Sviluppo di un algoritmo di segmentazione 2D su immagini CT scan di polmoni di pazienti affetti da COVID-19

Autori: Damiano Barcaro, Andrea Belli Contarini, Alessandro Ciorra (Gruppo 24)

Metodi di Intelligenza Artificiale e Machine Learning per la Fisica (A.A. 2021-2022)

1 Abstract

Il progetto si pone l'obiettivo di sviluppare un algoritmo di segmentazione 2D per le immagini CT di polmoni di pazienti con COVID-19, cioè un algoritmo in grado di individuare quali regioni nell'immagine corrispondano a diverse tipologie di tessuto polmonare. Ne verranno inoltre testate le prestazioni utilizzando le figure di merito più opportune per task di segmentazione, ottenendo un valore medio (tra tutte le classi) della metrica *Intersection over Union* pari a $IoU = 0.76$ (tra *ground truth mask* e *predicted mask*).

2 Introduzione: descrizione del problema

A fine gennaio 2020 l'Organizzazione Mondiale della Sanità (OMS) ha dichiarato che il COVID-19 è un'emergenza sanitaria pubblica di interesse internazionale. Fino a questo punto, nessun trattamento efficace è stato ancora dimostrato per la cura del COVID-19. Pertanto, per una tempestiva prevenzione della sua diffusione , test accurati e rapidi sono estremamente fondamentali. Tuttavia, la carenza di test disponibili e apparecchiature di prova in molte aree del mondo limita uno screening rapido e accurato di soggetti sospetti. Anche nelle migliori circostanze, l'ottenimento dei risultati di test quali la RT-PCR (reazione a catena della polimerasi a trascrizione inversa) richiede più di 6 ore e la sensibilità raggiunta è insufficiente. D'altro canto, le tecniche di imaging radiologico come l'impiego dei Raggi X e tomografia computerizzata (TC) seguita dall'analisi automatizzata delle immagini può integrare con successo la RT-PCR. Lo screening TC fornisce una vista tridimensionale del polmone ed è quindi più sensibile (sebbene meno ampiamente disponibile) rispetto alla radiografia del torace.
In questo progetto, presentiamo un framework basato sul deep learning per la segmentazione automatica di aree di tessuto associate al COVID-19 patologico da immagini TC cliniche disponibili da set di dati di segmentazione COVID-19 pubblicamente disponibili.

3 Descrizione del dataset

I dati sono quelli del database cinese *2019nCoVR*, scaricabili al link[1].
Si tratta di un campione di 750 immagini jpeg (di *size* 512×512) con maschere (*ground truth* di segmentazione), rappresentato da immagini della stessa shape di quelle del CT scan, ma con valori interi dei singoli pixel pari a 0, 1, 2, 3 (più avanti verrà spiegato il significato di tali valori), opportunamenrte riscalati dal momento in cui si utilizzerà la scala di grigi per le immagini.

Le immagini sono come questa:

Figura 1: *Slice* di un polmone di una paziente colpito da polmonite interstiziale da COVID-19; la freccia indica una zona del polmone che prende il nome di *glass opacity*, ossia una zona non più funzionale del polmone.

[1] `wgethttp://giagu.web.cern.ch/giagu/CERN/ct_lesion_seg.zip`

Per ogni paziente e per ognuna delle *slice* viene fornita, per ogni pixel, una label (fatta da medici) che ci dice se il particolare pixel è polmone normale (*pixel value* = 1), *glass opacity* (*pixel value* = 2), *consolidation* (zone che hanno completamente perso ogni funzione del polmone, *pixel value* = 3), oppure zona non polmonare (o di *background*, *pixel value* = 0).
L'obiettivo sarà proprio quello di addestrare una DNN per fare segmentazione di queste immagini.

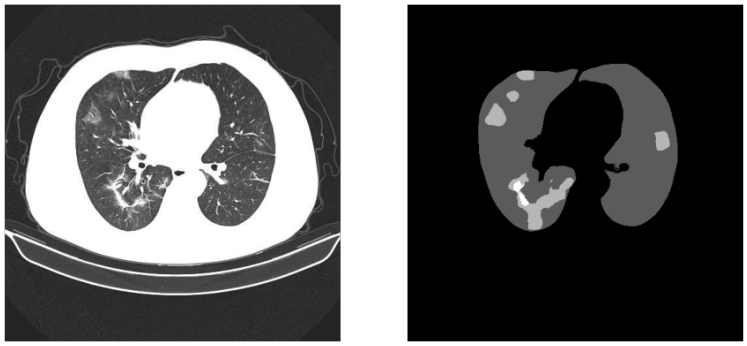

Figura 2: *image* del polmone e *ground truth mask* associata, a titolo di esempio. Si notino le varie tonalità di grigio per la *mask*, ad indicare appunto le varie zone citate prima.

4 Strategia

4.1 Data augmentation

Importiamo inizialmente le immagini in scala di grigi e le normalizziamo.
Avendo a disposizione relativamente poche immagini, onde evitare il problema di *overfitting*, abbiamo eseguito una *data augmentation*, ruotando le immagini di $\theta = 45°$ e "ribaltandole" verticalmente ed orizzontalmente (*VerticalFlip* ed *HorizontalFlip*). Così facendo viene quadruplicata la taglia del nostro dataset, passando da 750 *images*, con le relative 750 *masks*, a 3000 *images* e 3000 *masks*. Otteniamo così i dataset *IMAGE* e *MASK*, entrambi di taglia pari a $750 \cdot 4 = 3000$, su cui poi eseguiremo uno *split* (randomico) in train (80%), test (10%) e validation (10%).

4.2 U-Net

U-Net è una *Convolutional Neural Network* che tipicamente si usa per la segmentazione di immagini biomediche ed è proprio quella che implementeremo nel nostro lavoro.

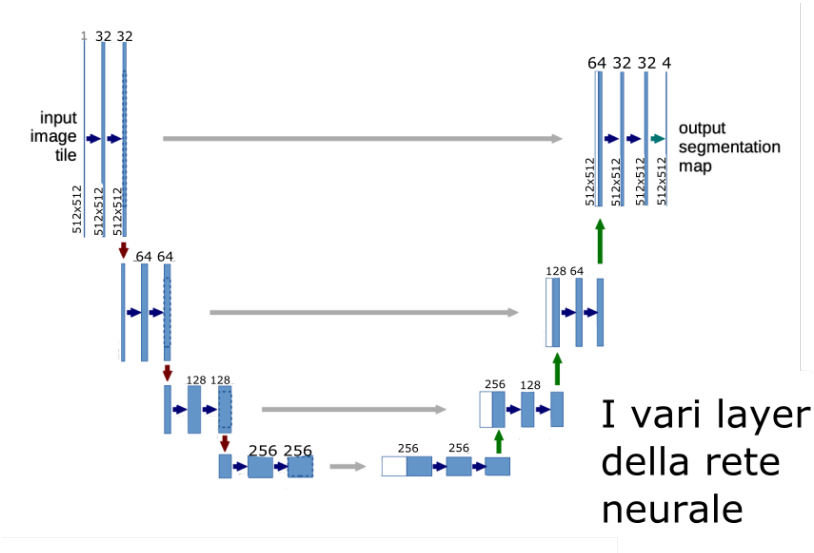

Figura 3: Schema dell'architettura *U-Net* utilizzata. Ogni box azzurro corrisponde ad una *multi-channel feature map*. Il numero di *channels* è scritto sopra ogni box; sotto ognuno di essi è poi fornita la *x-y-size*. Le frecce colorate corrispondono alle differenti operazioni effettuate (ad esempio: blu ↔ *conv2d+ReLu*, rossa ↔ *max pooling*).

E' basata su architettura neurale encoder-decoder e uso di *skip connections*. Nella parte dell'encoder, il modello ottiene un'immagine come input e applica più livelli di convoluzione, *max-pooling* e attivazione *ReLU* (*Rectified Linear Unit*) e comprime i dati in uno spazio latente. Nel decoder, la rete tenta di decodificare le informazioni dallo spazio latente usando l'operazione di convoluzione trasposta (deconvoluzione) e produrre la maschera di segmentazione dell'immagine. Il resto delle operazioni sono simili a quelle sopra menzionate nella parte dell'encoder. Il modello *U-Net*, inoltre, prevede l'uso di *skip connections* per inviare le informazioni dai corrispondenti strati ad alta risoluzione dall'encoder al decoder, aiutando così la rete a catturare meglio i piccoli dettagli presenti in alta risoluzione.

5 Ottimizzazione iperparametri e training del modello

- Come ***loss*** utilizziamo la *CrossEntropy*, che, tra due distribuzioni discrete p e q (che fissata p misura quanto q differisce da p), è definita da:

$$H(p,q) = -\sum_{x} p(x) log\big[q(x)\big] \tag{1}$$

  Vengono poi introdotti dei pesi per ridurre ancora il rischio di *overfitting* del *training set* (portando così ad una migliore generalizzazione). Ciò viene fatto in maniera automatica, solamente passando un tensore *torch* alla classe *torch.nn.CrossEntropyLoss()* definita in *Pytorch*.

- Come **ottimizzatore** utilizziamo *Adam*, con un *learning rate* pari a $LR = 1{\times}10^{-3}$.

- Fissiamo il *batch-size* a 5 e il numero delle epoche a 35; l'addestramento, così, durerà circa 100 minuti.

- Per valutare la performance dell'algoritmo di segmentazione, utilizziamo la **metrica** *Intersection over Union* ($IoU \in [0,1]$). Date 2 *masks*: y_{true} (*ground truth mask*) ed y_{pred} (*prediceted mask*), dunque, valutiamo:

$$IoU(y_{true}, y_{pred}) = \frac{y_{true} \cap y_{pred}}{y_{true} \cup y_{pred}} \tag{2}$$

Le immagini in input e le rispettive maschere di segmentazione sono usate per addestrare la rete attraverso l'implementazione della discesa stocastica lungo il gradiente.

6 Analisi dei risultati

6.1 Andamento della *loss*

Mostriamo ora l'andamento della *training loss* e della *validation loss* al variare delle varie epoche:

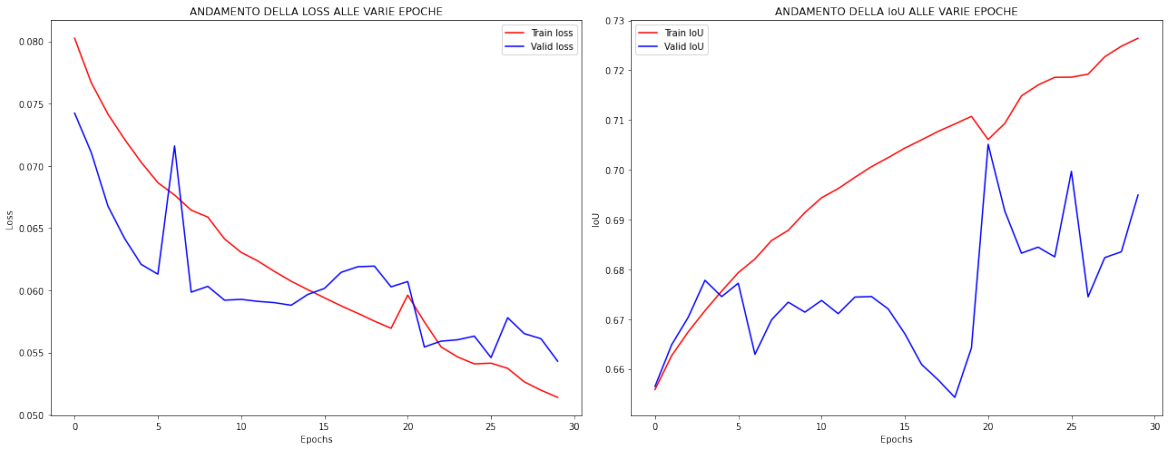

Figura 4: *Training loss* (in rosso) e *validation loss* (in blu) a sinistra, *training IoU* (in rosso) e *validation IoU* (in blu) a destra.

Si nota che si ha una convergenza abbastanza buona tra le due *loss*, dopo circa 30 epoche; la rete non sembra dunque soffrire di problemi di *underfitting* o *overfitting*.

Riportiamo nella figura che segue, la *image* (*slice* del polmone), la relativa *ground truth mask* e la *predicted mask*, ossia la maschera effettivamente "predetta" dal nostro algoritmo.

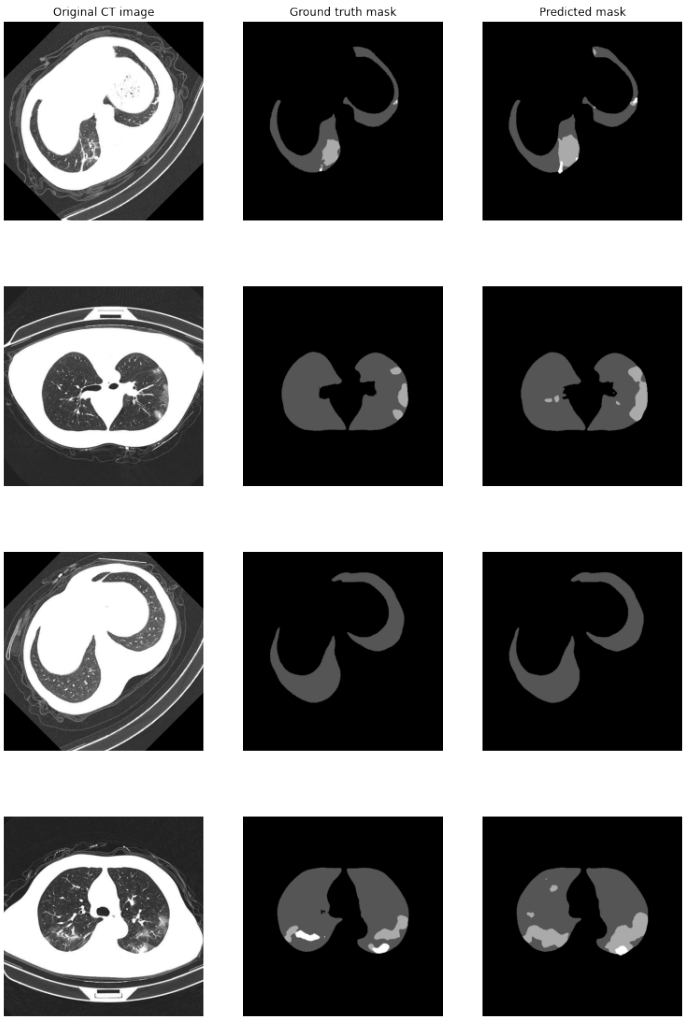

Figura 5: In ordine da sinistra a destra: *image*, *ground truth mask*, *predicted mask*.

Anche ad occhio è possibile apprezzare come la *mask* di predizione sia abbastanza smile a quella vera creata dai medici.
Procediamo adesso con una analisi dettagliata dei risultati ottenuti e dell'efficacia dell'algoritmo.

6.2 Metrica e *test loss*

Volendo effettivamente misurare la performance del nostro algoritmo di segmentazione, misuriamo la IoU (definita in formula 2) su tutte le classi a fine *training loop*. Otteniamo dunque:

$$\boxed{IoU = 76\%}$$

che essendo comunque maggiore del 50% ci fa pensare che l'algoritmo è in grado di individuare le varie tipologie di tessuto polmonare in maniera abbastanza efficacie.

Inoltre si ottiene una *test loss*:

$$\boxed{TEST\ LOSS: = 19\%}$$

7 Conclusioni

La rete svolge abbastanza bene il *task* desiderato, con una IoU, per valutare la bontà della predizione, pari al 76%.

Sarebbe possibile ottenere risultati migliori aumentando il numero di epoche e rendendo la rete neurale più profonda, ad esempio aggiungendo un altro blocco convoluzionale sia scendere che a salire nella *U-Net*. Tuttavia questo crea problemi o per alla RAM o alla GPU (non utilizziamo un *Google Colab Pro*).
L'esecuzione prolungata del codice incontra problemi di *runtime* e problemi legati alla stabilità della connessione, per cui non risulta affidabile lanciare *runtime* più lunghi di 100 minuti.