Università di Perugia
Dipartimento di Matematica e Informatica

Applied Image and Signal Processing

Relazione
Laboratorio Patologie Vascolari

Professore
Prof. Gianluca Vinti
Dott. Arianna Travaglini
Dott. Davide Belfiore

Studenti
Chiara Darielli
Giovanna Donato
Martina Longaroni
Lucrezia Rinelli
Tommaso Romani
Nicolò Vescera

Anno Accademico 2022-2023

Indice

1 Introduzione

Quando si parla di **patologie vascolari** ci si riferisce a problematiche relative a vene e arterie. Nella maggior parte dei casi queste provocano stenosi o aneurismi, fino anche ad occlusioni parziali o totali. Per una diagnosi accurata ci si affida a immagini ottenute tramite la **tomografia computerizzata**. Tale metodologia fornisce per ciascun paziente una serie di immagini a sezione assiale. Affinché il medico possa decidere se operare o meno, viene effettuata la TAC una seconda volta, immettendo nel paziente un mezzo di contrasto. Quest'ultimo serve per rendere radiopaco il sangue e quindi avere una visione migliore. Il problema che sorge a questo punto è che questo non può essere somministrato a pazienti che presentano problematiche renali, cardiache o di altro genere.

Il lavoro svolto in questo laboratorio vuole essere un contributo utile allo studio di un metodo alternativo all'iniezione del mezzo di contrasto in questa tipologia di pazienti.

Cenni sulle patologie vascolari

Tra le patologie vascolari troviamo l'**aneurisma aortico**, una condizione patologica che consiste in una dilatazione permanente e localizzata di un tratto dell'arteria aorta. Si forma quando le fibre dello strato intermedio del vaso si logorano, degenerano o si assottigliano e i meccanismi riparativi le rimpiazzano con tessuto fibroso che essendo poco elastico, tende a cedere sotto l'impatto delle sistoli.
Un aneurisma dell'aorta può avere l'aspetto di un palloncino (**aneurisma aortico sacciforme**) oppure a forma di fuso (**aneurisma aortico fusiforme**). In base al tratto interessato si divide poi in aneurisma dell'aorta addominale (AAA), aneurisma dell'aorta toracica (AAT) oppure aneurisma toraco-addominale (AAAT). Se un aneurisma non viene diagnosticato e trattato, si può rompere e il paziente può andare incontro al decesso a causa di un'emorragia interna imponente.

2 Obiettivo

L'obiettivo di questo laboratorio è addestrare due reti neurali in modo che siano in grado di estrarre:

1. l'immagine con il mezzo di contrasto partendo dall'immagine in basale ($BASALE \rightarrow MDC$);

2. il lume pervio del vaso aortico partendo dall'immagine in basale ($BASALE \rightarrow LP$).

Attraverso l'analisi dei risultati delle metriche adottate sulle reti neurali, forniremo in dettaglio il grado di affidabilità delle tecniche utilizzate.

3 Database e procedure

Sono state analizzate ed elaborate le immagini digitali del tratto aortico di sei pazienti. Per ciascuno di essi vanno effettuate alcune modifiche a tutte le immagini TC prima di poterle utilizzare per le reti neurali. A tal proposito si utilizzano due software: **ImageJ** e **GIMP**.

Per prima cosa è necessario assicurarsi che le immagini siano registrate. È stato quindi elaborato un algoritmo in grado di effettuare questa procedura per ogni paziente.
A questo punto si importano le sequenze di immagini in basale e con mezzo di contrasto su ImageJ per ricavare la ROI, cioè Region Of Interest, nel nostro caso la regione quadrata contenente l'aorta. È fondamentale utilizzare le stesse coordinate per entrambe le tipologie di immagini per il corretto funzionamento della rete. Dopo aver effettuato il crop agli stack e salvato le sequenze in formato `.png` si passa a GIMP.

La seconda fase consiste nella generazione delle maschere binarie. A partire dalle immagini MDC tagliate si aggiunge un nuovo livello e su di esso si va a selezionare il lume pervio del vaso a mano libera. A tal punto questa zona viene riempita di bianco e attraverso una sogliatura si ottiene l'immagine binaria. Infine, la maschera ottenuta viene esportata impostando `0` come livello di compressione e `8 bpc GRAY` come formato. In alternativa alla procedura manuale è stato elaborato un secondo algoritmo capace di svolgerla automaticamente.

3.1 Algoritmo per Registratura

```python
def register(ref_path: str, mov_path: str) -> np.ndarray:
    # reading images
    ref = io.imread(ref_path)
    mov = io.imread(mov_path)

    # perform Image Registration by Translation
    sr = StackReg(StackReg.TRANSLATION)
    out_tra = sr.register_transform(ref, mov)

    return out_tra
```

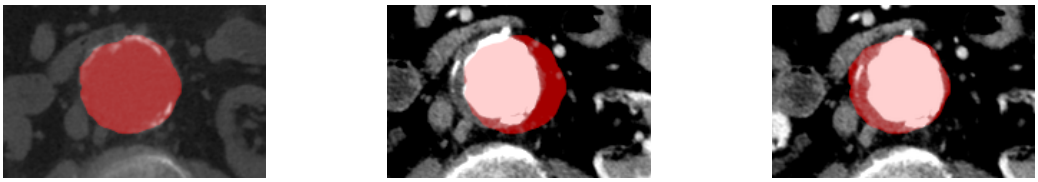
Listing 1: Funzione Python che registra automaticamente le due immagini passate usando traslazioni [2]

Come primo approccio per tentare di scrivere un algoritmo per la registrazione di immagini abbiamo usato la libreria **OpenCV**, in particolare la funzione di libreria `ORB_create()` (*Oriented BRIEF*) ci ha permesso di individuare un determinato numero di **Keypoint**, grazie ai quali abbiamo calcolato, tramite un **matching brute force**, la *Hamming norm*.

Tuttavia, il matching, data la diversa distribuzione degli istogrammi delle immagini in **Basale** e in **MDC**, è risultato poco preciso. Questo ha causato una **distorsione** in alcuni casi anche molto significativa dell'immagine in output.

Dopo aver cercato un modo di effettuare il **matching degli istogrammi** delle due immagini con scarso successo, abbiamo provato ad applicare delle trasformazioni rigide.

Abbiamo trovato la libreria `Pystackreg` che effettua una registrazione basata su traslazioni e quindi non va a distorcere l'immagine risultante.
Questo approccio di registrazione è un implementazione di una procedura descritta da *Philippe Thevenaz et al*[1].
Nel paper viene descritto un algoritmo di registrazione che va a minimizzare la **differenza quadratica media di intensità** tra la reference e il target sfruttando una variazione del *Marquardt-Levenberg Algorithm* per una *nonlienar least-square optimization*.
Questo approccio ha dimostrato di essere molto più efficace per il nostro problema.



(a) (b) (c)

Figura 1: Nella figura possiamo vedere come si sovrappone una maschera ottenuta manualmente dall'immagine in *Basale* (a) rispetto all'immagine in *MDC non registrata* (b) e all'immagine in *MDC registrata* (c).

3.2 Algoritmo per Sogliatura

```python
def process(fpath:str, low=200, upper=255, iter=5, ksize=5) -> np.ndarray:
    # read image and foce grayscale
    img = cv2.imread(fpath)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # thresholding
    _, thresh = cv2.threshold(gray, low, upper, cv2.THRESH_BINARY)

    # define kernel for OPENING operation
    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (ksize,ksize))

    # remove SALT and PEPPER noises
    thresh = cv2.medianBlur(thresh, ksize=5)

    # apply erosion to the thresholded image
    eroded = cv2.erode(thresh, kernel=kernel, iterations=iter)

    # find connected components
    totalLabels, label_ids, values, centroid =
        cv2.connectedComponentsWithStats(eroded)

    # init the output image
    output = np.zeros(thresh.shape, dtype="uint8")

    # get the image shape (width and height)
    x, y = thresh.shape

    # computes minimum of the difference in distance between
    # the centroids and the center of the image
    dists = []
    for i in range(1, totalLabels):
        xc, yc = tuple(centroid[i])

        diffx = abs(x/2-xc)
        diffy = abs(y/2-yc)

        dists.append((i, diffx+diffy))

    id, _ = min(dists, key=lambda x: x[1])

    # generate output image
    componentMask = (label_ids == id).astype("uint8") * 255
    output = cv2.bitwise_or(output, componentMask)

    # dilate the resulting image
    output = cv2.dilate(output, kernel=kernel, iterations=iter)

    return output
```

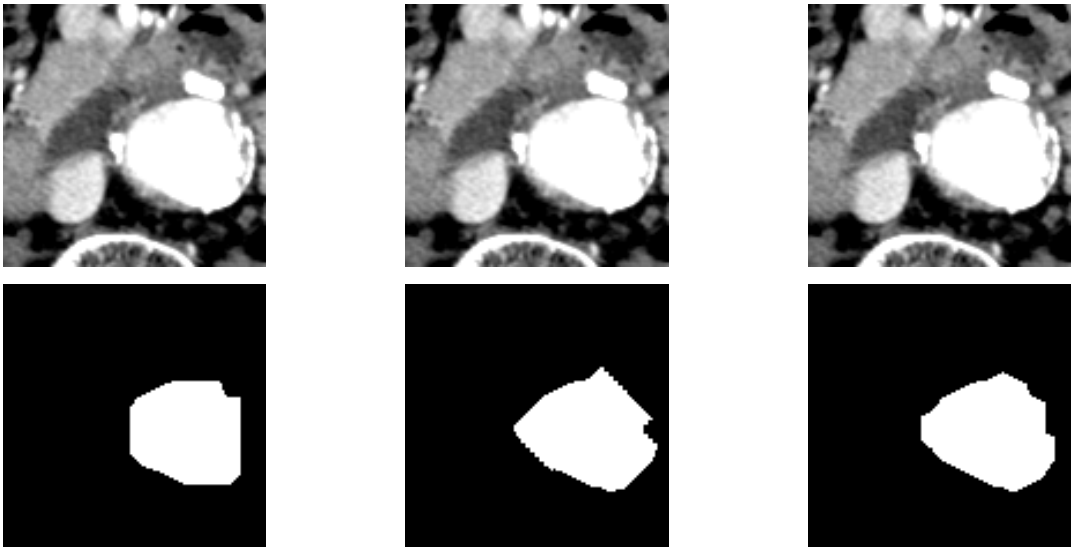
Listing 2: Funzione Python per automatizzare la generazione delle maschere del lume pervio [3]

È stato sviluppato questo algoritmo per rendere meno oneroso ed il più possibile **operator independent** il processo della creazione delle maschere, migliorando la consistenza tra immagini grazie all'uso di un processo **controllato** e **ripetibile**.
Il funzionamento dell'algoritmo si basa sui concetti appresi a lezione e applicati alle immagini utilizzando la libreria per elaborazione di immagini di Python **OpenCV**.
L'idea alla base dell'algoritmo è applicare una sogliatura che ci permetterà poi di individuare dei cluster, e in base ad un parametro di centralità ci farà estrapolare quello che rappresenta la *Region of Interest*.
Di seguito una descrizione più approfondita dell'algoritmo:

1. *Righe 3 - 4*: **Lettura** e **conversione** dell'immagine originale in scala di grigi per essere utilizzabile nei passaggi successivi.

2. *Riga 7*: **Sogliatura** dell'immagine per separare le zone più chiare (lume pervio) consentendo l'individuazione di **componenti connesse**.

3. *Riga 10*: Generazione dell'elemento strutturante da applicare nelle fasi successive di erosione e dilatazione. Effettuando varie prove si nota che un kernel di forma *ellittica* restituisce i risultati migliori, di seguito viene riportato un confronto:



(a) (b) (c)

Figura 2: Nelle immagini possiamo vedere (a) la maschera prodotta da un kernel *quadrato* di dimensione 5×5, (b) la maschera prodotta da un kernel a *croce* di dimensione 5×5 e (c) la maschera prodotta da un kernel *ellittico* di dimensione 5×5.

4. *Riga 13*: Si utilizza il filtro mediano per rimuovere il rumore *salt and pepper*, notato in alcune maschere.

5. *Riga 16*: Si utilizza un'operazione di **erosione** per separare eventuali pezzi di immagine che risultavano attaccati dopo la sogliatura, ottenendo così dei cluster migliori.

6. *Righe 19 - 20*: Si cercano le componenti connesse per individuare i cluster, inoltre grazie al metodo `connectedComponentsWithStats()` di **OpenCV** otteniamo anche delle informazioni utili, come ad esempio i centroidi del cluster, che useremo negli step successivi.

7. *Riga 23*: Si inizializza l'immagine di output.

8. *Righe 30 - 39*: Si filtrano i vari cluster trovati nelle *righe 19 - 20* in base alla loro centralità, per come è stato effettuato il crop, il cluster più vicino al centro dell'immagine è sicuramente quello che rappresenta la maschera di nostro interesse.

9. *Righe 42 - 43*: La maschera del cluster trovato nello step precedente viene inserita nell'immagine vuota creata nella *riga 23*.

10. *Riga 46*: Si effettua una **dilatazione** con lo stesso kernel utilizzato nell'operazione di erosione affinché la maschera di output venga riportata nella forma corretta.

3.3 Rete Neurale

Una rete neurale è una funzione del tipo $f(\theta, x)$, dove x rappresenta l'input della rete e θ l'insieme dei suoi parametri. Addestrare una rete neurale significa risolvere un problema di ottimizzazione del tipo

$$\min_{\theta} \Big/ \max_{\theta} \quad L(f(\theta, x), y),$$

dove L è la funzione di costo che valuta quanto l'output della rete si avvicina al dato reale y. Per il nostro scopo sono state utilizzate due reti con diverso funzionamento ma che condividono la stessa struttura. Entrambe fanno parte della famiglia di reti U-NET, che verranno approfondite in seguito. Nella prima rete si vuole minimizzare l'errore quadratico medio MSE, quindi si ha

$$\min_{\theta} \quad MSE(f(\theta, B), MDC),$$

dove B rappresenta l'immagine in basale, MDC l'immagine con mezzo di contrasto e

$$MSE = \sum_{i=1}^{M} \sum_{j=1}^{N} \frac{|I_A(i,j) - I_B(i,j)|^2}{M \times N},$$

dove I_A e I_B sono due immagini di dimensioni $M \times N$, delle quali andremo a confrontare i relativi pixel nelle posizioni (i, j).

Nella seconda rete, al contrario, si vogliono massimizzare gli indici di DICE e di Tanimoto e contemporaneamente minimizzare l'errore di Mis-Classificazione e il valore assoluto dell'indice di Bias. Il problema quindi si riduce a

$$\max_{\theta} \quad L(f(\theta, B), LP),$$

con

$$L(f(\theta, B), LP) = DCI + TI - E_m - |B_{pn}|.$$

Qui LP rappresenta la maschera del lume pervio del vaso e gli indici sono definiti come

$$DCI := \frac{2 \cdot \#(C_E \wedge MDC)}{\#C_E + \#MDC},$$

dove $\#C_E$ è il numero di pixel bianchi nella maschera estratta, $\#MDC$ è il numero di pixel bianchi nell'immagine con mezzo di contrasto e $\#(C_E \wedge MDC)$ è il numero di pixel bianchi in entrambe;

$$TI := \frac{\#(C_E \wedge MDC)}{\#C_E + \#MDC - \#(C_E \wedge MDC)},$$

cioè il rapporto tra il numero di pixel classificati correttamente e il numero totale di pixel bianchi nell'area di riferimento ed estratta;

$$E_m := \frac{\#m}{\#C_E + \#MDC - \#(C_E \wedge MDC)},$$

dove $\#m$ è il numero totale dei pixel classificati erroneamente;

$$B_{pn} := \frac{\#f_p - \#f_n}{\#t_p},$$

qui f_p è l'insieme dei falsi positivi, quindi i pixel considerati bianchi che dovrebbero essere neri, f_n è l'insieme dei falsi negativi, quindi i pixel considerati neri che dovrebbero essere bianchi e t_p è l'insieme dei veri positivi.

3.4 U-NET

U-Net è una rete neurale convolutiva sviluppata per la segmentazione rapida e precisa di immagini, il che la rende una candidata valida per l'utilizzo su immagini biomediche. Nel modello U-NET le dimensioni dell'output coincidono con quelle dell'input.
La struttura di questo tipo di rete si può dividere in due parti: una di *encoding* e una di *decoding*. Entrambe le parti possono essere scomposte in una sequenza di blocchi.

- *Encoding*: le operazioni in questa fase sono
    1. Primo **filtraggio convolutivo**
    2. Secondo filtraggio convolutivo (opzionale)
    3. Normalizzazione (opzionale)
    4. Funzione non lineare, detta funzione di attivazione
    5. **Dimezzamento** delle dimensioni

Si osserva che i pesi dei filtri convolutivi rappresentano l'oggetto dell'addestramento della rete neurale.

- *Decoding*: le operazioni svolte in questa fase sono analoghe alla precedente, con la differenza che in questo caso le dimensioni dell'output vengono raddoppiate e non dimezzate.

Unendo i blocchi delle due sequenze si ottiene una struttura a forma di U da cui il nome *U-NET*. Inoltre si possono aggiungere le **skip-connections**, ovvero l'output di ogni blocco di encoding viene concatenato con l'input del corrispondente blocco di decoding.

3.5 Addestramento Rete

Inizialmente le immagini vengono elaborate in maniera tale che tutte abbiano la stessa dimensione, condizione necessaria affinché possano essere inserite nello stesso tensore come input per la rete.

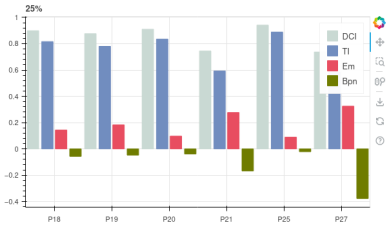
Per far sì che l'addestramento sia più efficace, questo viene effettuato con batches e quindi con più immagini alla volta. Per ogni paziente i dati vengono considerati in coppie (B+MDC), addestrare a livello di immagini significa estrarre casualmente alcune di queste coppie per comporre il batch. L'approccio alternativo sarebbe stato quello di addestrare per paziente, ma in questo caso il dataset risulterebbe troppo ridotto per la rete. D'altro canto avremmo il vantaggio che la rete si allenerebbe a riconoscere la sequenzialità, ma servirebbero molti più pazienti per agire in questo modo.

Il meccanismo di addestramento parte con l'inizializzazione dei pesi utilizzati per i filtri convolutivi. Dopo aver calcolato la funzione di loss si torna indietro con l'operazione di back propagation per aggiornare i pesi e far sì che, dopo diverse iterazioni, la **funzione di loss** converga su un **minimo locale**. Infine l'immagine di output viene riportata alla dimensione del target mediante un'interpolazione bilineare.

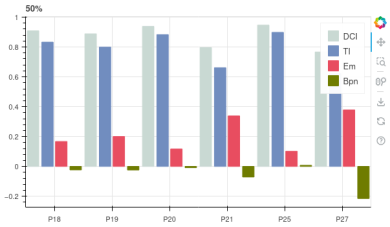
4 Risultati raggiunti

I risultati che si ottengono da entrambi le rete neurali sono raggruppati per paziente. In particolare, vengono fornite le predizioni di ogni rete e le maschere con la rimozione delle placche di calcio, effettuata con una serie di sogliature e sottrazioni. In aggiunta, vengono restituite le metriche contenenti gli indici di errori DCI, TI, B_{pn}, E_m. Per entrambe le reti si riportano gli istogrammi delle seguenti statistiche: primo quartile, mediana, terzo quartile, minimo, massimo, media e deviazione standard.

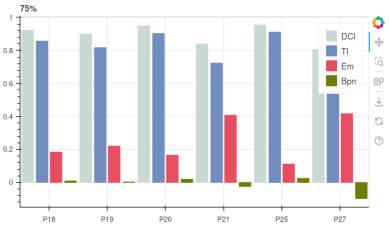
RETE 1



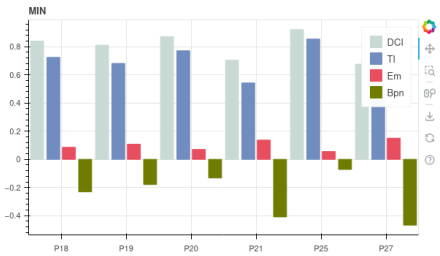
(a) Primo quartile



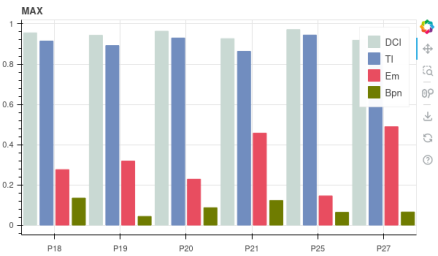
(b) Mediana



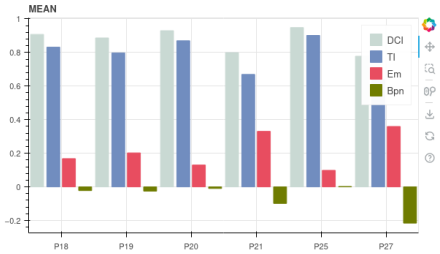
(c) Terzo quartile



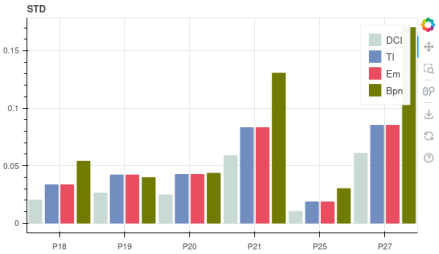
(d) Minimo



(e) Massimo

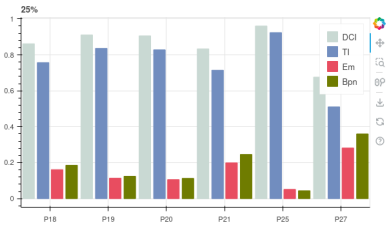


(f) Media

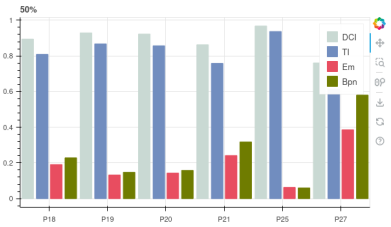


(g) Deviazione standard

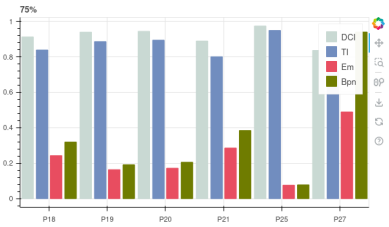
RETE 2



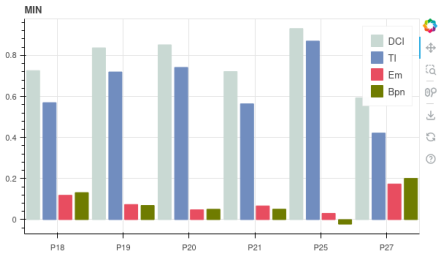
(a) Primo quartile



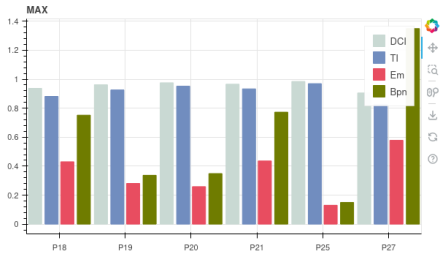
(b) Mediana



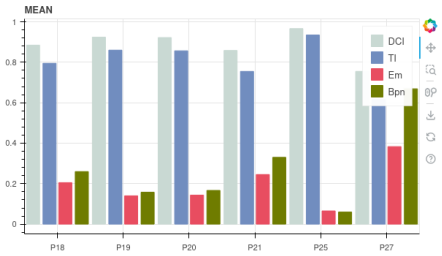
(c) Terzo quartile



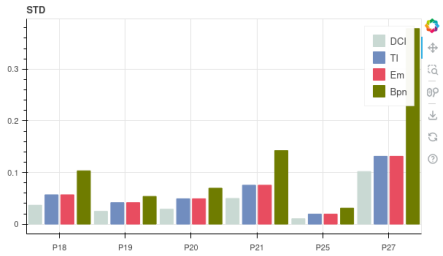
(d) Minimo



(e) Massimo



(f) Media



(g) Deviazione standard

- I tre indici di posizione riportati sono il primo quartile, la mediana e il terzo quartile, i quali forniscono informazioni sulla struttura della distribuzione dei dati. Una volta ordinati i valori degli indici, i quartili sono i tre valori che dividono l'insieme dei dati in quattro intervalli di ugual numerosità. Per il paziente 27 si osserva che la mediana dell'indice B_{pn} della prima rete assume valore pari a -0.22, mentre quella della seconda rete è circa pari a 0.6. Si nota, quindi, un notevole discostamento tra i due valori.

- I minimi ottenuti dalla prima rete per gli indici DCI, TI e E_m si accostano molto a quelli ottenuti dalla seconda rete, ad eccezione del B_{pn}. In particolare, per il paziente 27, i valori minimi del DCI nella prima e nella seconda rete si registrano in corrispondenza di 0.68 e di 0.59, rispettivamente. Per l'indice TI, invece, i valori sono 0.51 e 0.42, mentre, per l'indice E_m 0.15 e 0.17. Questa similarità non si riscontra per i minimi del B_{pn} che si registrano in corrispondenza di -0.47 e di 0.2. Considerazioni analoghe si hanno per i valori di massimo.

- Analizzando i risultati del paziente 27, si osserva che la media dell'indice di DCI oscilla tra lo 0.78 per la prima rete e lo 0.75 per la seconda. Questo significa che nel caso

peggiore, quindi con la prima rete, sono classificati bene il 78% dei pixel bianchi, mentre nel migliore si arriva a un picco del 75%. Per quanto riguarda l'indice di Tanimoto, questo si trova tra lo 0.64 e 0.62, quindi non c'è molta differenza fra le due reti. Anche i valori dell'errore di misclassificazione sono abbastanza vicini, si passa dallo 0.36 della rete 1 allo 0.38 della 2. L'indice dove il risultato è più distante è l'indice di bias, in quanto risulta essere -0.22 per la prima rete e 0.67 per la seconda. Si osserva quindi che nel primo caso la rete va a sottostimare la zona del lume pervio mentre nel secondo la va a sovrastimare. Dal punto di vista matematico, comunque, il B_{pn} è più vicino allo zero nella rete 1. Considerando invece solo i primi tre indici risulta complessivamente migliore la rete 2.

- La deviazione standard di DCI, TI e E_m nella prima e nella seconda rete assume mediamente valori bassi e questo significa che la media ben rappresenta la distribuzione. Mentre, per l'indice B_{pn} assume valori mediamente alti: 0.17 per la rete uno contro 0.37 per la due. In questo caso le osservazioni sono lontane fra di loro e quindi la distribuzione è rappresentata meglio dalla mediana piuttosto che dalla media. Calcolando il coefficiente di variazione, ovvero il rapporto tra la deviazione standard e la media, si ottiene un valore di 0.79 per il B_{pn} della rete uno, contro 0.57 della rete due.

5 Conclusioni

Una volta analizzati i risultati numerici, siamo in grado di trarre le conclusioni in merito all'addestramento delle due reti neurali. Ci concentriamo in particolar modo sul paziente 27 poiché è quello utilizzato per la validazione. I dati ottenuti sugli altri risultano essere poco rilevanti in quanto provengono da dati utilizzati durante la fase di addestramento e dunque già noti alla rete. Dagli indici DCI, TI e E_m si evince che la rete che performa meglio è la seconda. L'unica eccezione si verifica per l'indice B_{pn}, in quanto in valore assoluto si trova più vicino allo zero per la rete 1.

L'addestramento della prima rete prende come input l'immagine in basale e la corrispondente immagine MDC, per estrarre in fase di test l'immagine MDC sempre in scala di grigi. Tuttavia, per utilizzare le metriche proposte (DCI, TI, E_m, B_{pn}) si è reso necessario convertire sia l'output che l'immagine originale in maschere binarie, private delle placche di calcio, per confrontarle tra di loro. Probabilmente questo passaggio ulteriore da scala di grigi a maschera binaria ha interferito sulla loss function ed i risultati non hanno pienamente soddisfatto le aspettative. Infatti si ha un indice di misclassificazione pari al 36% e una dispersione più elevata dei dati, condizioni che portano a pensare che lavori meglio la rete 2.

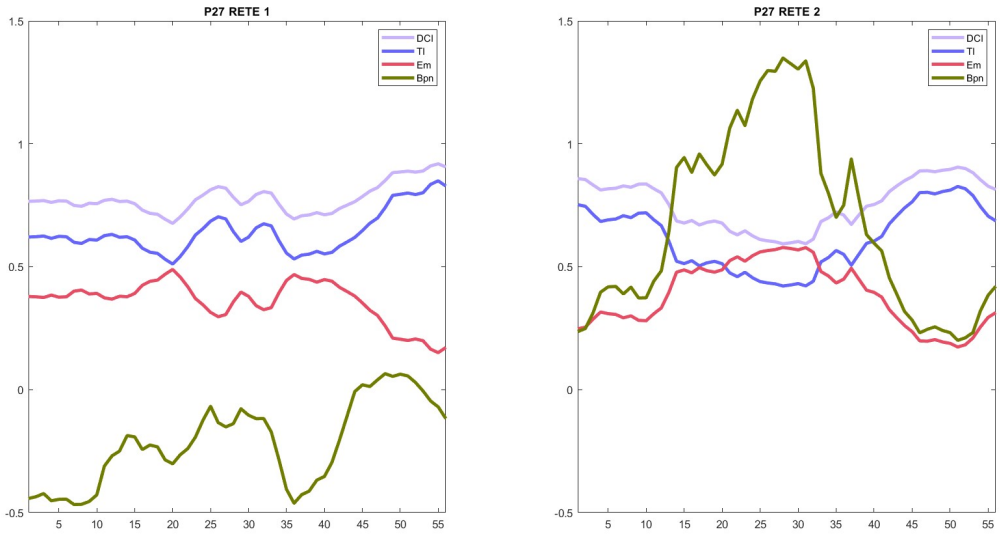
Al contrario, nella seconda rete sia l'output che l'immagine originale erano maschere binarie e quindi la funzione di loss ha lavorato su dati che erano già predisposti per il calcolo delle metriche. Nonostante ciò, l'indice di bias del paziente 27 ha evidenziato in alcune immagini, più precisamente dalla numero 13 alla 38, delle discrepanze notevoli rispetto alle altre e questo ha influenzato il risultato finale della media e della deviazione standard. Molto probabilmente questo è dato dal fatto che diverse immagini in basale prese in input dalla rete per l'addestramento, comprendevano, oltre alla ROI, anche altre sezioni che possono aver disorientato la rete. Il motivo è che per agevolare il lavoro della rete è stato deciso di impostare la stessa dimensione dell'immagine in basale per ogni paziente, cercando quella ottimale che andasse bene per tutti e rispettasse il vincolo di contenere nell'inquadratura tutta la ROI. Come riferimento è stato preso il paziente 18 in quanto presentava l'aneurisma più grande e questo giustifica il sovradimensionamento della ROI degli altri pazienti.

Per ovviare a questo inconveniente e quindi provare a migliorare i risultati delle predizioni si possono considerare diverse soluzioni.

- Dimensionare in modo adeguato le ROI di ogni paziente, indipendentemente dalle dimensioni scelte per gli altri;
- Dividere in sottosequenze lo stack dei pazienti che presentano marcate differenze tra le posizioni dell'aorta nella ROI;
- Eliminare dal database quelle immagini che presentano scostamenti eccessivi, questo provocherebbe una diminuzione del set di addestramento, a fronte di una maggiore precisione;

- Rivedere l'architettura della rete, eventualmente scegliendo un numero di epoche differente o provando la validazione con altri pazienti.

Di seguito viene riportata la rappresentazione grafica dei quattro indici per entrambe le reti.



(a) Andamento degli indici per il paziente 27

Da quanto riportato si deduce la presenza di un lieve fenomeno di overfitting. Questo concetto è fondamentale per capire quanto bene l'algoritmo di apprendimento è in grado di adattarsi a nuovi dati, cioè dati non usati in fase di addestramento. Un altro fenomeno che si riscontra è il data leakage, legato alla forte correlazione tra le immagini dello stesso paziente. Questo va a falsare i dati di validazione alla fine di ogni epoca facendo intendere una generalizzazione migliore di quella reale.

Riferimenti bibliografici

[1] Thevenaz, P.; Ruttimann, U.E.; Unser, M. (1998). A pyramid approach to subpixel registration based on intensity. IEEE Transactions on Image Processing, 7(1), 27–41. doi:10.1109/83.650848

[2] Chiara Darielli, Giovanna Donato, Martina Longaroni, Lucrezia Rinelli, Tommaso Romani, Nicolò Vescera (2023). Script python che registra automaticamente un set di immagini di riferimento con un altro set di immagini target. URL: https://github.com/Typing-Monkeys/AppliedImageProcessing-PatologieVascolari/blob/main/image_registration.py

[3] Chiara Darielli, Giovanna Donato, Martina Longaroni, Lucrezia Rinelli, Tommaso Romani, Nicolò Vescera (2023). Script python in grado di estrarre da un set di immagini in input, la relativa maschera binarizzata. URL: https://github.com/Typing-Monkeys/AppliedImageProcessing-PatologieVascolari/blob/main/mdc_to_mask.py