una delle operazini fondamentali utilizzate nelle reti neurali convoluzionali per estrarre le caratteristiche significative da un'input, la convoluzione coinvolge un filtro (o kernel) e l'input su cui si applica.

Il processo di convoluzione consiste nell'sommare ogni elemento di un'immagine al suo vicino, pesando ogni singola operazione mediante l'utilizzo del filtro(o kernel) il calcolo della feature map di uscita è calcolata come segue:

$$\left(\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} * \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \right)[2,2] = \tag{3.1}$$

$$= (i \cdot 1) + (h \cdot 2) + (g \cdot 3) + (f \cdot 4) + (e \cdot 5) + (d \cdot 6) + c \cdot 7) + (b \cdot 8) + (a \cdot 9) \tag{3.2}$$

3.3.2 Max pooling

Il *max pooling* è un'operazione chiave all'interno della rete U-Net e delle reti neurali convoluzionali (CNN) in generale.

Il *max pooling* è utilizzato per ridurre la dimensione delle feature map, consentendo di ridurre la complessità del problema da approssimare, comportando una maggior resistenza all'*overfitting*, migliorando la capacitá di generalizzazione del modello e di ottenere una rappresentazione più invariante rispetto alle piccole variazioni spaziali nell'input.

3.3.3 Encoder

TODO: aggiungere immagini singole per i vari pezzi dell'encoder

La fase di *encoding* è la prima fase della rete U-Net, composta da una serie di strati di convoluzione(3.3.1) e max pooling(3.3.2) che riducono progressivamente la dimensione spaziale dell'immagine mentre aumentano il numero di canali di *features*.

Nello specifico, la fase di *encoding* è composta da 3 parti principali:

- **Strato iniziale**: Questo strato applica diverse operazioni di convoluzione ai dati di input per estrarre le caratteristiche di basso livello, come bordi e texture. Queste operazioni iniziali consentono al modello di comprendere dettagli fondamentali dell'immagine.

- **Downsampling**: Dopo lo strato iniziale, la fase di encoding utilizza operazioni di max pooling o convoluzione con un passo (stride) superiore a 1 per ridurre la dimensione delle feature map. Questo processo di downsampling riduce la risoluzione spaziale, ma aumenta il numero di canali delle feature, catturando informazioni di livello superiore. Ogni strato di downsampling estrae caratteristiche sempre più astratte e globali dall'immagine.

- **Strati intermedi**: Nel cuore della fase di encoding si trovano gli strati intermedi. Questi strati applicano operazioni di convoluzione multiple con l'obiettivo di catturare

caratteristiche di complessità crescente. A ogni strato intermedio, le feature map si allargano, consentendo al modello di comprendere dettagli più ampi e contestuali. Questi strati intermedi sono cruciali per l'acquisizione di informazioni di alto livello.

3.3.4 Bridge

Il *bridge* è un'innovazione dell'architettura U-Net che contribuisce in modo significativo alla precisione della segmentazione delle immagini.

La fase di *bridge* permette di trasferire informazioni rilevanti tra l'encoder e il decoder attraverso skip connections, che consentono il trasferimento di informazioni rilevanti. Questo approccio multi-scala è fondamentale per ottenere una segmentazione precisa delle immagini, poiché consente al modello di considerare dettagli sia di basso che di alto livello durante il processo di segmentazione.

3.3.5 Decoder

Nella rete U-Net, la fase di decoding é responsabile della ricostruzione dell'immagine segmentata a partire dalle informazioni estratte durante l'encoding. Questa fase é fondamentale per ottenere una segmentazione di alta qualitá.

Le fasi principali del *decoder* sono:

- **Upsampling**: La fase di decoding inizia con l'operazione di upsampling, che serve a ripristinare gradualmente la dimensione delle feature map ai livelli originali dell'immagine. Ciò viene fatto utilizzando operazioni come la trasposta della convoluzione (deconvoluzione) o l'interpolazione bilineare. L'obiettivo è ottenere feature map di dimensioni compatibili con quelle dell'immagine di input.

- **Skip Connections**: Un aspetto distintivo della U-Net sono le skip connections, o connessioni di salto. Queste connessioni collegano le feature map estratte durante l'encoding alle corrispondenti feature map nella fase di decoding. Ciò consente di combinare informazioni multi-scala, in modo che il modello possa accedere sia a dettagli fini che a contesto di alto livello. Le skip connections sono fondamentali per migliorare la precisione della segmentazione.

- **Convoluzione nel Decoding**: Dopo l'upsampling e l'integrazione delle skip connections, vengono applicate operazioni di convoluzione per raffinare ulteriormente le feature map. Queste convoluzioni possono avere lo scopo di "mescolare" le informazioni o di catturare dettagli specifici a livelli più alti.

3.3.6 Output

La parte finale della rete U-Net é composta da uno o più strati di convoluzione che riducono la profondità delle feature map alla dimensione desiderata per l'output finale. Questi strati producono l'immagine segmentata in cui ogni pixel è etichettato con la classe di appartenenza (esempio: sfondo, oggetto, ecc.).

3.4 Rimozione sliding window

Grazie agli avanzamenti tecnologici portati avanti negli anni da costruttori hardware e software, si è riusciti a ridurre notevolmente i tempi di elaborazione delle immagini e la grandezza massima delle immagini che possono essere elaborate.

Si è quindi scelto di rinunciare all'approccio sliding window che risulta più conservativo in termini di memoria e di tempo di elaborazione, per un approccio più moderno che sfrutta la potenza di calcolo delle GPU e la loro memoria dedicata notevolmente più grande rispetto alle GPU del passato.

Un *effetto collaterale* riscontrato con l'utilizzo di immagini intere è quello della miglior comprensione spaziale delle immagini da parte della rete, questo effetto è dovuto al fatto che la rete ha a disposizione l'intera immagine e non solo una parte di essa, questo permette alla rete di comprendere meglio il contesto spaziale dell'immagine e di migliorare la segmentazione.

3.5 Modifica encoding e decoding

Mediante svariate prove si è notato che aumentando il numero di iterazioni di convoluzione nei vari strati di encoding e decoding, si ottengono risultati migliori in termini di segmentazione a scapito di un aumento del tempo di elaborazione e del consumo di memoria.

Si è quindi scelto di modificare le fasi di *decoding* e di *encoding*:

Layer	In channels	Out channels
Encoder	1	64
Encoder	64	128
Encoder	128	256
Encoder	256	512

Layer	In channels	Out channels
Decoder	1024	512
Decoder	512	256
Decoder	256	128
Decoder	128	64

Tabella 3.1: Encoding e Decoding originale

Layer	In channels	Out channels
Encoder	1	16
Encoder	16	32
Encoder	32	64
Encoder	64	128
Encoder	128	256
Encoder	256	512

Layer	In channels	Out channels
Decoder	1024	512
Decoder	512	256
Decoder	256	128
Decoder	128	64
Decoder	64	32
Decoder	32	16

Tabella 3.2: Encoding e Decoding modificati

L'idea alla base di questa modifica risiede nel significato intrinseco dell'operazione di convoluzione e di max pooling, aggiungendo iterazioni di convoluzioni, si permette alla rete in primo luogo di estrarre più feature importanti alla classificazione dei pixel e successivamente mediante il max pooling si eliminano le feature meno importanti e si riduce la dimensione delle feature map.

3.6 Metriche

Considerando la tipologia di *task* si è pensato di usare la metrica *Dice BCE Loss* per la *loss* e *Intersection over Union* per l'*accuratezza*.

3.6.1 Dice BCE Loss

La *Dice BCE Loss* è una metrica che combina due metriche, la *Dice Loss* e la *BCE Loss*.

$$L = L_{\text{Dice}} + L_{\text{BCE}} \tag{3.3}$$

$$L_{\text{Dice}} = 1 - \frac{2\sum_i^N p_i g_i + \varepsilon}{\sum_i^N p_i^2 + \sum_i^N g_i^2 + \varepsilon} \tag{3.4}$$

$$L_{\text{BCE}} = -\frac{1}{N}\left[g_i \sum_{i=1}^{N} p_i + (1 - g_i) \sum_{i=1}^{N} (1 - p_i)\right] \tag{3.5}$$

Quindi ne risulta che la *loss* sarà calcolata mediante:

$$L = 1 - \frac{2\sum_i^N p_i g_i + \varepsilon}{\sum_i^N p_i^2 + \sum_i^N g_i^2 + \varepsilon} - \frac{1}{N}\left[g_i \sum_{i=1}^{N} p_i + (1 - g_i) \sum_{i=1}^{N} (1 - p_i)\right] \tag{3.6}$$

Dove:

- p_i è il valore di *ground truth*
- g_i è il valore *predetto* dal modello

3.6.2 Intersection over Union(IoU)

Per la metrica dell'accuratezza della segmentazione, è stata utilizzata la metrica *Intersection over Union* (IoU), poichè è una metrica che permette di valutare la capacità di segmentazione del modello facendo il rapporto tra l'area di intersezione tra la maschera predetta e quella di *ground truth* e l'area di unione tra le due maschere, formalmente:

$$\text{IoU} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}} \tag{3.7}$$

Dove **TP** è il numero di *True Positive*, **FP** è il numero di *False Positive* e **FN** è il numero di *False Negative*.

3.7 Validazione del modello

La *cross-validation* (validazione incrociata) è una tecnica fondamentale nell'ambito del machine learning e dell'addestramento di modelli predittivi. Essenzialmente, la cross-validation è un metodo per valutare le prestazioni di un modello in modo robusto, valutandolo su più insiemi di dati per ottenere stime più affidabili delle sue capacità predittive. Questo processo aiuta a mitigare il rischio di overfitting (sovradattamento) e offre una migliore comprensione delle prestazioni del modello.

Fornisce stime più affidabili delle prestazioni del modello, riducendo il rischio di ottenere stime di prestazioni spurie a causa di una singola divisione dei dati.

Figura 3.2: Cross-validation

Capitolo 4

Risultati sperimentali

4.1 Addestramento

Il modello è stato addestrato mediante l'uso della *cross-validation*(Figura 3.2) con una suddivisione dei dati in 5 parti uguali(5 folds).

Essendo una tipologia di apprendimento supervisionata, al modello sono fornite immagini originali e le loro segmentazioni effettuate manualemte.

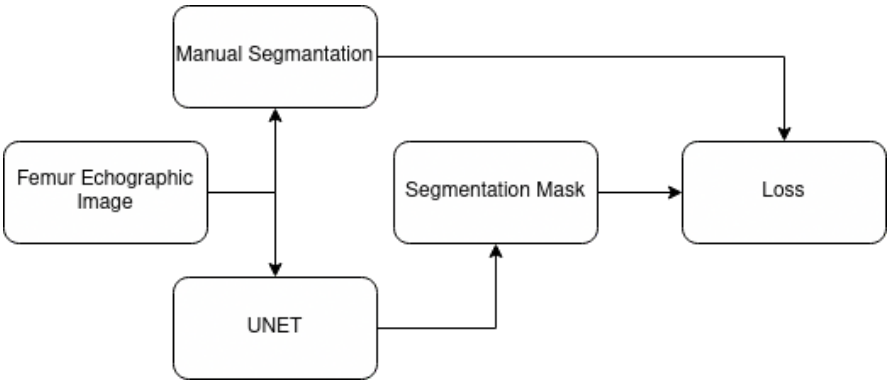

Figura 4.1: Addestramento del modello

Effettuando l'addestramento con 5 folds, il modello viene addestrato 5 volte, ogni volta con un fold diverso, l'errore finale è dato dalla media degli errori ottenuti dalle 5 iterazioni.

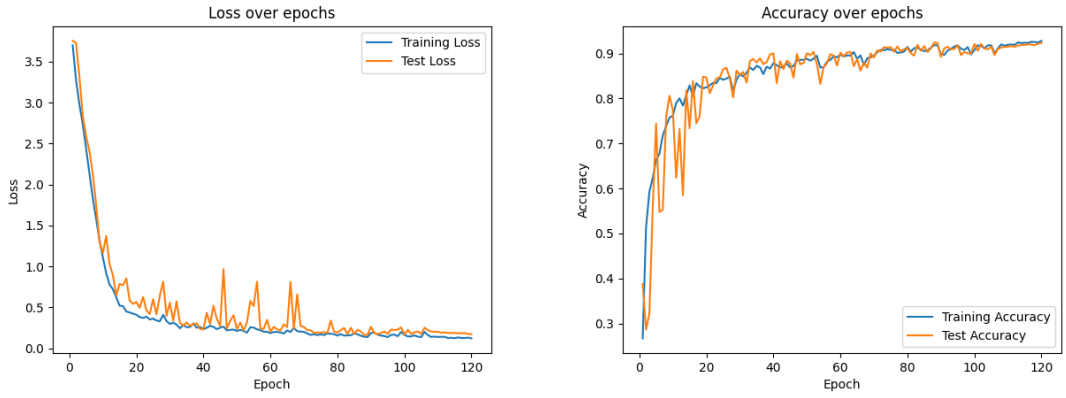

Figura 4.2: Errore e accuratezza della prima porzione di dati

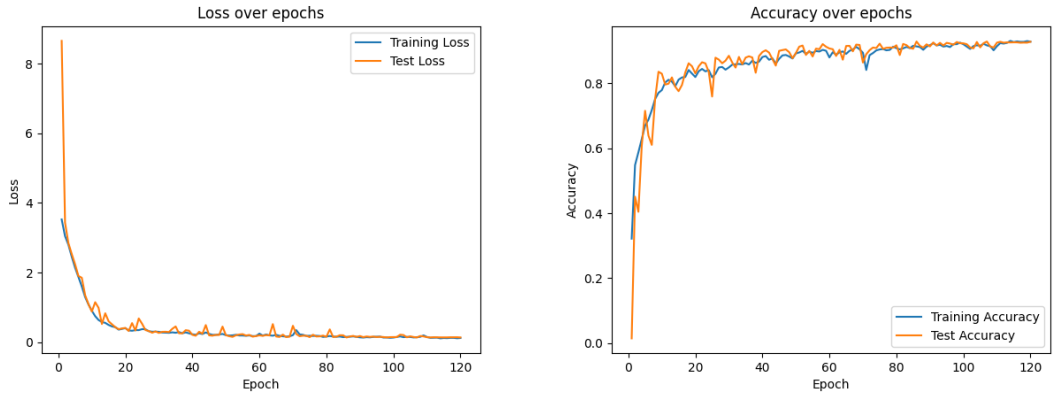

Figura 4.3: Errore e accuratezza della seconda porzione di dati

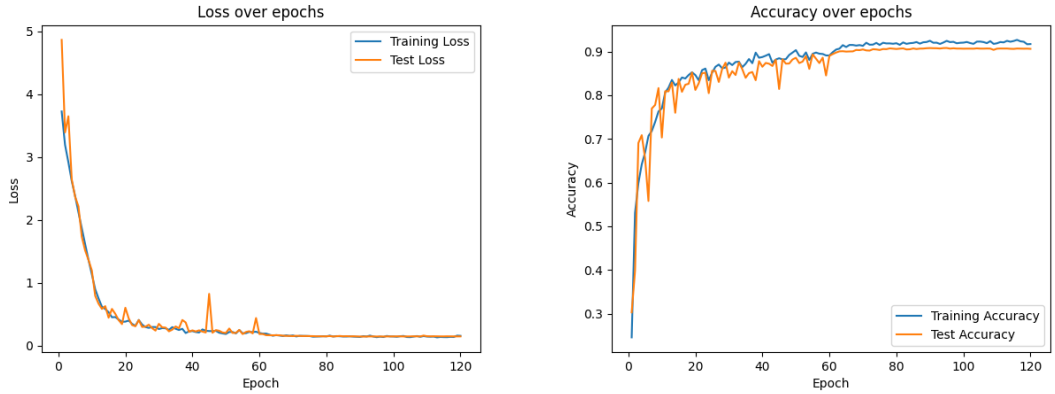

Figura 4.4: Errore e accuratezza della terza porzione di dati

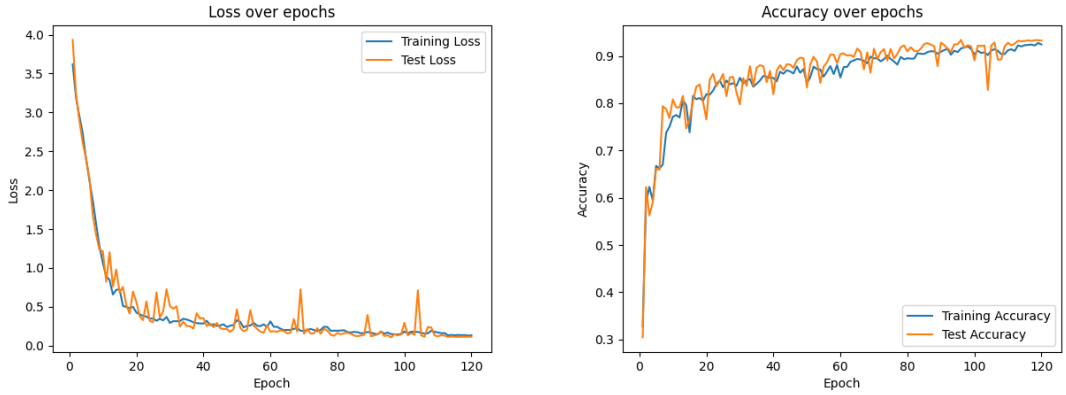

Figura 4.5: Errore e accuratezza della quarta porzione di dati

L'**errore** medio del modello è di 8% mentre l'**accuratezza** media è di 92%(le metriche utilizzate sezione 3.6).

Considerando che questo modello è stato utilizzato in ambito medico per velocizzare e standardizzare la segmentazione dei femori per un'analisi su questi ultimi, oltre ad analisi quantitative, è stato necessario effettuare delle analisi qualitative sulal segmentazione ottenuta dal modello.

Nelle immagini seguenti viene riportato uno delle immagini prese in considerazione per l'addestramento del modello e vengono mostrate le segmentazione manuali, le segmentazioni

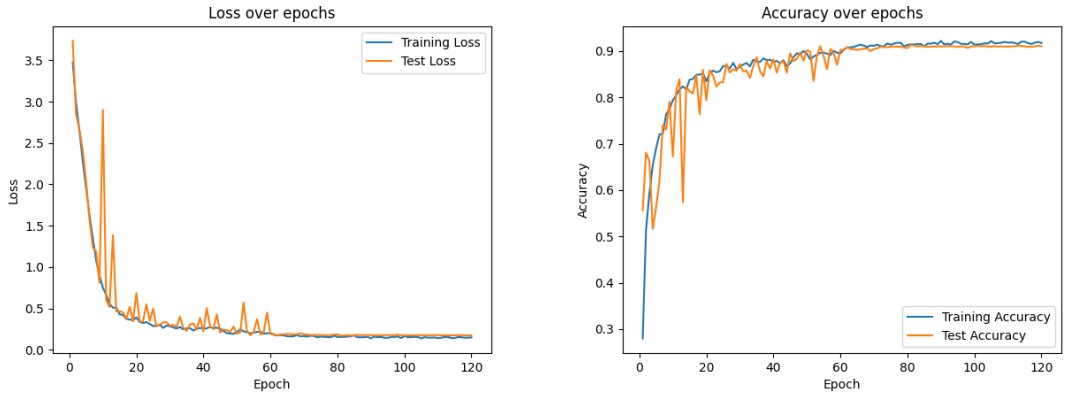

Figura 4.6: Errore e accuratezza della quinta porzione di dati

ottenute dal modello e la differenze nella classificazione dei pixel tra le due segmentazioni.

Partendo da una immagini (Figura 4.7) ottenuta mediante la raccolta dati effettuata dai medici,

Figura 4.7: Immagine originale

I risultati ottenuti mediante la segmentazione manuale e la segmentazione del modello sono i seguenti:

Per sostenere la tesi che il modello riesca a segmentare correttamente le immagini, è stata calcolata la distribuzione dei pixel per controntare la segmentazione manuale con quella del modello.

Un'altra rappresentazione a confronto dei risultati ottenuti è la seguente:

Dai risultati qualitatitivi e quantitativi si può constatare che il modello ha una performance molto promettente in quanto supera abbondantemente un'accuratezza del 90% e può essere addestrato incrementando il numero di imamgini a disposizione.

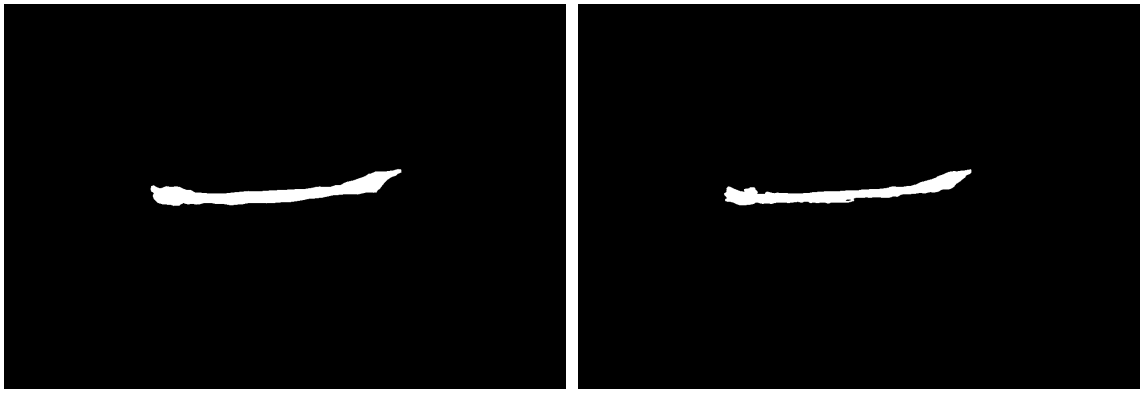

Figura 4.8: Segmentazione manuale(sinistra) e segmentazione del modello(destra)

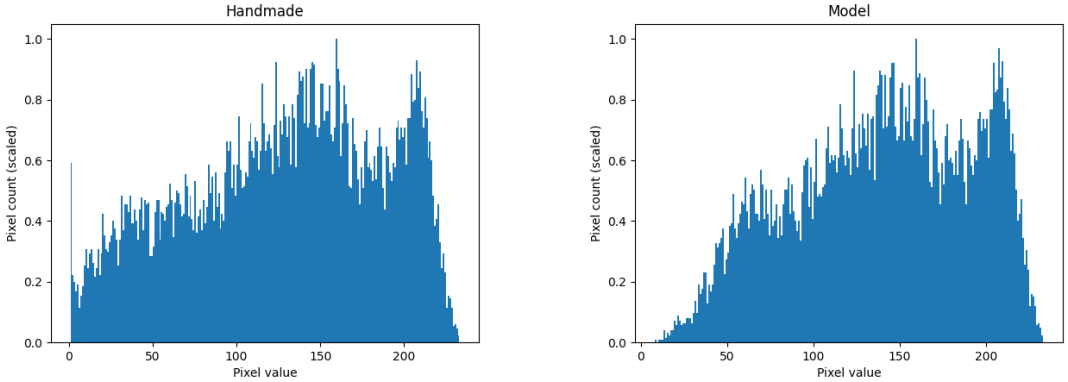

Figura 4.9: Distribuzione dei pixel della segmentazione manuale(sinistra) e del modello(destra)

Non sono necessari ulteriori segmentazioni manuali ma si possono direttamente sfruttare le nuove immagini raccolte, segmentarle mediante l'uso del modello e utilizzarle per l'addestramento.

4.2 Problema

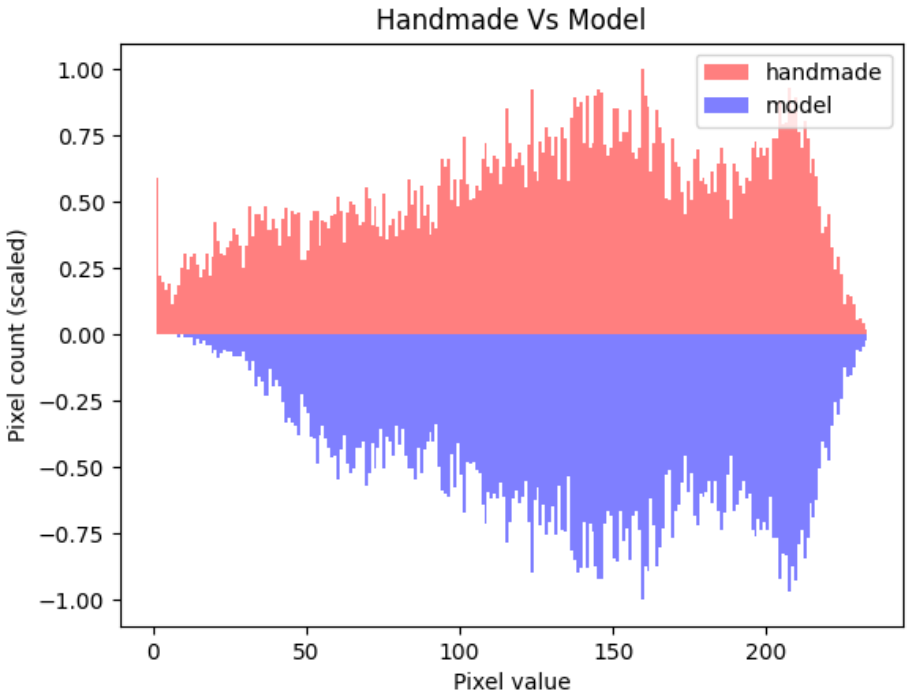

Figura 4.10: Confronto tra la segmentazione manuale e quella del modello

Bibliografia

[1] Yunpeng Chen, Haoqi Fan, Bing Xu, Zhicheng Yan, Yannis Kalantidis, Marcus Rohrbach, Shuicheng Yan, and Jiashi Feng. Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution, 2019.

[2] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[3] Zhun Fan, Jiajie Mo, Benzhang Qiu, Wenji Li, Guijie Zhu, Chong Li, Jianye Hu, Yibiao Rong, and Xinjian Chen. Accurate retinal vessel segmentation via octave convolution neural network, 2020.

[4] The MathWorks Inc. What is a convolutional neural network? https://www.mathworks.com/discovery/convolutional-neural-network-matlab.html.

[5] André Klein, Jan Warszawski, Jens Hillengaß, and Klaus Hermann Maier-Hein. Towards whole-body CT bone segmentation. In *Bildverarbeitung für die Medizin 2018*, pages 204–209. Springer Berlin Heidelberg, 2018.

[6] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation, 2015.

[7] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[8] PyTorch Team. Pytorch.

[9] Kentaro Wada. labelme, Sep 25, 2011 – Oct 25, 2023.