Politecnico di Torino

Elaborazione di immagini mediche

Master's Degree in Biomedical Engineering

A.A. 2021-2022

Contest U-Net

Rigazio Sofia (282247), Romano Anastasio (289707),
Roccaro Lucia (289423), Ruzzante Elena (292194)

1 Introduzione

La segmentazione della ghiandola prostatica ha molte applicazioni nella diagnosi e trattamento del carcinoma della prostata. L'esame di risonanza magnetica (MRI) ricopre un ruolo sempre più importante nell'individuazione del cancro della prostata in pazienti con sospetto clinico: questo tipo di esame permette infatti di stabilire l'estensione e conseguentemente la gravità del tumore.

La segmentazione manuale può tuttavia richiedere molto tempo, oltre ad essere intrinsecamente soggetta a variabilità sia inter- che intra-operatore.

Lo scopo del Contest è proprio quello di automatizzare questa analisi attraverso l'allenamento di una Deep Neural Network in grado di individuare la ghiandola prostatica in immagini acquisite tramite risonanza magnetica pesata T_2. Per la segmentazione è stato scelto il modello di rete convoluzionale U-net.

1.1 Descrizione del Dataset

Il dataset a disposizione è costituito in totale da 50 volumi acquisiti mediante MRI e suddivisi in Training set (32 volumi), Validation set (8 volumi) e Test set (10 volumi). Ogni volume è costituito da 24 slice di dimensione 512×512 pixel in scala di grigi (1 canale colore) e formato `uint8`.
Fanno parte del dataset anche le maschere binarie delle lesioni dei volumi di Training e Validation set, ottenute mediante segmentazione manuale da parte di un operatore esperto. Anch'esse sono presentano un singolo canale colore e sono in formato `uint8`.

La segmentazione è stata modellizzata ed eseguita su ogni singola slice dei volumi a disposizione. Le immagini sono state pre-processate e le maschere ottenute in output dalla rete sono state post-processate al fine di migliorare la qualità del risultato.

2 Implementazione

2.1 Pre-processing

Prima di procedere con l'allenamento della rete, abbiamo effettuato una serie di operazioni di pre-processing sulle immagini: tutte le operazioni qui descritte sono contenute nella funzione `preprocess`.

Abbiamo osservato che un gruppo di immagini (probabilmente proveniente da un particolare macchinario) presenta una cornice esterna nera: abbiamo pertanto deciso di rimuoverla, eliminando righe e colonne i cui valori dei pixel sono tutti a 0. Per non perdere la dualità immagine-maschera, le medesime righe e colonne vengono rimosse anche dalle corrispondenti maschere. In seguito, viene eseguito un `resize` che riporta le immagini e le maschere alle dimensioni originali 512×512.

In fase di training abbiamo anche tentato di rimuovere il rumore testando l'applicazione di un filtro di *averaging* con *kernel* di dimensione 3×3 (il più piccolo possibile per non introdurre eccessivo *smooth*) e un filtro mediano. Nelle prove, questi filtri sono stati applicati sia singolarmente, sia in cascata. Questi tentativi hanno smussato eccessivamente le immagini, portando a risultati non soddisfacenti.
Abbiamo dunque optato nello script finale per l'implementazione di un lieve denoising mediante filtraggio passabasso con Kernel di Gauss (`size` $= 3$, $\sigma = 3$). La soluzione porta a valori di DSC comparabili alle prove senza filtro, ma la sovrapposizione delle maschere automatiche alle immagini di Test risulta, a nostro avviso, più accurata.

In seguito abbiamo tentato un aumento del contrasto per valori compresi tra il $+5\,\%$ e il $+20\,\%$, con l'obiettivo di enfatizzare i bordi della prostata. I risultati ottenuti sono però risultati peggiorativi di circa il $10\,\%$ nei DSC: abbiamo ritenuto quindi inopportuna la modifica del contrasto delle immagini.

Per poter utilizzare le maschere per l'allenamento della rete, le abbiamo convertite in formato `float32`. Dopo la conversione, per ogni maschera abbiamo identificato le due classi "sfondo" e "prostata" tramite il comando `to_categorical` proveniente dalla libreria `keras`. Il procedimento è stato ripetuto in modo analogo per tutte le immagini e le maschere appartenenti al dataset.

2.2 Data augmentation

Per ampliare il Training set abbiamo testato l'utilizzo di operazioni di data augmentation. Inizialmente abbiamo allenato le reti senza effettuarlo, in modo da avere dei valori di riferimento.
In seguito, abbiamo introdotto uno shift verticale e orizzontale pari al $+20\,\%$ e `fill_mode='reflect'`.
I risultati ottenuti in questa prova, mostrati in Tabella 1, presentano una percentuale di DSC nel

Training set più elevata (circa +8 %) rispetto al DSC nel Validation set. Abbiamo interpretato questo risultato come un possibile overfitting sui dati di Training, e quindi da scartare.

Abbiamo quindi deciso di introdurre, oltre ai parametri della prova precedente, anche un angolo di rotazione di 180° e un ribaltamento orizzontale e verticale. Questo terzo setting ha portato a risultati più bilanciati seppure leggermente inferiori alla prova senza data augmentation.
In definitiva abbiamo deciso di mantenere quest'ultimo setting di parametri per avere un dataset di immagini più cospicuo ed una rete più robusta.

2.3 Architettura e parametri della rete

La rete utilizzata è una U-Net, implementata utilizzando la libreria `keras`. L'architettura tipica di una U-Net è costituita da:

- Un *encoder* che riduce l'immagine in ingresso in una *feature map* (*subsampling*), attraverso particolari *pooling layer*, estraendone gli elementi chiave
- Un *decoder* che amplifica la *feature map* in un'immagine, usando i livelli di deconvoluzione (*upsampling*), impiegando i *pooling layer* appresi per permettere la localizzazione degli elementi.

Tra le architetture di rete disponibili è stata scelta la `resnet34`, costituita da 34 layer convoluzionali. Abbiamo anche provato ad allenare la rete con un il modello `resnet50` ma le prestazioni ottenute non risultano migliorative rispetto ai precedenti risultati.

Le condizioni di stop, definite in `EarlyStopping`, sono state impostate come segue:

- Abbiamo scelto l'accuratezza (`monitor = 'val_binary_accuracy'`) come parametro da monitorare. Le prove effettuate monitorando la `loss` hanno portato a valori di DSC inferiori di circa il $-2\,\%$ sul Training set. Di conseguenza, avendo scelto come parametro l'accuratezza, il target sul parametro è la massimizzazione (`mode = 'max'`).
- Abbiamo impostato il numero di epoche senza miglioramento dopo le quali l'allenamento verrà interrotto (`patience`) su un valore pari a 4. Questo parametro viene scelto cercando di minimizzare il costo computazionale, mantenendo le migliori prestazioni possibili e cercando di non incorrere in overfitting. Dopo diversi tentativi abbiamo osservato che:
  - valori di `patience` pari a 2 non hanno permesso alla rete di effettuare un numero sufficiente di epoche di allenamento (circa 3 epoche) con DSC molto bassi (rete `A` in tabella Tabella 1);
  - valori di `patience` da 6 a 15 aumentano invece il tempo di allenamento della rete senza miglioramenti, anzi portando a risultati visivamente non soddisfacenti sul Test set (come visibile ad esempio in Figura 1).
- variazione minima della quantità monitorata per qualificarsi come miglioramento (`min_delta`) è stata invece settata a $0,01$.

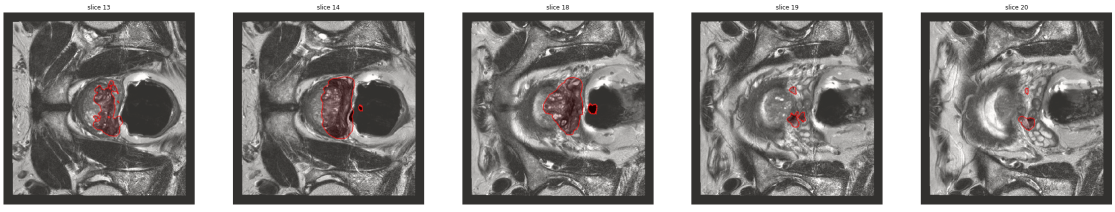


Figura 1: Esempio del risultato di una cattiva segmentazione su alcune slice del Test set (immagine "3258.tiff", rete allenata con `patience`= 15)

Dato l'esiguo numero di elementi nel dataset per allenare una Deep Neural Network, il parametro `encoder_weights` è stato settato a `'imagenet'`. Questo ci ha permesso inizializzare la rete con pesi pre-allenati e non randomici. Per poter utilizzare questi pesi, pre-allenati per immagini RGB, abbiamo forzato le immagini da grayscale a RGB impilando 3 layer contenenti gli stessi valori.

La funzione di attivazione scelta per la rete finale è la sigmoide, ovvero quella tipicamente utilizzata per effettuare classificazione binaria.

Il parametro `batch size` è stato scelto pari a 8, che è risultato essere il miglior compromesso tra costo computazionale e accuratezza dei risultati. Infatti, con valori di `batch` maggiori la rete si allena

più velocemente e con minor costo computazionale, seppur la stima sia meno accurata rispetto al caso di `batch` minore.

La funzione di loss utilizzata è la `binary_crossentropy` e quella di accuratezza la `binary_accuracy`. L'algoritmo di ottimizzazione utilizzato è `Adam`, un metodo di *gradient descent* basato su una stima adattiva dei momenti del primo e secondo ordine.
Le epoche richieste per l'allenamento della rete in tutte le prove effettuate con il parametro `patience` pari a 4 sono risultate essere sempre inferiori a 10 per via delle condizioni di stop imposte. Per mantenere un po' di margine abbiamo comunque imposto un numero massimo di epoche pari a 20, aumentandolo nelle prove con valori di `patience` maggiori.

2.4 Post-processing

In primo luogo ci siamo posti nella condizione di poter confrontare coerentemente le maschere ottenute in output dalla rete con quelle ottenute manualmente dalle immagini originali (in cui è presente l'eventuale bordo nero da noi rimosso nella fase di pre-processing). Per fare ciò, abbiamo eseguito il procedimento inverso a quello di rimozione delle cornici, controllando la presenza del bordo nell'immagine originale, effettuando l'eventuale resize della maschera alla giusta dimensione (inferiore a 512×512) e aggiungendo il bordo alla posizione originaria (in modo da ottenere in output nuovamente una maschera di dimensione 512×512).

Al termine, abbiamo rimosso gli eventuali piccoli buchi e ed oggetti dalla maschera utilizzando le funzioni `remove_small_holes` e `remove_small_objects` della libreria `skimage.morphology`, mantenendo i parametri di dimensione di default (area di 64 px). Abbiamo provato anche l'utilizzo di parametri con dimensioni diverse, ma i settaggi di default sono risultati essere i migliori.

I passaggi sopra citati sono implementati all'interno della funzione `postprocess`.

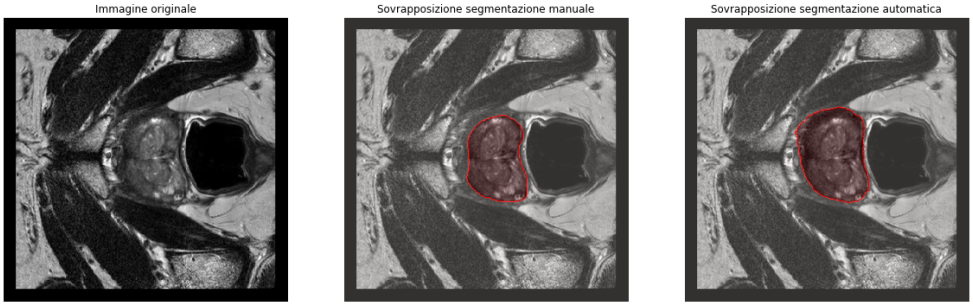
3 Risultati

La seguente Tabella 1 riporta i risultati delle prove più significative da noi effettuate. In giallo è evidenziata la configurazione della rete finale scelta, inserita nel materiale consegnato.
In Figura 2 è invece rappresentato un esempio di segmentazione per una slice di una immagine rispettivamente del Training e del Validation set.

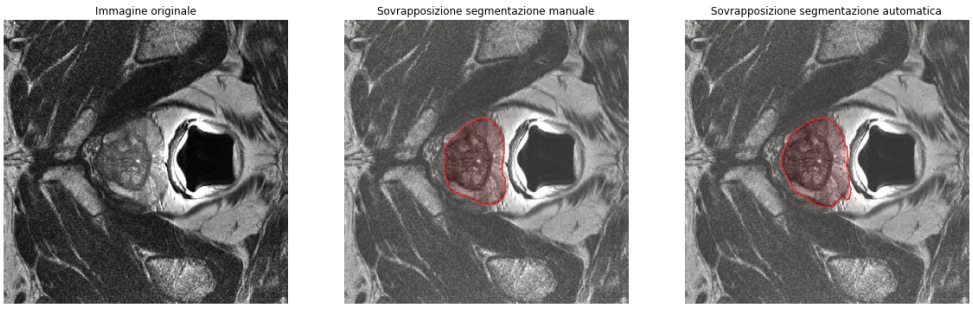
Label rete		A	C	E	H	L	N	M	O	S	T	U
Pre-processing	Filtri	-	-	-	-	-	-	-	(A)	(B)	(C)	(A), (B)
	Rimozione cornici	no	no	no	no	no	si	si	si	si	si	si
	Data augmentation	rotazione, flip, shift	solo shift	rotazione, flip, shift	rotazione, flip, shift	-	rotazione, flip, shift	solo shift	rotazione, flip, shift	rotazione, flip, shift	rotazione, flip, shift	rotazione, flip, shift
Model settings	Backbone	resnet34	resnet34	resnet50	resnet34	resnet34	resnet34	resnet34	resnet34	resnet34	restnet34	resnet34
	Patience	2	4	4	4	4	4	4	4	4	4	4
	Batch size	8	8	8	4	8	8	8	8	8	8	8
Post-processing (small holes, small obj.)		no	si	si	si	no	si	no	si	si	si	si
Risultati	DSC (%) - Training	0,460	0,920	0,827	0,881	0,890	0,894	0,920	0,884	0,838	0,747	0,791
	DSC (%) - Validation	0,430	0,850	0,810	0,866	0,840	0,889	0,845	0,868	0,799	0,760	0,801
	Commenti	-	Probabile overfitting	-	-	-	-	Probabile overfitting -	-			Probabile overfitting

FILTER LEGEND: (A) LPF con Kernel Gaussiano (size = 3, $\sigma = 3$), (B) Aumento contrasto (+20 %), (C) Aumento contrasto (+5 %)

Tabella 1: Riepilogo riassuntivo delle varie soluzioni implementate



(a) Training set - immagine MRI: MIP-PROSTATE-01-0024.tiff, slice 10



(b) Validation set - immagine MRI: 25.tiff, slice 10

Figura 2: rappresentazione risultati rete - Training set e Validation set

Qui di seguito sono riportati i valori di Dice Similarity Coefficient calcolati sulle maschere automatiche ottenute con la rete selezionata:

`DSC Training = 0,8839 - DSC Validation = 0,8681`

I valori riportati sono rispettivamente una media dei DSC calcolati su tutti i volumi appartenenti al Training e al Validation set. Riportiamo per completezza anche un grafico contenente tutti i valori di DSC calcolati per i singoli volumi (Figura 3). Dal grafico si può osservare che i risultati per ogni volume sono molto simili tra loro.

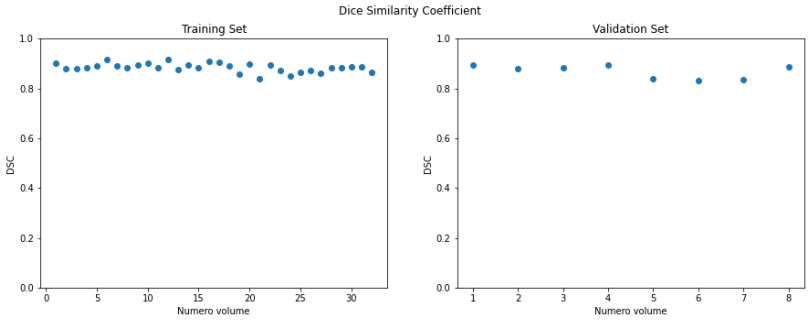


Figura 3: Grafico riassuntivo DSC - Training e Validation Set

4 Conclusioni

In conclusione, considerando il dataset a disposizione, riteniamo i risultati soddisfacenti e coerenti con le nostre aspettative. I valori di DSC ottenuti per Training e Validation set risultano essere bilanciati tra loro. I risultati della rete nel Test set sono stati visualizzati mediante sovrapposizione di maschere automatiche alle immagini e sembrano non mostrare overfitting della rete.
Tuttavia i risultati ottenuti possono senz'altro essere migliorati incrementando il numero di immagini per l'allenamento della rete. Le performance delle reti di Deep Learning, infatti, migliorano all'aumentare del numero di dati in input.