

ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

SCUOLA DI INGEGNERIA E ARCHITETTURA

Dipartimento di Informatica  
Scienza e Ingegneria  
DISI

**Corso di Laurea in Ingegneria Informatica**

**Classificazione della severità di patologia da  
Covid-19 mediante Transfer Learning su  
dataset eterogeneo**

Candidato:

*Gianmiriano Porrazzo*

Relatore:

*prof. Andrea Camisa*

Correlatori:

*dott. Andrea Testa  
prof. Giuseppe Notarstefano*

Anno Accademico  
*2021–2022*

Sessione  
*II*

# Indice

<b>Elenco delle figure</b>	<b>4</b>
<b>Introduzione</b>	<b>5</b>
<b>1 Conoscenze preliminari</b>	<b>8</b>
1.1 Rete neurale e come si allenano . . . . .	8
1.1.1 Neuroni nelle reti neurali . . . . .	9
1.1.2 Tipologie di Layer . . . . .	10
1.1.3 Segmentazione delle immagini e U-Net . . . . .	12
1.2 Codifica one-hot per dati categorici . . . . .	13
1.3 Strumenti usati . . . . .	14
<b>2 Descrizione dataset radiografie polmoni</b>	<b>15</b>
2.1 Composizione . . . . .	15
2.2 Immagini delle radiografie dei pazienti . . . . .	16
2.3 Segmentazione immagini mediante U-Net . . . . .	18
2.4 Selezione delle sezioni di interesse nelle immagini . . . . .	19
2.5 Metadati relativi alle radiografie dei pazienti . . . . .	20
<b>3 Risultati training con sole immagini</b>	<b>23</b>
3.1 Creazione rete neurale . . . . .	23
3.2 Transfer Learning . . . . .	24
3.3 Risultati dei test effettuati per il transfer learning . . . . .	25
3.4 Fine Tuning . . . . .	26
3.5 Risultati dei test effettuati per il fine tuning . . . . .	27
3.6 Overfitting e Data Augmentation . . . . .	28
3.7 Risultati dei test effettuati per la data augmentation . . . . .	30
<b>4 Risultati training rete neurale con dati eterogenei</b>	<b>33</b>
4.1 Selezione di un sottinsieme completo del dataset . . . . .	33
4.2 Struttura rete neurale . . . . .	35
4.3 Risultati dei test effettuati per la rete composta . . . . .	37
4.4 Confronto tra le varie reti adoperate nel progetto di tesi . . . . .	40

<b>Conclusioni</b>	<b>42</b>
<b>Bibliografia</b>	<b>43</b>

# Elenco delle figure

1.1	Esempio di architettura di rete neurale . . . . .	9
1.2	Esempio di neurone . . . . .	10
1.3	Esempio di dense layer . . . . .	10
1.4	Esempio di convolutional layer . . . . .	11
1.5	Esempio di Global Average Pooling layer . . . . .	11
1.6	Esempio di architettura di una U-Net . . . . .	13
2.1	Campione del dataset per evidenziare le differenze tra le immagini, con relativa label . . . . .	16
2.2	Effetti dei vari passaggi sulle immagini del dataset . . . . .	19
3.1	Test effettuato usando come learning rate $10^{-3}$ . . . . .	25
3.2	Test effettuato usando come learning rate $10^{-4}$ . . . . .	26
3.3	Test effettuato usando come learning rate $10^{-3}$ . . . . .	27
3.4	Test effettuato usando come learning rate $10^{-4}$ . . . . .	28
3.5	Immagini a seguito dell'augmentation, con relativa label . . . . .	29
3.6	Training effettuato con learning rate pari a $10^{-3}$ a seguito dell'augmentation . . . . .	30
3.7	Training effettuato con learning rate pari a $10^{-4}$ a seguito dell'augmentation . . . . .	31
4.1	Rappresentazione della rete finale composta da CNN e MLP .	36
4.2	Test effettuato usando come learning rate $10^{-3}$ . . . . .	37
4.3	Test effettuato usando come learning rate $10^{-4}$ . . . . .	38

# Introduzione

Il machine learning ha vari campi applicativi. Uno di questi è quello di riconoscere dei pattern in modo che tali algoritmi possano apprendere e fare predizioni su un insieme di dati. Un settore nel quale gli algoritmi di machine learning sono molto applicati è quello della medicina. In questo documento si tratterà di come usare gli algoritmi di machine learning per fare predizioni su un dataset che contiene dati relativi a pazienti affetti da Covid-19.

Esistono vari metodologie per analizzare i dati e fare apprendere la rete neurale in modo che le predizioni da essa effettuate abbiano senso. Tali metodologie possono variare per via dei dati che compongono il dataset, ma anche dal tipo di previsioni che la rete deve effettuare. L'algoritmo usato consiste in una rete neurale artificiale. Tale rete è composta da vari neuroni, i quali hanno il compito di prendere dei dati in input, apprenderne le caratteristiche principali e sulla base di ciò effettuare le previsioni. I neuroni che si occupano dell'apprendimento sono organizzati in vari layer nascosti. Ogni layer è composto da uno o più neuroni che interagiscono tra di loro. Come già accennato in base al tipo di dato da studiare si può prediligere una tipologia di rete ad un'altra. Nel caso di questa tesi si è scelto di usare una rete neurale convoluzionale (*CNN-Convolutional Neural Network*) per apprendere dalle immagini e una MLP (*Multi-Layer Perceptron*) per gestire i metadati relativi alle immagini.

Il dataset usato nell'elaborato, per allenare le reti neurali, contiene immagini raffiguranti radiografie del torace di pazienti affetti da Covid-19 e altri dati relativi al loro stato di salute all'arrivo in ospedale. Per tale motivo si è cercato di suddividere le reti da creare in base agli input che prende.

Il primo passo per implementare la CNN è comprendere meglio come il dataset delle immagini è composto, per tale motivo si è effettuata un'analisi accurata di tali immagini. Da tale analisi è emerso che le immagini sono molto eterogenee e che necessitano di alcuni accorgimenti prima di poter essere date in input alla rete. Questa serie di passi prende il nome di preprocessing delle immagini. Nel caso considerato tale preprocessing prevede:

regolazione di alcune caratteristiche dell’immagine (contrasto, luminosità,...), riconoscimento dei polmoni mediante una rete apposita (U-Net) e la creazione dell’immagine finale mediante l’uso di bounding box.

Per quanto riguarda le ultime due fasi sono effettuate partendo dall’uso di una rete convoluzionale, che ha come scopo quello di eseguire la segmentazione dell’immagine per trovare la posizione dei polmoni, dalla quale otteniamo delle maschere, ovvero immagini in cui si nota il contorno del soggetto dell’immagine. Da tali maschere si ricava poi la bounding box, il più piccolo riquadro contenente il soggetto d’interesse. Grazie a questi due artefatti si ottiene l’immagine finale, di qualità migliore rispetto all’originale. In tal modo allenare la rete convoluzionale che deve effettuare previsioni risulta più semplice poiché tali immagini, sono alla fine del processo appena descritto, ridimensionata ad una dimensione e sono incentrate sui polmoni.

Ottenute le immagini si procede con l’allenare la rete neurale convoluzionale e, in base ai risultati ottenuti, si decide se applicare tecniche per migliorare l’apprendimento della rete. Nel nostro caso sono state usate la Data Augmentation, il Transfer Learning e il Fine Tuning.

La Data Augmentation è una tecnica usata per modificare immagini presenti nel dataset applicando alcune trasformazioni ad esse, in modo che la rete non si abitui alle immagini e restituisca previsioni più accurate.

Il Transfer Learning consiste nell’uso di una rete pre-allenata per affrontare problemi correlati a quelli per cui era stata allenata. In tal modo la rete, in seguito ad opportuni accorgimenti e nuovi allenamenti, risulta essere più precisa. Uno di tali accorgimenti consiste nel fine tuning che permette di allenare gli ultimi layer della rete, ovvero quelli relativi alla classificazione, in modo da essere performanti sul nuovo dataset.

Per quanto riguarda la gestione dei metadati, si parte, anche in questo caso, con l’analizzare il dataset che ci viene fornito. A seguito di tale analisi si verifica che il dataset non è completo, ovvero presenta dati mancanti. Per tale motivo si decide di trovare un sottinsieme del dataset contenente il maggior numero di dati, in modo che questo sia completo. A questo punto si procede con la creazione del multilayer perceptron, ovvero una piccola rete neurale composta da pochi layer con il compito di gestire gli input e di classificare gli output nel modo corretto. Prima di passare i dati in ingresso al multilayer perceptron questi devono essere convertiti in un formato comune per via del fatto che, come si vedrà più nel dettaglio, i dati scelti sono di varia natura.

Terminata tale fase si è pronti per concatenare le reti create e gestire i dati mediante un’unica rete che gestisce sia le immagini che gli altri dati relativi

al paziente.

Questo elaborato di tesi inizia col trattare la struttura relativa al dataset considerato ed usato per allenare le reti neurali, descrivendo le varie tecniche usate per ovviare ai vari problemi che sono stati riscontrati durante il lavoro. Successivamente si procede col dettagliare le modalità con cui sono state implementate ed allenate le reti usate, fornendo le motivazioni di tali scelte ed i grafici per confrontare i risultati ottenuti. Si inizia col trattare la rete che gestisce le immagini come input (CNN), successivamente si descrive la tecnica usata per creare un multilayer perceptron in base ai dati che verranno passati in input ed, infine, sono state combinate queste due reti al fine di crearne una più accurata nelle previsioni poiché prende in input sia i dati che le immagini. Infatti, come si osserverà alla fine, si è passati dal 55% dei primi training con le sole immagini all'82% finale ottunuto usando dati eterogenei.

# 1 | Conoscenze preliminari

## 1.1 Rete neurale e come si allenano

Le reti neurali, spesso chiamate anche reti neurali artificiali o simulate, sono algoritmi con nomi e strutture ispirate al cervello umano ed hanno lo scopo di ricreare il modo in cui i neuroni umani comunicano. Le reti artificiali sono, nella maggior parte dei casi, sistemi adattivo, ovvero possono cambiare la propria struttura in base alle informazioni che circolano all'interno della rete. Una rete neurale non è altro che una funzione matematica. Una rete neurale solitamente è definita da una serie di neuroni (vedi sezione 1.1.1) connessi tra di loro. Usualmente le reti artificiali sono composte da più livelli: un livello relativo alla gestione dell'input, uno o più livelli nascosti ed un livello per la gestione degli output. Ogni livello è composto da nodi o neuroni artificiali. La connessione tra ognuno di questi nodi avviene mediante l'uso di soglie e pesi. Se, ad esempio, l'output generato da un nodo è al di sopra di una soglia specificata, il nodo viene attivato in modo da inviare dati al livello successivo della rete, altrimenti non vengono inviati dati.

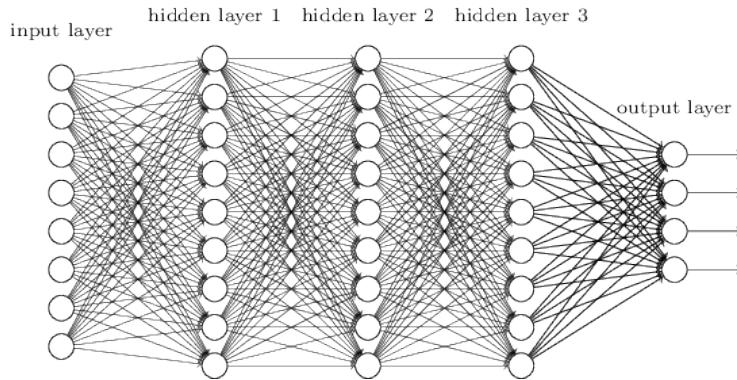
Ogni neurone è composto da dati di input, pesi, una soglia (detta anche bias) e dai dati restituiti in output. una volta che le dimensioni dei dati in ingresso vengono determinati, si assegnano anche i pesi. I pesi sono fondamentali, poiché aiutano a determinare l'importanza di ogni variabile in ingresso al neurone: pesi più grandi indicano che la varabile ha maggior importanza nella determinazione dell'output, pesi minori indicano che la variabile è meno importante per la determinazione dell'output.

Altro valore di rilievo è quello relativo alla soglia, poiché se il valore determinato come output supera la soglia, il nodo viene attivato. In tal modo vengono passati i dati al neurone successivo. Così l'output di un nodo diventa l'input del successivo.

All'inizializzazione della rete, i pesi vengono scelti in maniera casuale, portando la rete ad avere scarsi risultati. Allenare una rete significa passare da una rete poco performante ad una con un accuratezza elevata. Ciò è reso

possibile dal fatto che possiamo modificare la funzione che gestisce la rete, effettuando degli accorgimenti sui pesi.

Essendo la rete neurale una funzione, quando alleniamo la rete per avere prestazioni migliori, ciò che facciamo è minimizzare la funzione di perdita (loss function). Esistono vari algoritmi per ottimizzare funzioni, questi possono essere basati sul gradiente o meno.



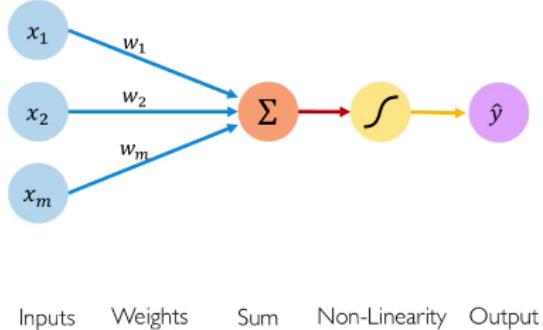
**Figura 1.1:** Esempio di architettura di rete neurale

In virtù del fatto che l’obiettivo è trovare il minimo della funzione, vengono effettuati tentativi ad ogni iterazione dell’allenamento. Un parametro fondamentale in questo processo è il *learning rate*. Tale parametro indica la grandezza del salto che viene effettuato ad ogni iterazione per avvicinarsi al minimo della funzione. Per tale motivo se tale valore è troppo elevato, saranno effettuati salti troppo grandi con il rischio di superare il minimo e conseguente divergenza dell’algoritmo. Se, al contrario si usa un learning rate troppo piccolo i tempi per trovare il minimo potrebbero aumentare ed il minimo a cui si converge potrebbe essere un minimo locale.

### 1.1.1 Neuroni nelle reti neurali

Un neurone non è altro che una funzione matematica. Tale funzione ha un input, di dimensione determinata, e restituisce un output. Sono, inoltre, presenti dei coefficienti moltiplicativi fondamentali per determinare l’output chiamati pesi. I pesi diversificano i neuroni e vengono determinati durante l’allenamento della rete, ovvero la fase in cui si regolano la rete.

La funzione che descrive un neurone può essere divisa in due parti: combinazione lineare di input e pesi preceduta da una funzione non lineare. La funzione non lineare più usata è ReLU (Rectified Linear Unit). Tale funzione risulta essere pari a zero se  $x$  è minore di zero e restituisce  $x$  se tale



**Figura 1.2:** Esempio di neurone

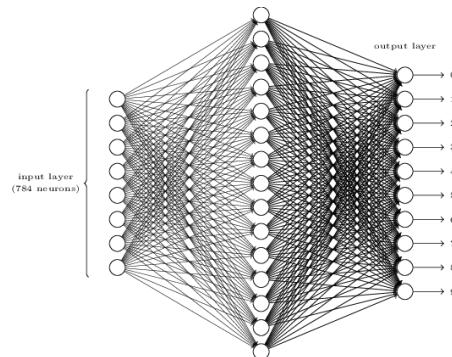
valore è maggiore di zero. Oltre a ReLU esistono altre funzioni, tra cui: Sigmoid, Softmax, entrambe usata anche nel progetto di tesi, SELU e GELU. Di seguito viene riportata l'espressione matematica descrivente la funzione ReLU.

$$\text{ReLU}(x) = \max(x, 0)$$

Infine è utile ricordare che il valore restituito dal neurone è indicato col termine attivazione, poiché determina se il neurone verrà attivato, passando i dati al neurone successivo nella catena, o meno.

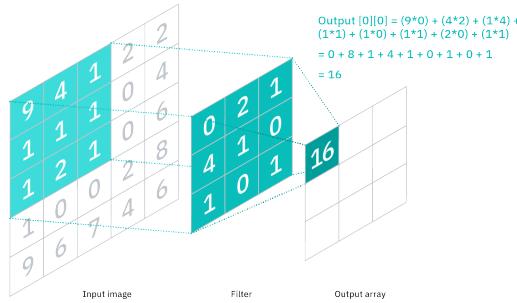
### 1.1.2 Tipologie di Layer

Esistono vari tipi di layer usati nella creazione delle reti neurali. I tipi più comuni di layer sono quattro: Fully connected, Convolutional, Deconvolutional e Recurrent.



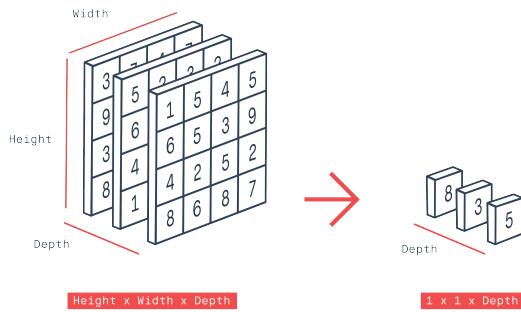
**Figura 1.3:** Esempio di dense layer

Nel caso considerato si è deciso di usare, in particolare, layer fully connected e convolutional. I layer fully connected sono caratterizzati dalla presenza di connessioni tra i neuroni di un layer e i tutti i neuroni del layer successivo. Tale tipologia di layer si trova in varie reti neurali, tra cui le reti convoluzionali. Spesso tali layer sono chiamati anche Dense layer. I layer convoluzionali sono i layer principali all'interno delle reti convoluzionali, a cui danno il nome. Sono spesso usati per ricavare caratteristiche nelle immagini grazie all'uso di filtri per analizzare l'immagine stessa prendendo pochi pixel alla volta e restituendo in output una mappa delle caratteristiche.



**Figura 1.4:** Esempio di convolutional layer

È importante citare anche un'altra tipologia di layer sfruttata all'interno del progetto. Questo è chiamato Global Average Pooling. Lo scopo di tali layer è rimpiazzare i dense layer mediante un'operazione di raggruppamento. L'idea di base è creare mappe delle caratteristiche per ogni categoria di cui si deve effettuare la classificazione. Al posto di aggiungere dei dense layer all'inizio di tale mappa, si esegue una media di ogni mappa e il vettore risultante viene passato in un layer softmax in modo da ottenere come risultato un tensore di dimensione uno (ossia una sequenza di dati).



**Figura 1.5:** Esempio di Global Average Pooling layer

### 1.1.3 Segmentazione delle immagini e U-Net

Nel caso di classificazione delle immagini la rete neurale assegna un'etichetta (o classe) ad ogni immagine in input. Se si vuole conoscere la forma di un oggetto presente nell'immagine, quali pixel appartengono a quale oggetto o cose simili, si deve assegnare una classe ad ogni pixel dell'immagine e non all'immagine intera. Tale attività è conosciuta con il nome di segmentazione. Grazie a tale tecnica tutti i pixel appartenenti allo stesso oggetto avranno la stessa etichetta e sarà più facile compiere analisi più dettagliate sull'immagine.

Nel caso del database usato, si è interessati ai soli polmoni del paziente per tale motivo è utile effettuare segmentazione in modo da distinguere ciò che fa parte dal polmone dal resto.

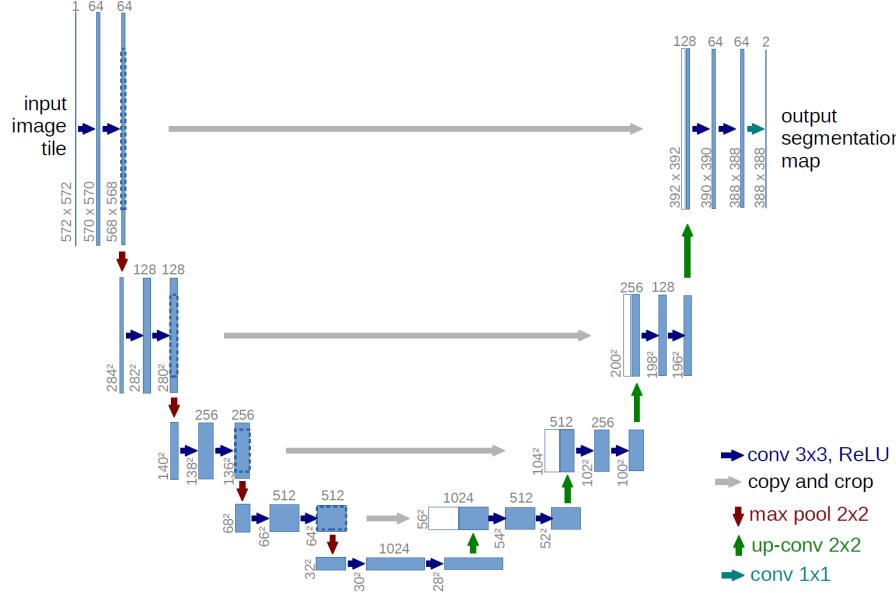
Esistono vari modi per effettuare segmentazione delle immagini tra cui:

- Segmentazione basata su soglia
- Segmentazione basata sulle regioni
- Segmentazione basata sui margini
- Segmentazione basata su gruppi
- Segmentazione basata su reti artificiali

Nel caso d'interesse si è scelto di usare la segmentazione basata su reti artificiali. Per tale motivo è stato necessario usare una rete neurale apposita per effettuare segmentazione. Tale tipologia di rete è conosciuta con il nome di U-Net. Le U-Net sono reti neurali convoluzionali specializzate nella segmentazioni di immagini mediche.

La U-Net è stata usata per creare delle maschere, ovvero immagini che separano ciò che è all'interno del polmone con ciò che non appartiene ai polmoni. Dunque la previsione che tale rete produce è un'immagine con la stessa dimensione dell'immagine iniziale.

Dalla maschera creata sarà poi più semplice effettuare ulteriori analisi e modifiche sull'immagine iniziale.



**Figura 1.6:** Esempio di architettura di una U-Net

## 1.2 Codifica one-hot per dati categorici

La codifica one-hot risulta essere molto utile quando si è in presenza di dati categorici. Tale tipologia di dati contengono etichette relative al dato invece che valori numerici: ogni etichetta solitamente rappresenta categorie diverse. Il problema legato a tale tipologia di dati è che una rete neurale non riesce ad operare direttamente c sulle etichette dei dati, poiché richiedono che tutte le variabili in ingresso ed uscita siano numeriche. Per tale motivo si necessita i dati categorici devono essere convertiti in numerici. La conversione da dati categorici a numerici prevede due passi:

- Integer Encoding
- One-Hot Encoding

Il primo passo prevede che ad ogni categoria venga assegnato un valore numerico, questo potrebbe in alcuni casi essere sufficiente per convertire i dati. Il secondo passo prevede la conversione dei valori numerici in valori binari. Ciò che solitamente accade è che vengono scelti tanti bit quanti sono le categorie da rappresentare e, successivamente, ogni categoria verrà rappresentata da una serie di bit posti come 0 ed un unico bit posto ad 1. Di seguito si

riporta un esempio di una codifica one-hot sulle categorie utilizzate.

$$\begin{aligned} 0 &\longrightarrow \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \\ 1 &\longrightarrow \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \\ 2 &\longrightarrow \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \end{aligned}$$

Tale processo viene usato all'interno dell'elaborato di tesi al momento dello studio del dataset eterogeneo e la realizzazione del multilayer perceptron.

### 1.3 Strumenti usati

Esistono vari framework per lo sviluppo di reti neurali, uno dei più usati è TensorFlow [1].

TensorFlow presenta modelli già creati e allenati, i cui pesi possono essere reperiti per effettuare test in maniera rapida. Insieme all'uso di altre librerie come Keras, l'implementazione di una rete neurale che gestisca i dati a disposizione è molto semplificato.

La rete di base usata per partire con l'analisi delle immagini è la MobileNetV2. Tale rete è una CNN che, prendendo come input delle immagini. A tale rete, tuttavia, sono state apportate delle modifiche per via delle dimensioni dell'input e della tipologia di dato che si vuole ottenere come previsione.

Oltre a questa tipologia di rete si è sfruttata anche una U-Net, al fine di omogeneizzare tutte le immagini che vengono date come input della MobileNetV2. La U-Net usata è EfficientNet, con dei pesi preallenati [2] per riconoscere immagini simili a quelle presenti nel dataset di riferimento.

## 2 | Descrizione dataset radiografie polmoni

### 2.1 Composizione

Il dataset usato per effettuare lo studio è stato preso da un hackaton [3] riguardante la creazione di una rete neurale in grado di apprendere da immagini relative a radiografie di pazienti affetti da Covid in modo da predire la gravità della prognosi.

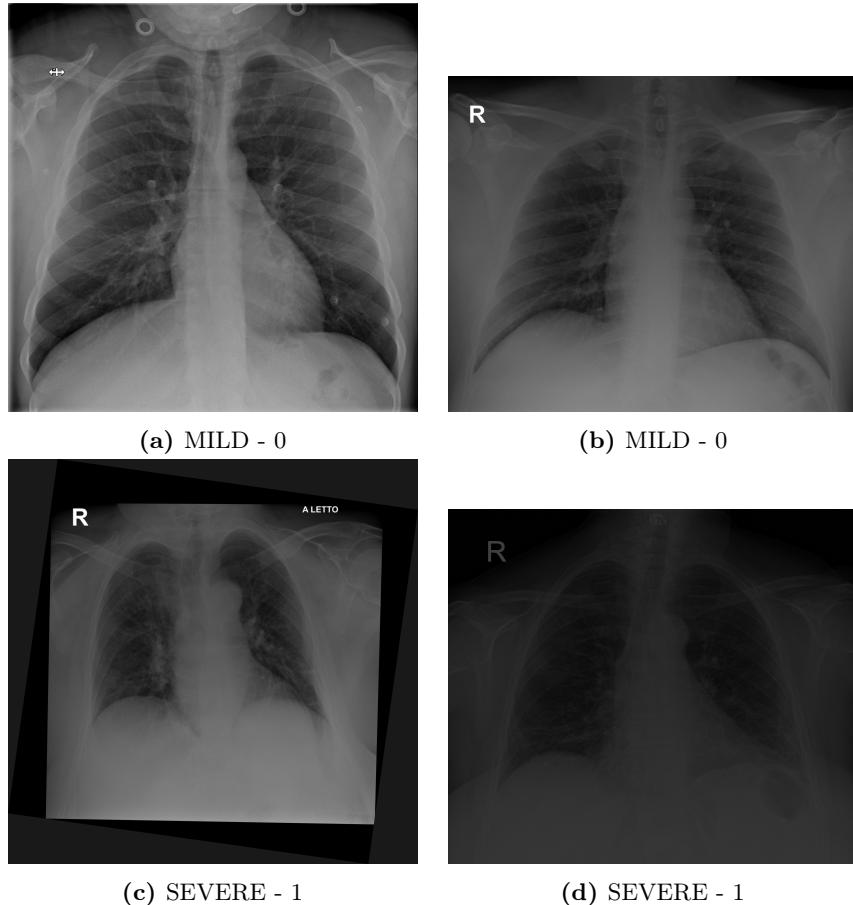
La metodologia risolutiva usata è stata ispirata dal lavoro fatto da uno dei gruppi partecipanti all'hackaton, ovvero COSBI [4].

Dunque il dataset usato è formato da immagini raffiguranti radiografie dei polmoni, le quali tuttavia non sono omogenee. Per tale motivo si è dovuto effettuare un passaggio preliminare in modo da rendere le immagini tutte della stessa dimensione, ma non distorcendole.

Oltre alle immagini è presente anche un insieme di metadati riguardanti lo stato in cui il paziente è entrato all'ospedale e l'anamnesi dello stesso. Tali metadati descrivono varie informazioni relative al paziente e, per via di ciò, sono rappresentate anche in modo diverso: esistono dati di tipo categòrico, sia con più categorie che con due sole categorie, dati che possono essere considerati come booleani e dati interi, come ad esempio l'età. (vedi 20 per l'elenco completo)

## 2.2 Immagini delle radiografie dei pazienti

Dando uno sguardo alla composizione del dataset si può notare come le immagini risultano avere dimensione diversa.



**Figura 2.1:** Campione del dataset per evidenziare le differenze tra le immagini, con relativa label

Oltre alla differenza nelle dimensioni delle varie immagini si possono notare diversità relative a luminosità, contrasto ed alcune sono in negativo. Per tali motivi risulta utile eseguire un preprocessing delle immagini, in cui si effettuano vari passaggi:

- Regolare contrasto, luminosità ed effettuare un check sulle immagini in negativo,
- Riconoscere i polmoni e creare una maschera,
- Ricavare una bounding-box che contenga i polmoni,

- Ritagliare la bounding-box contenente i polmoni, in modo che abbiano tutte la stessa dimensione.

È importante sottolineare che nell'eseguire tali passaggi si necessita di evitare di distorcere l'immagine, per tale motivo si faranno considerazioni sull'aspect ratio dell'immagine.

Per prima cosa si sfrutta la libreria OpenCV [5] al fine di normalizzare il contrasto dell'immagine mediante la funzione convertScaleAbs() ed equalizzare l'istogramma, mediante l'uso di CLAHE (Contrast Limited Adaptive Histogram Equalization) in modo anche da correggere la luminosità. convertScaleAbs() è una funzione che svolge varie operazioni in sequenza:

- effettua un rescaling basato sul fattore passato come argomento, nel caso in questione 1 (non effettuiamo rescaling),
- aggiunge un offset passato come secondo parametro, nel caso considerato tale offset è impostato a 0,
- calcola il valore assoluto della somma ottenuta dal passaggio precedente.

L'uso del CLAHE è dovuto al fatto che immagini di buona qualità hanno pixel in tutte le regioni dell'immagine, per tale motivo si necessita di allungare l'istogramma dell'immagine alle estremità. Tale operazione è detta equalizzazione dell'istogramma. In alcune immagini, come quelle più luminose, i pixel sono invece confinati in una sola estremità dell'istogramma. L'istogramma di un'immagine rappresenta la distribuzione dei colori all'interno dell'immagine. Quello che si vuole dunque ottenere è un istogramma più piatto possibile, ovvero far sì che tutti i colori abbiano la stessa distribuzione nell'immagine. Tale operazione ha come conseguenza il miglioramento del contrasto dell'immagine, per questo risulta essere utile.

Si è notato inoltre che alcune immagini del dataset risultavano essere in negativo, ovvero erano più luminose sullo sfondo e meno luminose al centro (dove si trovano i polmoni). Per affrontare tale problematica si è deciso di prendere un campione di pixel alle estremità dell'immagine e confrontarlo con uno preso al centro. Tale confronto viene effettuato calcolando una soglia, facendo la media dei colori dei pixel dell'immagine. Se il campione supera tale soglia indica che il campione è più luminoso. Poi si calcola la percentuale di bianco in entrambi i campioni. Se la percentuale di bianco presente nel campione relativo agli angoli risulta essere maggiore di quella presente nel campione preso dal centro dell'immagine, allora l'immagine è in negativo e per tale motivo si riporta l'immagine allo stato originale sottraendo i valori dei pixel attuali ad un'immagine completamente nera. In tal

modo si è riportata l'immagine "in positivo".

Tali passaggi serviranno per ottenere immagini omogenee, in grado di consentire alla rete di identificare i polmoni.

### 2.3 Segmentazione immagini mediante U-Net

Al fine di effettuare il riconoscimento dei polmoni all'interno dell'immagine, si sfrutta una U-Net. Una U-Net è una rete convoluzionale sviluppata per eseguire la segmentazione dell'immagine. Il processo di segmentazione consiste nell'assegnare ad ogni gruppo di pixel selezionato una categoria. Ad esempio si potrebbe avere un insieme di pixel dell'immagine che identificano il contorno dei polmoni.

Lo scopo della U-Net è dunque quello di prendere in input l'immagine e creare una maschera che raffiguri i polmoni. La maschera è un'immagine formata da pixel con valore in  $[0,1]$  e mira a creare un'immagine raffigurante la distinzione tra cosa rappresenta i polmoni e cosa rappresenta lo sfondo. L'utilità della maschera è quella di consentirci di trovare con maggior facilità la bounding box.

La rete usata per la segmentazione è EfficientNet. Tale rete prende un input di dimensione predefinita e restituisce un'immagine della stessa dimensione, rappresentante la maschera. Nel caso considerato, inoltre si è deciso di usare come funzione di attivazione Sigmoid e di sfruttare i pesi preallenati su ImageNet, per avere maggior precisione.

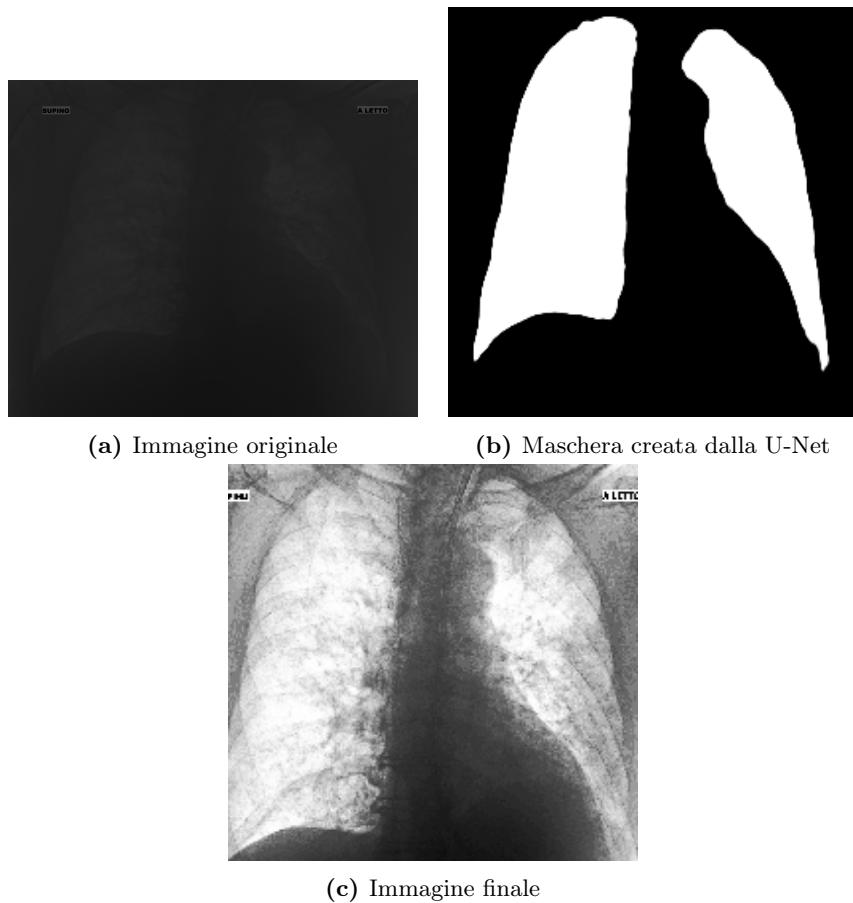
Essendo le immagini di dimensione diversa, prima di darle in input alla rete viene effettuato un ridimensionamento alle dimensioni accettate dalla rete. Tale passaggio viene effettuato mantenendo l'aspect ratio dell'immagine e inserendo padding nero per evitare distorsione. L'aspect ratio rappresenta il rapporto tra le dimensioni dell'immagine, per evitare distorsione della stessa, si deve conservare tale rapporto. Nel caso in cui l'immagine non soddisfi i requisiti delle nuove dimensioni si procede con l'aggiunta di padding di colore nero, ovvero si inseriscono i pixel necessari all'immagine per raggiungere la dimensione necessaria usando un colore non influente.

La rete performa molto bene con immagini di dimensioni piccole, tuttavia si è potuto notare, a seguito di esperimenti, che nel caso considerato, la rete performa meglio con immagini, in input, di dimensione 256 x 256 rispetto a quelle di dimensione 224 x 224. Per tale motivo, a seguito della segmentazione dovrà essere effettuato un ridimensionamento dell'immagine.

## 2.4 Selezione delle sezioni di interesse nelle immagini

Ottenute così le maschere retive alle immagini, queste sono alla base della costruzione delle bounding-box. La bounding-box rappresenta il riquadro di dimensione minima, contenente i polmoni. Per trovare tale riquadro si usano i punti estremi delle maschere: si cercano il punto più alto, più basso, il più a destra e l'estremo a sinistra. Tali punti rappresentano le coordinate di una box, la quale conterrà per certo i polmoni.

Avendo ora le bounding box, l'ultima cosa da fare è ritagliare le dalle immagini originali le bounding-box e ridimensionarle a 224 x 224. È necessario in questi passaggi di ridimensionamento mantenere presente che l'aspect-ratio deve essere conservato, per cui si effettua nuovamente padding ove necessario. Alla fine di tale procedimento si sono ottenute delle immagini raffiguranti i polmoni e della dimensione desiderata.



**Figura 2.2:** Effetti dei vari passaggi sulle immagini del dataset

## 2.5 Metadati relativi alle radiografie dei pazienti

Per quanto riguarda i metadati la situazione è meno complessa, poiché gestita mediante file csv. Il dataset in questo caso è sempre diviso in training e test set, ma è composto da un insieme di righe e colonne: le righe rappresentano i vari pazienti, mentre le colonne rappresentano i dati relativi al paziente.

Le colonne iniziali di tali dataset sono le stesse e sono: ospedale, età, sesso, positività all'ammssione, temperatura corporea, giorni di febbre, tosse, difficoltà respiratorie, WBC, RBC, CRP, fibrina, glucosio, PCT, LDH, INR, D\_dimer, percentuale di ossigeno, PaO<sub>2</sub>, SaO<sub>2</sub>, PaCO<sub>2</sub>, pH, malattie cardiovascolari, cardiopatia ischemica, fibrillazione atriale, insufficienza respiratoria, arresto cardiaco, ictus, alta pressione sanguigna, diabete, demenza, BPCO, cancro, malattia renale cronica, insufficienza respiratoria, obesità, posizione.

Questi rappresentano i dati. Oltre a tali colonne sono presenti: una colonna che rappresenta un identificativo del paziente, una colonna col nome del file contenente la radiografia dei polmoni e una colonna contenente la prognosi (vuota nel caso del test set).

Estratto del dataset di test									
ImageFile	H.	Age	Sex	.....	T.C.	D.F.	Diab.	Prognosis	
...	...	...	...	...	...	...	...	...	...
P_3_428.png	F	78	1	...	35.8	1	0	nan	
P_3_158.png	F	65	1	...	36.6	3	0	nan	
P_3_429.png	F	45	0	...	38	4	0	nan	
P_3_299.png	F	80	0	...	38.4	3	0	nan	
P_3_483.png	F	63	0	...	36	1	1	nan	
...	...	...	...	...	...	...	...	...	...

Ciò che si vuole ottenere, al fine di usare tali metadati insieme alle immagini per effettuare il training, è un insieme di dati che sia completamente pieno, ovvero presenti corrispondenze per tutte le righe e colonne del dataset.

Dunque si cerca di ridurre il dataset di partenza nel più grande sottinsieme che contenga tutte le corrispondenze. Tale omogeneità deve essere presente sia nel dataset di training che in quello di test. Oltre a ciò, alla fine, si cercheranno le colonne comuni ai due dataset, scartando le altre. In tal modo si possono unire i due dataset così da formarne uno solo.

Per iniziare si eliminano le colonne che non presentano alcun dato, o che presentano dati solo per un numero molto ristretto di paziente, per :

- nel dataset di training sono state scartate le colonne: giorni di febbre, fibrina, glucosio, PCT, LDH, INR, D\_dimer, percentuale di ossigeno, PaO2, SaO2, PaCO2, pH, cardiopatia ischemica, insufficienza respiratoria, obesità, positività all'ammssione, temperatura corporea, RBC, CRP.
- nel dataset di test sono state scartate le colonne: giorni febbre, fibrina, LDH, PaO2, PaCo2, pH, PCT, INR, D\_dimer, SaO2, tempertaure corporea, CRP.

Eliminate le colonne in eccesso, si procede con l'eliminazione dei paziente che non presentano tutti i dati. A seguito di tale operazione le righe, ovvero i pazienti, che sono presenti nel dataset di train sono 946, mentre quelle del dataset di test 472. Le colonne in comune tra i due dataset sono:

- Ospedale,
- Età,
- Sesso,

- Tosse,
  - Difficoltà respiratorie
  - Numero di cellule bianche,
  - Pressione sanguigna alta,
  - Diabete,
  - Demenza,
  - BPCO (Broncopneumopatia cronica ostruttiva),
  - Cancro,
  - Malattia renale cronica

In tal modo si è ottenuto un dataset che contiene 12 colonne significativa (sopra elencate) contenenti dati per 1418 pazienti.

# 3 | Risultati training con sole immagini

## 3.1 Creazione rete neurale

Lo sviluppo della rete neurale convoluzionale, per effettuare previsioni partendo dalle immagini, è basata sulla rete MobileNetV2. L'uso di tale rete come base consente di usare dei pesi preallenati in modo da rendere il processo di training sul dataset più efficiente.

Tuttavia tale processo necessita di alcune fasi aggiuntive per fare in modo che la rete riesca a prendere i dati della dimensione esatta e restituire delle previsioni nell'insieme desiderato.

Tali operazioni sono:

- Inserire la dimensione delle immagini nel layer di input
- Effettuare fine tuning e transfer learning per poter usare i pesi preallenati
- Inserire dei layer finali per ottenere degli output significativi

La dimensione scelta delle immagini è (224x224), dunque la rete prende in ingresso degli array di dimensione 224x224x3 (dove quest'ultimo indica l'uso dei colori RGB). Per tale motivo il layer di input deve accettare tale dimensione.

I pesi preallenati che si sono scelti sono pesi allenati su ImageNet. ImageNet è un dataset di immagini suddivise in 1000 classi. Il dataset su cui si deve svolgere il training, tuttavia, possiede unicamente una classe, per via del fatto che abbiamo trasformato il valore della prognosi da una stringa ad un valore binario. Per cui l'immagine può appartenere o meno a tale classe.

Al fine di poter usare tali pesi per effettuare il training, si necessita dunque di ulteriori passaggi:

- Transfer Learning
- Fine tuning

## 3.2 Transfer Learning

Il transfer learning consente di sfruttare la rete già allenata a risolvere problemi diversi, ma comunque correlati con quello di interesse. Nel caso considerato tale tecnica permette di usare una rete allenata per prevedere l'appartenenza di una immagine ad una delle 1000 classi di ImageNet per creare una rete in grado di classificare le immagini del dataset in una unica classe.

Per sfruttare la rete allenata, congeliamo lo stato dei layer di classificazione, al fine di non alterarli, e settiamo la rete come non allenabile. In tal modo si è ottenuto un nuovo modello basato sulla MobileNetV2.

Ora si presenta un problema relativo alla classificazione. Per ovviare al fatto che tale operazione sarà fatta per una sola classe, si necessita dell'inserimento di altri layer alla fine del modello precedentemente ottenuto.

Tali layers sono:

- un GlobalAveragePooling2D()
- due Dense()

Il primo layer serve per via del fatto che allo stato attuale la rete produce un output multidimensionale e, per ottenere previsioni formate da un singolo vettore della dimensione prevista, usiamo tale layer, il quale genera previsioni basate sul blocco multidimensionale e facendone una media.

Il primo Dense layer viene usato per generare l'output prodotto dalle immagini dei polmoni. Tale layer è formato da 100 neuroni ed ha come funzione di attivazione ReLu.

Il secondo Dense layer svolge il lavoro di classificazione vero e proprio. Per via del fatto che si è scelto di usare una label binaria, ovvero classifichiamo su un'unica classe, tale layer è composto da un solo neurone, ed ha come funzione di attivazione Sigmoid.

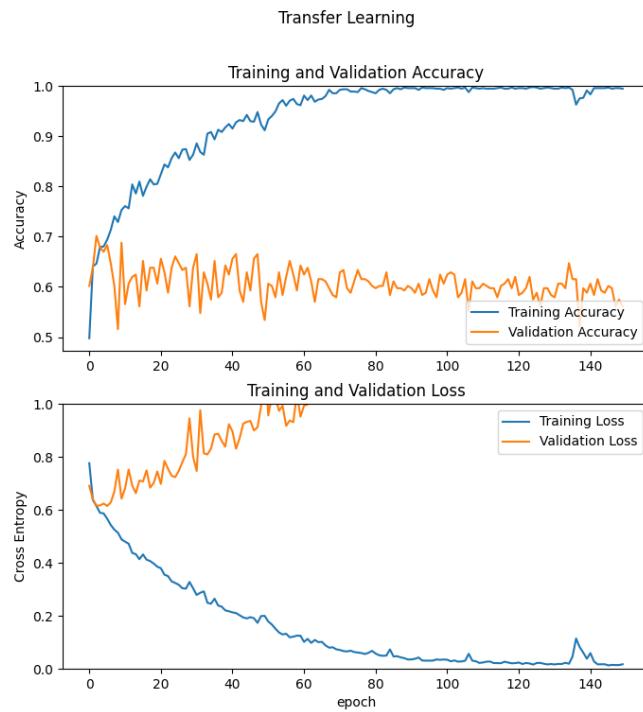
Una volta creati i layers, per completare la fase di transfer learning, si deve compilare il modello finale, ottenuto aggiungendo i nuovi layers.

Ora la rete è pronta per una prima fase di training. In tale fase sono stati usati i seguenti parametri per il training:

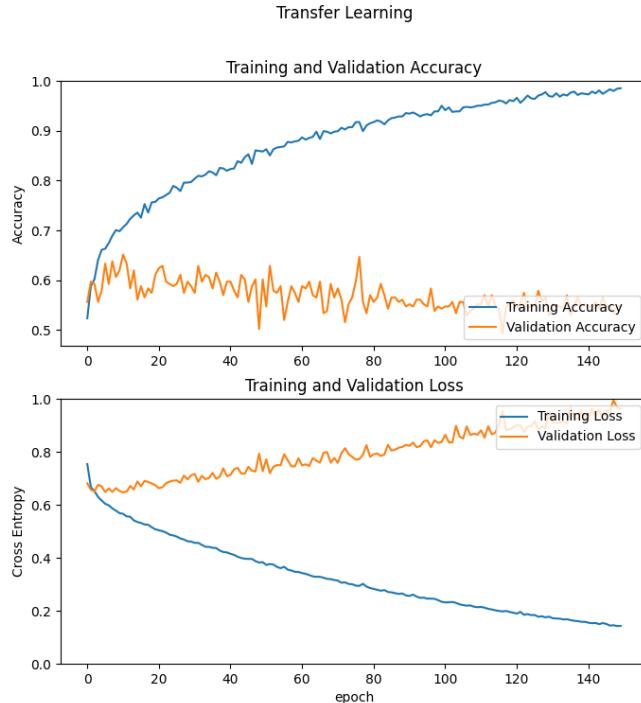
- Loss function: Adam
- Metrica: accuracy
- Epoche 150
- Batch size: 30

### 3.3 Risultati dei test effettuati per il transfer learning

Per quanto concerne il valore del learning rate sono si è deciso di effettuare prove sia con  $10^{-3}$  che con  $10^{-4}$ . I risultati ottenuti dal training sono espressi nei seguenti grafici.



**Figura 3.1:** Test effettuato usando come learning rate  $10^{-3}$



**Figura 3.2:** Test effettuato usando come learning rate  $10^{-4}$

Dai grafici si può notare come il training effettuato con learning rate pari a  $10^{-3}$  risulta essere più preciso, anche se non di molto. Possiamo inoltre notare come la training loss diminuisce suggerendo il corretto funzionamento della rete.

### 3.4 Fine Tuning

Per procedere con la fase di fine tuning, si deve rendere nuovamente allenabile il modello creato, in modo tale che i pesi possano essere allenati considerando i nuovi layers. Così facendo si permette ai pesi di regolarsi sul dataset d'interesse, partendo da quello su cui sono stati inizialmente allenati.

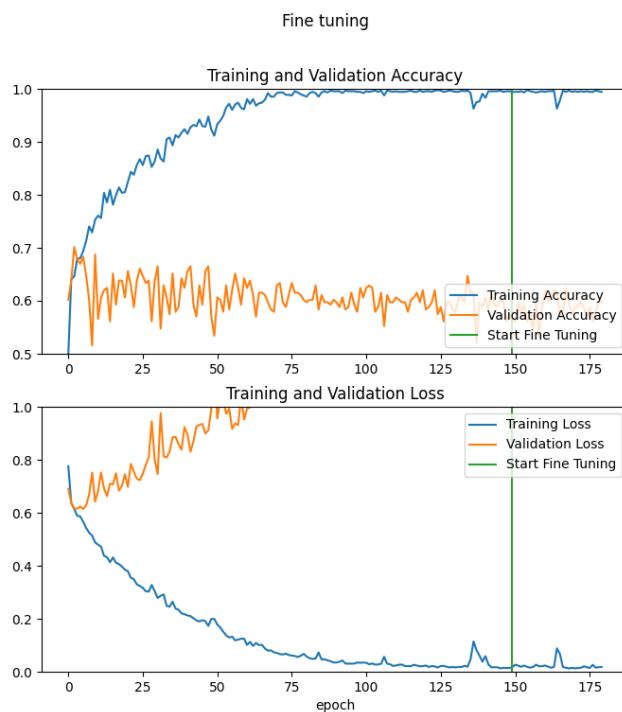
Essendo gli ultimi layers di una rete inutili in termini di classificazione, possiamo decidere di congelarli, ovvero non considerarli durante il training, in modo da risparmiare risorse. Effettuato tale passaggio si può procedere con la compilazione del modello ottenuto e procedere con il training.

I parametri relativi al training in questa fase sono:

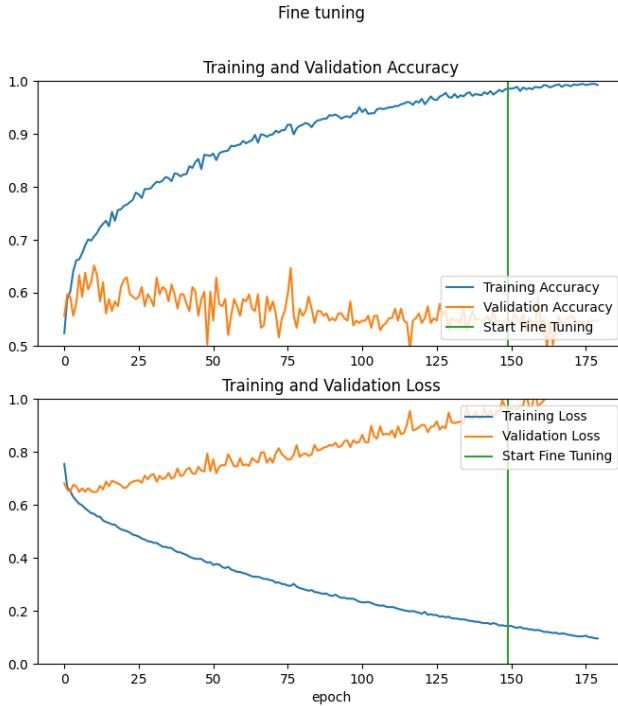
- Loss function: Adam
- Metrica: accuracy
- Epoche 30
- Batch size: 30

### 3.5 Risultati dei test effettuati per il fine tuning

Di seguito sono riportati i risultati ottenuti a seguito del transfer learning e fine tuning, sempre considerando i due valori scelti per il learning rate.



**Figura 3.3:** Test effettuato usando come learning rate  $10^{-3}$



**Figura 3.4:** Test effettuato usando come learning rate  $10^{-4}$

Anche dal grafico relativo al fine tuning si una validation accuracy maggiore usando un learning rate pari a  $10^{-3}$ . Importante è notare la linea verde che demarca l'inizio del fine tuning e la fine del transfer learning.

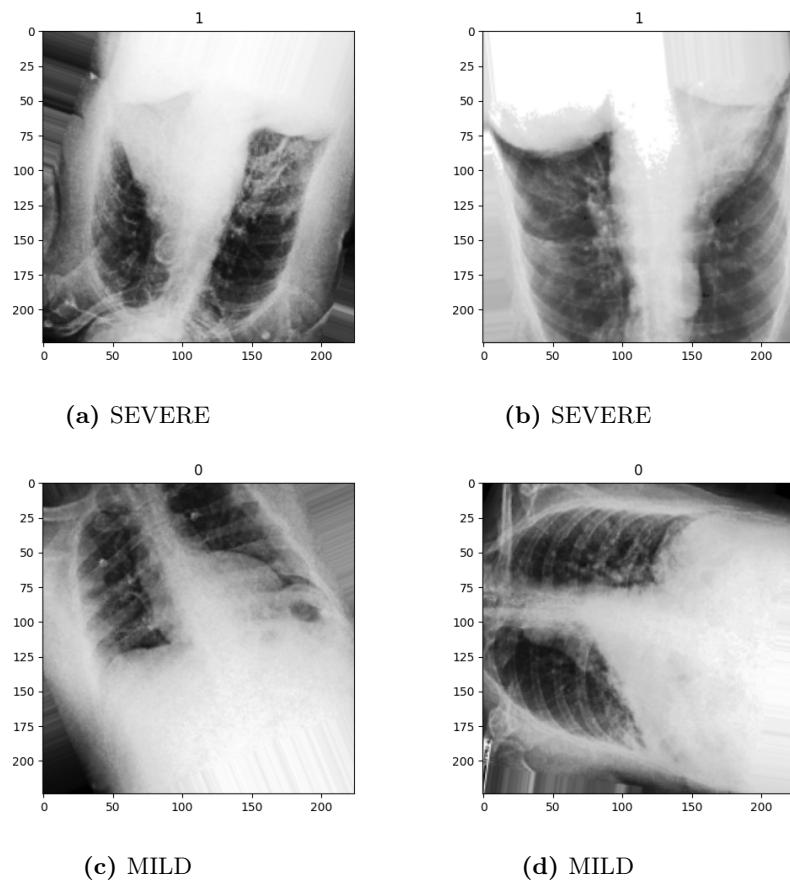
### 3.6 Overfitting e Data Augmentation

Dai risultati ottenuti risulta evidente la presenza di overfitting nella rete. Tale problema si è potuto notare per via del fatto che, anche a seguito del fine tuning, la validation accuracy tende a diminuire.

L'overfit può essere causato dal fatto che la rete si abitui agli input del training set, o al fatto che questi siano molto simili tra di loro e risponda in maniera casuale a nuovi dati. Per tale motivo si è scelto di usare la data augmentation come tecnica risolutiva. La data augmentation consiste nel prendere le immagini del dataset, effettuare delle modifiche ad esse (come rotazioni), per creare una immagine nuova, in modo da introdurre diversità all'interno del dataset.

Per implementare tale tecnica si è optato per l'uso del `ImageDataGenerator()`. L'adozione di tale funzionalità presente in TensorFlow è dovuta al fatto che consente di creare immagini partendo dal dataset iniziale, applicando trasformazioni casualmente, scegliendo tra quelle selezionate.

Le trasformazioni che sono state scelte, in base alla possibilità che la rete consideri anche le nuove immagini come valide, sono la rotazione verticale ed orizontale, specificando anche il range di angolo in cui effettuare tale rotazione. La scelta di queste trasformazioni è stata validata anche osservando le immagini prodotte e riflettendo sul fatto che fossero sensate come input della rete. È importante sottolineare che l'augmentation viene effettuata

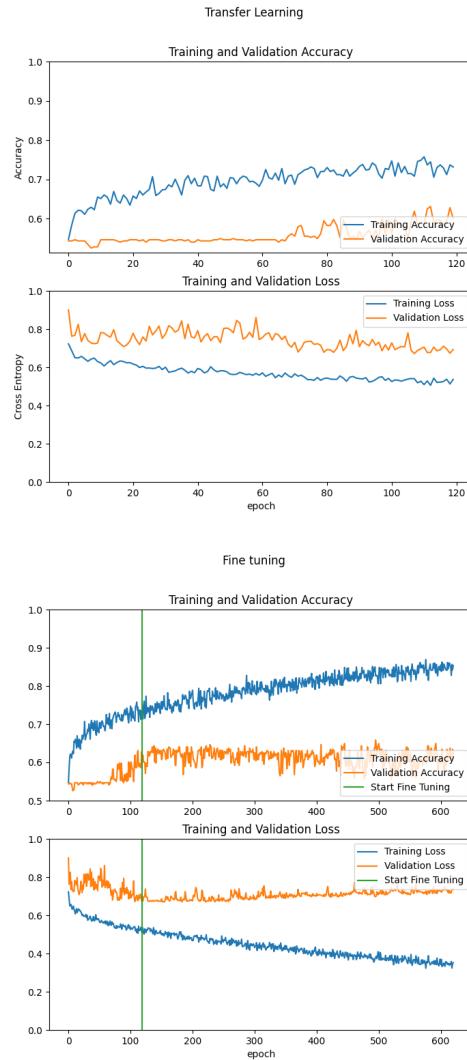


**Figura 3.5:** Immagini a seguito dell'augmentation, con relativa label

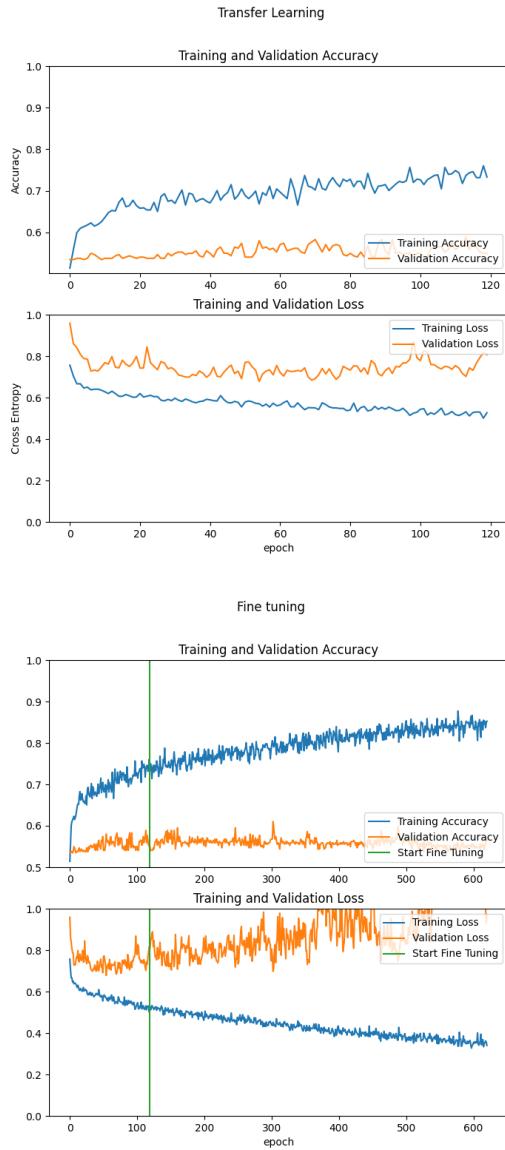
unicamente sul training set.

### 3.7 Risultati dei test effettuati per la data augmentation

Per verificare se la data augmentation ha avuto effetto si procede con dei training della rete. Tali training sono stati effettuati mantenendo gli stessi parametri precedentemente usati.



**Figura 3.6:** Training effettuato con learning rate pari a  $10^{-3}$  a seguito dell'augmentation



**Figura 3.7:** Training effettuato con learning rate pari a  $10^{-4}$  a seguito dell'augmentation

Nei grafici precedenti si riporta sia la versione del solo transfer learning che del test comprendente anche il fine tuning. Nelle seconde figure, per tale motivo si mostra quando inizia il fine tuning (linea verde). Anche in questo caso la diminuzione della training loss suggerisce il corretto funzionamento nell’allenamento della rete. Per quanto concerne i risultati ottenuti sono stati riscontrati lievi miglioramenti relativi alla validation accuracy delle reti, sempre osservando migliori prestazioni da parte della rete con learning rate pari a  $10^{-3}$ . Si può infine notare come l’uso della data augmentation ha avuto l’effetto sperato, osservando che la validation accuracy dei grafici sopra mostrati non scende.

# 4 | Risultati training rete neurale con dati eterogenei

## 4.1 Selezione di un sottinsieme completo del dataset

I risultati ottenuti con la CNN possono essere migliorati o resi più attendibili con l'utilizzo di altre tipologie di dati. Questi sono le informazioni relative al paziente al momento dell'arrivo in ospedale.

Tali dati sono di vario tipo, quelli usati dal dataset considerato sono:

- Categorico, esprimono l'appartenenza ad una o più categorie
- Booleano, esprimono se il paziente è affetto da una certa patologia o presenta alcuni
- Numerici

In questa fase è importante che tutti i dati siano presenti per ogni paziente. Per tale motivo si è dovuto trovare un sottinsieme del dataset che presenta il maggior numero di dati. Per fare ciò sono state eliminate le colonne che non avevano corrispondenze tra i pazienti, ovvero erano vuote oppure presentavano dati solo per pochi pazienti. Al fine di ottenere un risultato migliore sono state eliminati anche i pazienti che non presentavano dati per la maggior parte delle colonne del dataset.

Effettuata tale operazione, si è proceduto col creare un unico dataset ottenuto dall'unione di training set e test set. Per fare ciò le colonne presenti tra i due devono essere le stesse. Per tale motivo si è giunti ad un dataset formato dalle seguenti colonne:

- Ospedale, dato categorico che rappresenta l'ospedale che ha accolto il paziente tra A, B, C, D, E e F
- Età, dato numerico

- Sesso, categorico binario, ovvero maschio o femmina
- Tosse, binario
- Difficoltà respiratorie
- Numero di cellule bianche, dato numerico che indica la percentuale di globuli bianchi nel sangue
- Pressione sanguigna alta, binario
- Diabete, binario
- Demenza, binario
- BPCO (Broncopneumopatia cronica ostruttiva), binario
- Cancro, binario
- Malattia renale cronica, binario

Tali dati sono presenti per 946 pazienti del training set e 472 del test set, per cui abbiamo un dataset di 1218 pazienti. Questo dataset sarà poi suddiviso in training, validation e test set, per effettuare l'allenamento della rete e per verificare la capacità di effettuare previsioni.

Estratto del dataset dato dall'unione dei due di partenza									
ImageFile	H.	Age	....	Cough	WBC	Ic.	H.B.P.	Prognosis	
...	...	...	...	...	...	...	...	...	...
P_281.png	E	68	...	0	7.33	0	1	MILD	
P_544.png	F	72	...	0	9.6	0	1	SEVERE	
P_657.png	C	83	...	1	9	0	1	SEVERE	
P_1_93.png	F	66	...	0	10	0	0	SEVERE	
P_73.png	A	48	...	1	9.13	0	0	MILD	
...	...	...	...	...	...	...	...	...	...

Ora che si presenta il dataset completo, si procede col trasformare tutti i dati nello stesso formato, ovvero in valori binari [6]. Partendo dai dati categorici, si procede usando la codifica one-hot. Tale procedura prevede che le categorie relative al dato vengano trasformate in una rappresentazione binaria, in cui ogni categoria è rappresentata da una serie di zeri ed un unico uno presente nella categoria codificata. Per quanto riguarda il sesso, essendo una categoria binaria, si può usare un valore binario, per cui un sesso sarà

rappresentato da 0 e l'altro da 1. L'ospedale è un dato che prevede sei categorie, per cui queste saranno rappresentate da sei bit. Ogni rappresentazione presenterà cinque zeri ed un unico uno (es. 001000).

A livello pratico tali trasformazioni sono state implementate usando la libreria scikit-learn, in particolare le funzioni MultiLabelBinarizer(), per l'ospedale, e LabelBinarizer() per il sesso. In tal modo si riesce a trasformare le categorie in array di bit, ovvero valori compresi in [0,1].

Per gestire i valori numerici si sfrutta un'altra funzione di scikit-learn, ovvero MinMaxScaler(). Tale funzione scala i valori dati in input in un range specificato, nel caso in considerazione [0,1]. La trasformazione avviene mediante:

$$X_{std} = \frac{(X - X.\min(axis=0))}{(X.\max(axis=0) - X.\min(axis=0))}$$

$$X_{scaled} = X_{std} * (max - min) + min$$

I valori binari infine sono rimasti invariati, poiché esprimono il valore nel range desiderato.

## 4.2 Struttura rete neurale

Avendo ora convertito i dati del dataset in modo da essere in [0,1], si può iniziare a costruire la rete che deve fare previsioni prendendo come input tali dati. È importante notare che una volta effettuata la codifica one-hot della colonna relativa all'ospedale, si ottiene un vettore di dimensione 6 (una per ogni categoria) al posto dell'unica colonna che rappresenta il dato. Per tale motivo la dimensione dell'input della rete non è più 12 (in base al numero di colonne presenti inizialmente), ma 17.

La rete dunque sarà formata da due layer:

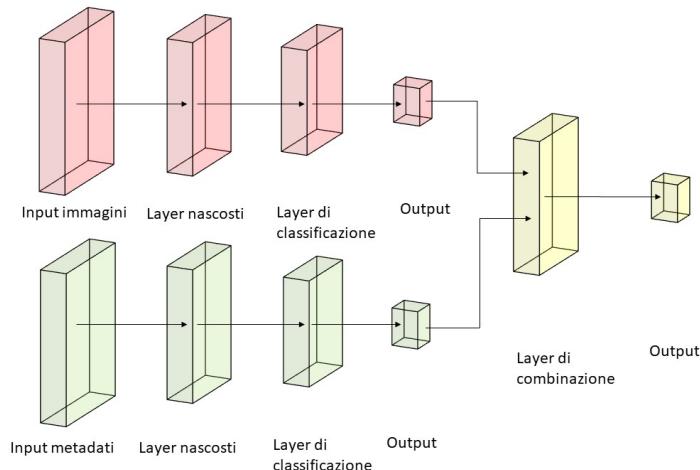
- un Dense() layer formato da 30 neuroni
- un Dense() layer formato da 5 neuroni

Il primo layer è colui che si occupa di ricevere anche gli input, per cui la dimensione degli input dovrà essere pari a 17. Il secondo layer, invece, è formato da soli 5 layer e si occupa di effettuare una prima classificazione. Entrambi i layer presentano come funzione di attivazione ReLu. Con tale rete, si riesce dunque ad effettuare il training relativo all'uso dei metadati.

Per poter usare contemporaneamente le due reti neurali create, ovvero la CNN e il MLP, si necessita di un ulteriore passaggio. Lo scopo di tale passaggio è avere una rete che accetti come input sia immagini che i metadati relativi, al fine di ottenere una previsione più affidabile, poiché basata sull'uso di più dati.

Per gestire gli output delle due reti questi vengono concatenati, in modo da ottenere un unico array di output. Al fine di effettuare le predizioni su tale output si aggiunge, alla fine della rete, un ulteriore Dense layer, formato da un unico neurone, con funzione di attivazione Sigmoid e che prende un input della dimensione della combinazione degli output delle due reti.

Si può, infine, creare la rete che prende in input la combinazione degli input della rete, mentre l'output sarà determinato dal nuovo layer inserito.

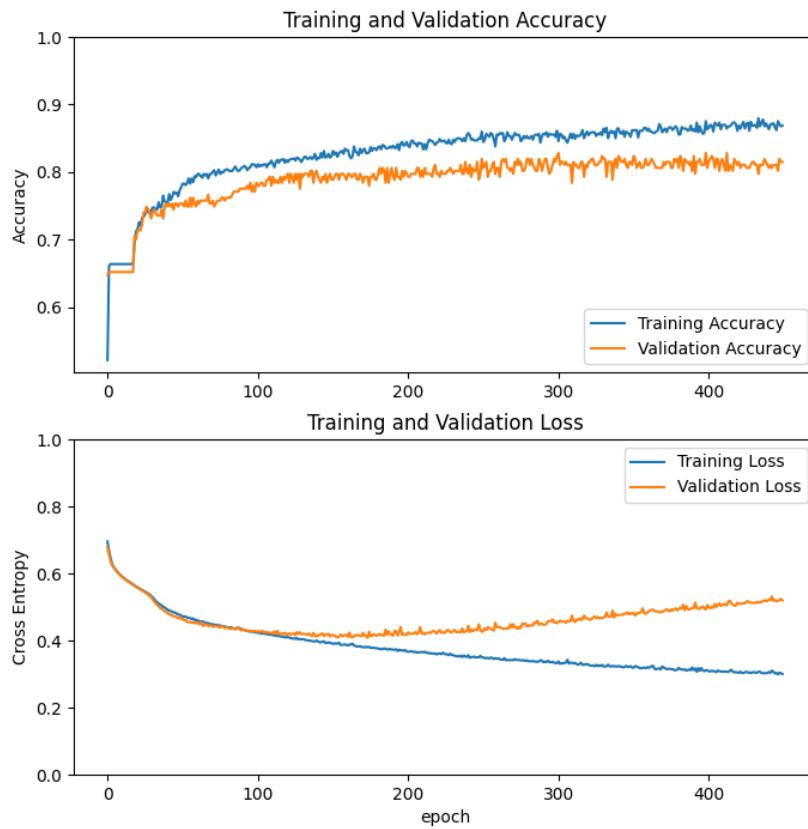


**Figura 4.1:** Rappresentazione della rete finale composta da CNN e MLP

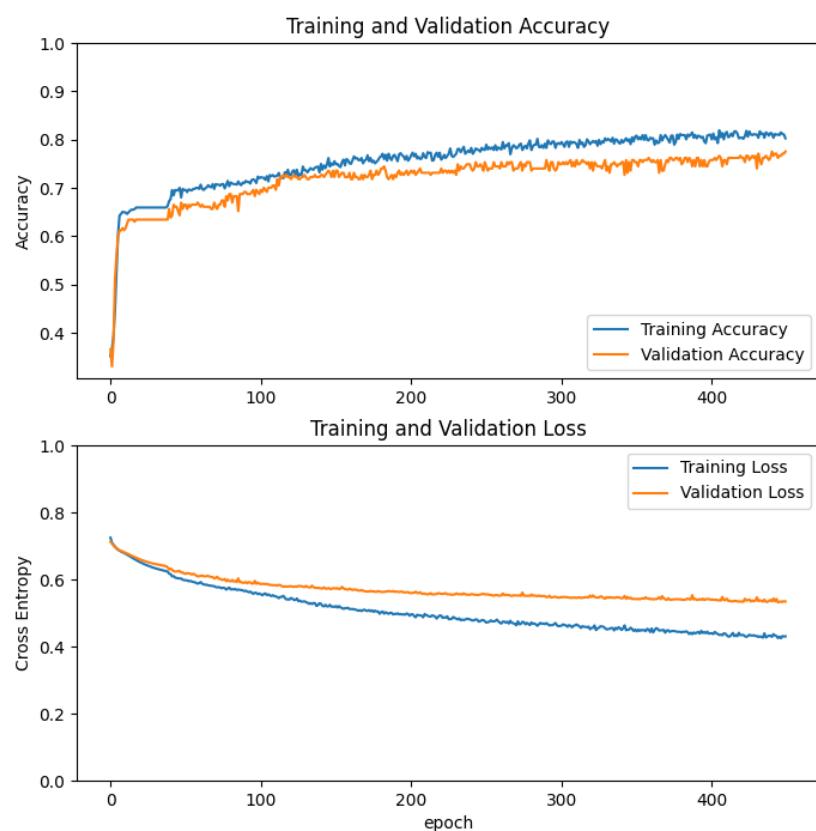
Nella figura precedente sono riportate schematicamente la CNN (in rosso) e la MLP (in verde), le quali, una volta combinate, creano la rete finale. L'output è dato dunque dalla composizione degli output. I primi layer rappresentano la gestione degli input da parte delle rispettive reti, mentre i layer nascosti servono per svolgere varie funzionalità al fine di arrivare nei layer di classificazione avendo i dati delle dimensioni opportune. Infine vengono restituiti gli output che saranno combinati per produrre la previsione finale.

### 4.3 Risultati dei test effettuati per la rete composta

Ora si può dunque procedere con la compilazione della rete e con il training. I training sono stati effettuati usando i parametri usati nel caso della rete convoluzionale.



**Figura 4.2:** Test effettuato usando come learning rate  $10^{-3}$



**Figura 4.3:** Test effettuato usando come learning rate  $10^{-4}$

Dai precedenti grafici si nota il corretto funzionamento suggerito dall'andamento decrescente della training loss. Oltre a ciò è possibile notare un importante aumento della validatino accuracy rispetto a ciò che si era ottenuto nel caso di training con sole immagini. Infine si può notare come il learning rate pari a  $10^{-3}$  sia anche in questo caso più performante.

#### 4.4 Confronto tra le varie reti adoperate nel progetto di tesi

Confronto dei risultati ottenuti con le varie reti usate

Nome	accuracy	epoche	l.r.	descrizione
CNN t.l. (pagina 25)	55%	150	$10^{-3}$	rete neurale convoluzionale transfer learning con learning rate pari a $10^{-3}$
CNN t.l. (pagina 26)	53%	150	$10^{-4}$	rete neurale convoluzionale transfer learning con learning rate pari a $10^{-4}$
CNN f.t. (pagina 27)	60%	150+30	$10^{-3}$	rete neurale convoluzionale fine tuning con learning rate pari a $10^{-3}$
CNN f.t. (pagina 28)	55%	150+30	$10^{-4}$	rete neurale convoluzionale fine tuning con learning rate pari a $10^{-4}$
CNN d.a. (pagina 30)	63%	150+500	$10^{-3}$	rete neurale convoluzionale data augmentation con learning rate pari a $10^{-3}$
CNN d.a. (pagina 31)	57%	150+500	$10^{-4}$	rete neurale convoluzionale data augmentation con learning rate pari a $10^{-4}$
Composta (pagina 37)	82%	450	$10^{-3}$	rete neurale convoluzionale concatenato al multilayer perceptron con learning rate pari a $10^{-3}$
Composta (pagina 38)	79%	450	$10^{-4}$	rete neurale convoluzionale concatenato al multilayer perceptron con learning rate pari a $10^{-3}$

La tabella sopra riportata mostra un confronto tra le varie reti usate, le epoche dell'allenamento, il valore del learning rate e la validation accuracy ottenuta.

Per quanto concerne le CNN con fine tuning sono state considerate anche le epoche precedenti relative al transfer learning. In maniera analoga per la rete con data augmentation sono evidenziate le epoche relative al transfer learning e quelle relative al fine tuning.

Da tale tabella si evince come, nonostante le varie tecniche usate per migliorare la precisione della CNN, l'uso di dati eterogenei comporta un netto miglioramento, dunque la combinazione di CNN e multilayer perceptron risulta essere la scelta ottimale per ottenere risultati più precisi.

# Conclusioni

In questo progetto di tesi si è focalizzata l'attenzione sull'uso dei reti neurali convoluzionali e di multilayer perceptron per analizzare ed effettuare previsioni usando un dataset fornito. Il dataset si può suddividere nelle immagini, raffiguranti radiografie del torace, e nei metadati relativi alle immagini.

In primo luogo si è compreso la composizione del dataset, si è osservato che le immagini avevano dimensioni varie e presentavano alcune differenze a livello di luminosità dell'immagine. Per tali motivi sono stati effettuati alcuni accorgimenti per migliorare la qualità delle immagini e ridimensionarle.

In tal modo si è migliorata la qualità del dataset. Per procedere con il training della rete convoluzionale, si è dovuto prima creare immagini in cui si vedono solamente i polmoni. Tale passaggio sfrutta la U-Net per individuare la collocazione dei polmoni nell'immagine.

Una volta terminato questo studio sulle immagini si è passato al training vero e proprio della rete convoluzionale. Per migliorare le prestazioni della rete sono stati applicate le tecniche di Fine Tuning e Transfer Learning. Alla fine si è proceduto con l'effettuare dei training di test e i risultati ottenuti sono stati analizzati.

Per l'implementazione dell'MLP si è deciso di usare dei layer che calcolano l'output partendo da un insieme di metadati presi dal dataset. Anche in tal caso sono stati effettuati accorgimenti sul dataset, in modo da avere un suo sottinsieme completo, poiché erano presenti dati mancanti.

In fine è stata creata un'unica rete, composta dalle precedenti in modo che possa prendere in ingresso sia immagini che metadati e restituire un risultato più accurato.

Una modifica interessante del lavoro effettuato è rendere il modello utilizzabile su dataset che presentano dati mancanti. In alternativa si potrebbero comparare le prestazioni ottenute con reti differenti.

# Bibliografia

- [1] google. «TensorFlow.» (2022), indirizzo: <https://www.tensorflow.org/>.
- [2] P. Yakubovskiy. «pesi U-Net.» (2018), indirizzo: [https://segmentation-models.readthedocs.io/en/latest/tutorial.html?highlight=encoder\\_weights#models-and-backbones](https://segmentation-models.readthedocs.io/en/latest/tutorial.html?highlight=encoder_weights#models-and-backbones).
- [3] «ai4covid.» (2022), indirizzo: <https://ai4covid-hackathon.it/>.
- [4] Valerio Guerrasi e Paolo Soda. «Covid CXR Hackathon.» (2022), indirizzo: <https://github.com/cosbidev/COVIDCXRChallenge>.
- [5] Intel corp., Itseez e Willow Garage. «opencv.» (2000), indirizzo: <https://docs.opencv.org/4.5.5/>.
- [6] A. Rosebrock, «Keras: Multiple Inputs and Mixed Data,» 2021. indirizzo: <https://pyimagesearch.com/2019/02/04/keras-multiple-inputs-and-mixed-data/>.