

ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

SCUOLA DI INGEGNERIA E ARCHITETTURA

Dipartimento di Informatica
Scienza e Ingegneria
DISI

Corso di Laurea in Ingegneria Informatica

Title

Candidato:

Gianmiriano Porrazzo

Relatore:

prof. Andrea Camisa

Correlatore:

prof. Andrea Testa

Anno Accademico
2021–2022

Sessione
II

Indice

Abstract	3
Introduzione	4
1 Strumenti usati	5
2 Dataset	6
2.1 Composizione	6
2.2 Immagini	7
2.3 U-Net	8
2.4 Bounding-box	9
2.5 Metadati	10
3 Implementazione: CNN	13
3.1 Creazione rete neurale	13
3.2 Transfer Learning	14
3.3 Fine Tuning	16
3.4 Overfit e Data Augmentation	19
4 Gestione dati eterogenei	23
4.1 Gestione dei dati	23
4.2 Struttura MLP	25
4.3 Creazione rete composta	26
5 Risultati	29
5.1 Risultati ottenuti	29
Conclusioni	30
5.2 Sviluppi Futuri	31
Bibliografia	32

Abstract

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 sul 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 modi 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.

1 | 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 presi per sfruttare tale modello. 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 per riconoscere immagini simili a quelle presenti nel dataset di riferimento.

2 | Dataset

2.1 Composizione

Il dataset usato per effettuare lo studio è stato preso da un hackaton [2] riguardante la creazione di una rete neurale in grado di apprendere da immagini relative a radiografie di pazienti affetti da covid in modo da predirre la gravità della prognosi.

La metodologia risolutiva usata è stata ispirata dal lavoro fatto da uno dei gruppi partecipanti all'hackaton, ovvero COSBI [3].

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 sono presenti anche un insieme di metadati riguardanti lo stato in cui il paziente è entrato all'ospedale e l'anamnesi dello stesso. Tali metadati presentano descrivono varie informazioni relative al paziente, per tale motivo sono rappresentate anche in modo diverso: esistono dati di tipo categorico, sia con più categorie che con due sole categorie, dati che possono essere considerati come booleani e dati interi, come ad esempio l'età.

2.2 Immagini

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

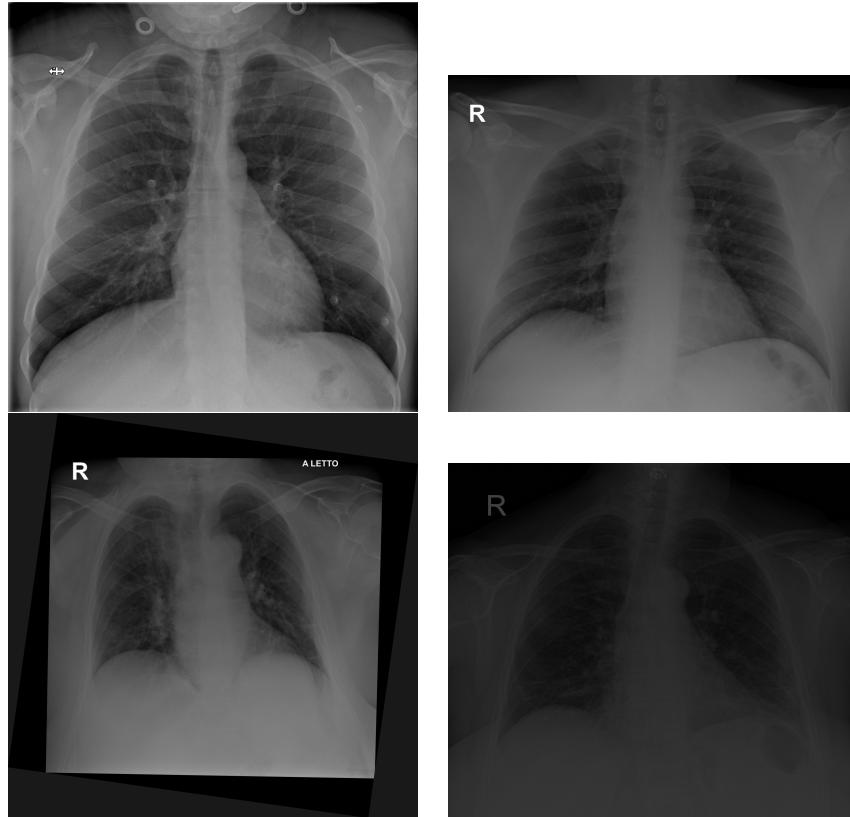


Figura 2.1: Campione del dataset per evidenziare le differenze tra le immagini

nota immagini scure

Per tale motivo risulta utile eseguire un preprocessing delle immagini, in cui si effettuano vari passaggi:

- Riconoscere i polmoni e crearne 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.

Tali passaggi serviranno per ottenere immagini omogenee, in grado di consentire alla rete di identificare i polmoni ed avranno una dimensione pari a 224 x 224

2.3 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 Bounding-box

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.

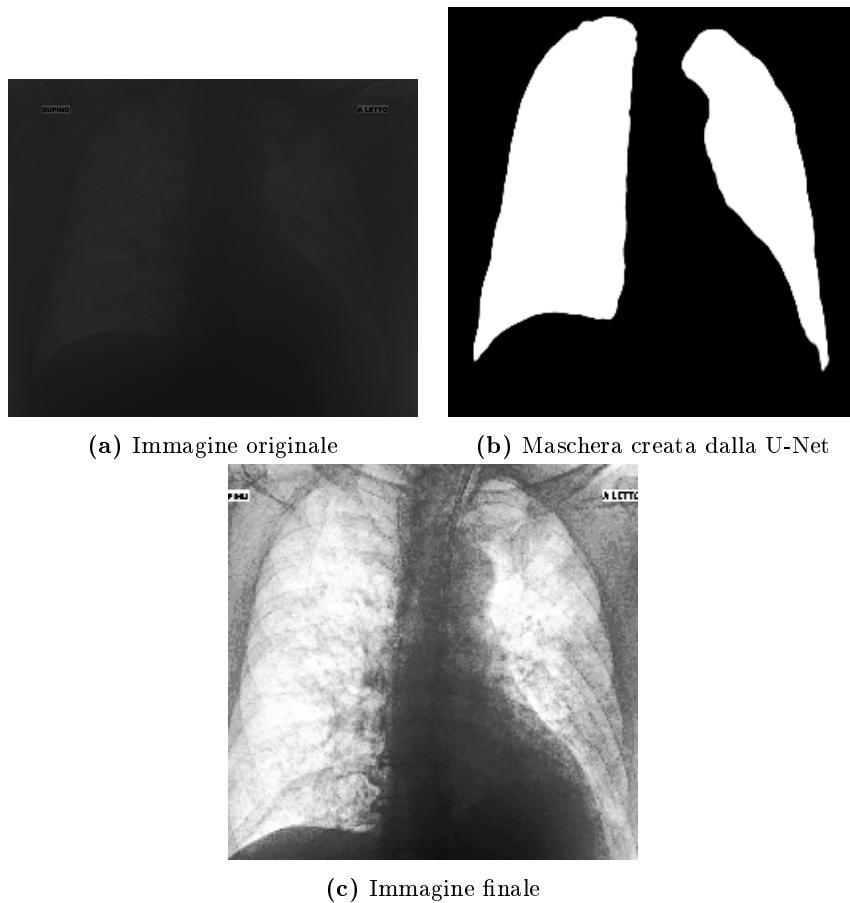


Figura 2.2: Effetti dei vari passaggi sulle immagini del dataset

2.5 Metadati

Per quanto riguarda i metadati la situazione è meno complessa, poichè 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 vari 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, PaO₂, SaO₂, PaCO₂, pH, malattie cardiovascolari, cardiopatia ischemica, fibirlazione 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	Hospital	Age	Sex	Temp_C	DaysFever	Diabetes	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 cereranno 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 | Implementazione: CNN

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 string 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

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.

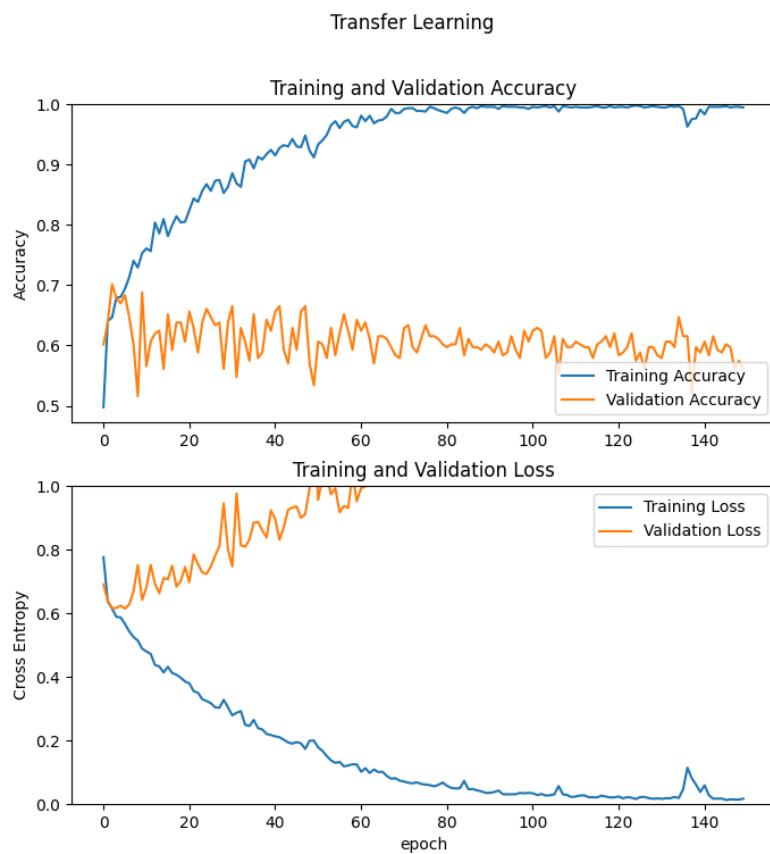


Figura 3.1: Test effettuato usando come learning rate 10^{-3}

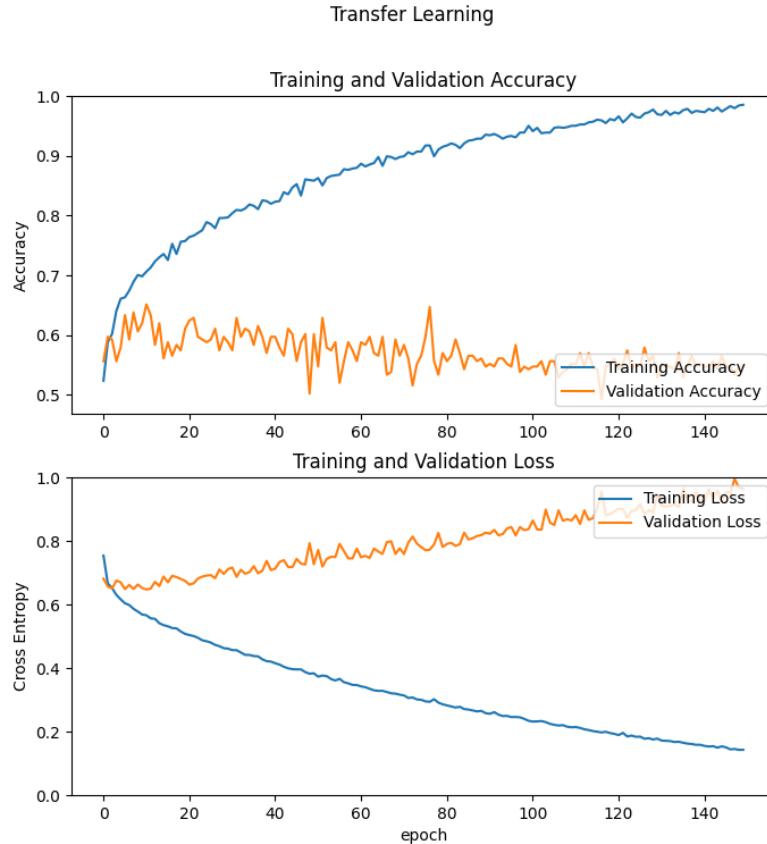


Figura 3.2: Test effettuato usando come learning rate 10^{-4}

3.3 Fine Tuning

Per procedere con la fase di fine tuning, si deve rendere nuovamente trainabile 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

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.

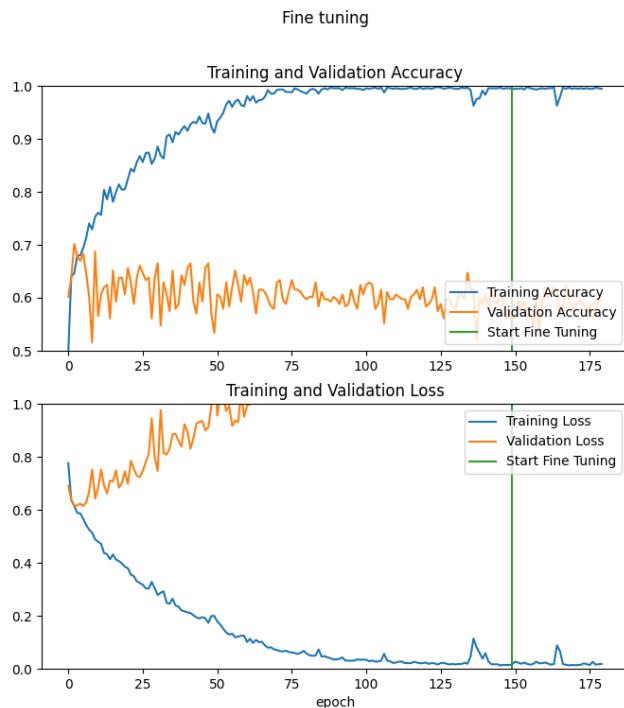


Figura 3.3: Test effettuato usando come learning rate 10^{-3}

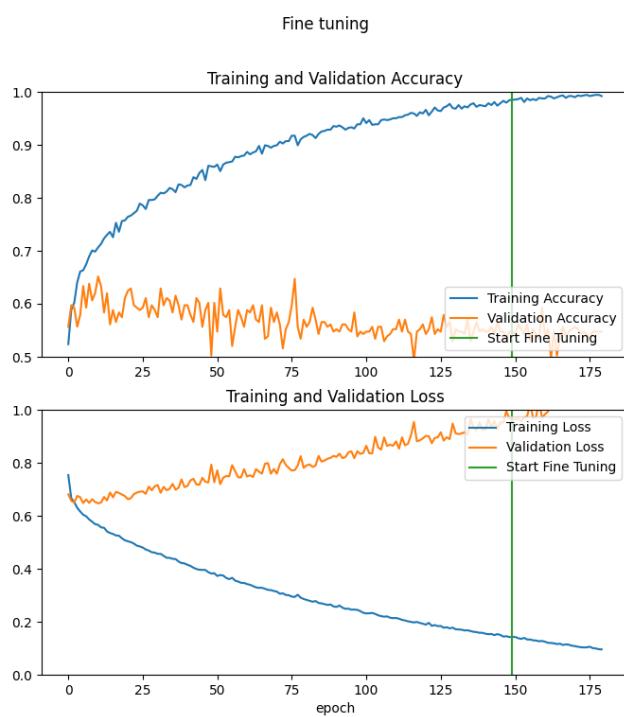


Figura 3.4: Test effettuato usando come learning rate 10^{-4}

3.4 Overfit 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 non tende ad aumentare.

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 orizzontale, 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 unicamente sul training set.

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.

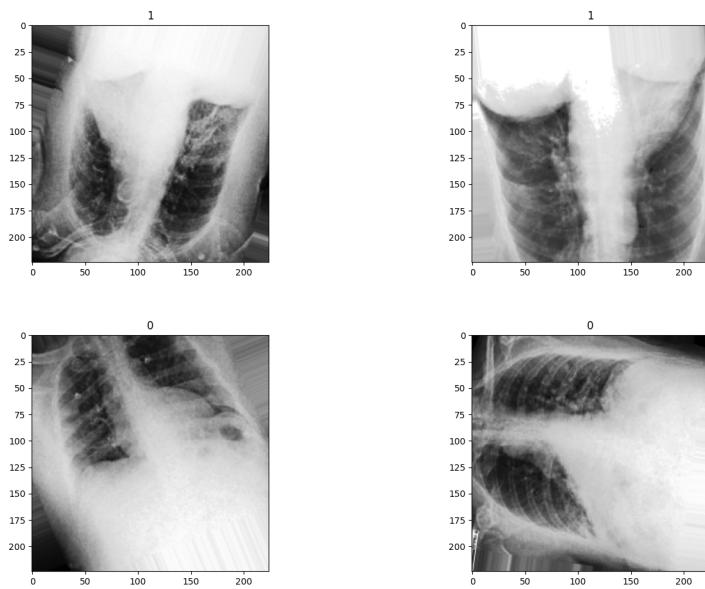


Figura 3.5: Immagini a seguito dell'augmentation, con relativa label

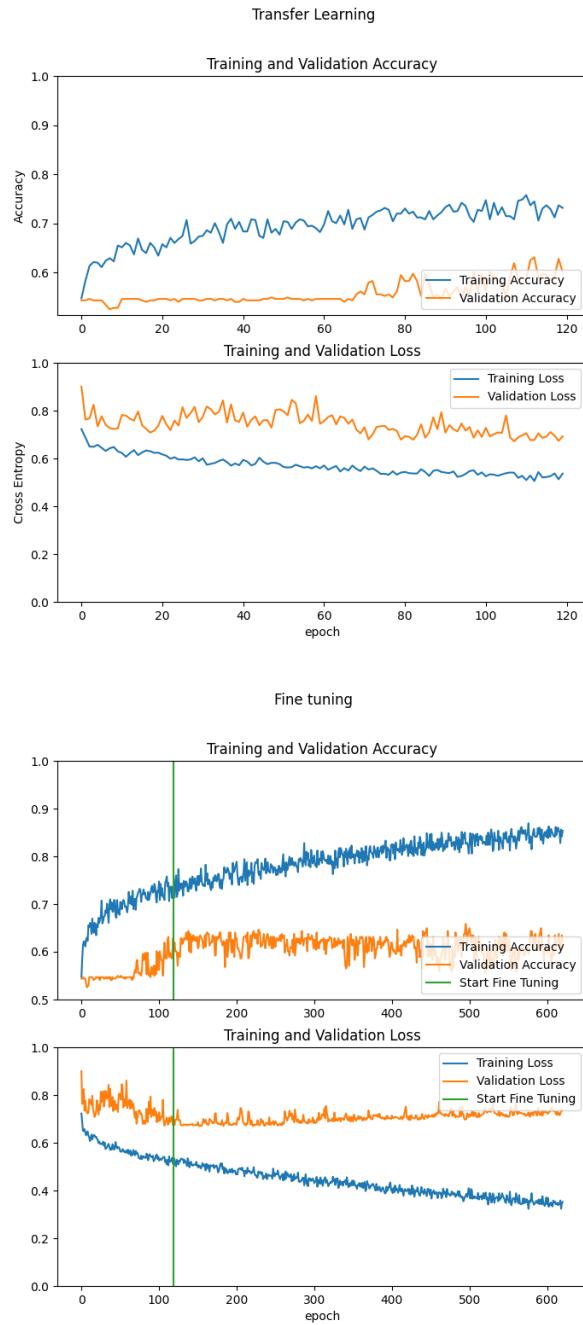


Figura 3.6: Training effettuato con learning rate pari a 10^{-3} a seguito dell'augmentation

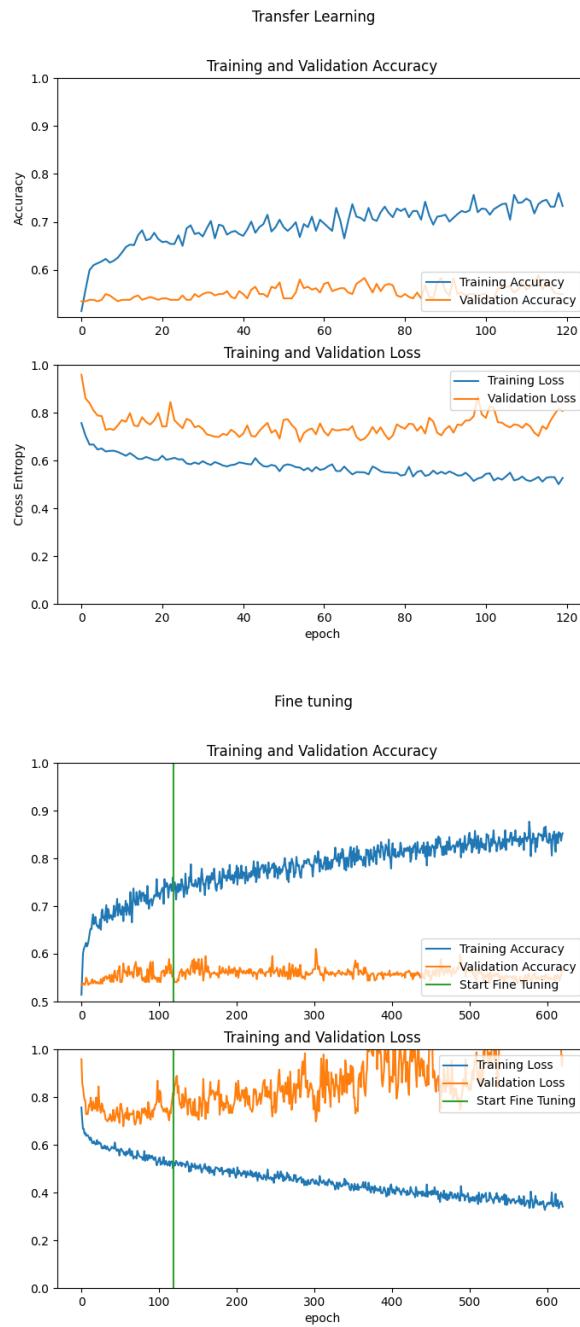


Figura 3.7: Training effettuato con learning rate pari a 10^{-4} a seguito dell'augmentation

4 | Gestione dati eterogenei

4.1 Gestione dei dati

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, esprimo 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 presenti 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	Hospital	Age	Cough	WBC	HeartFailure	Ictus	HighBloodPressure	Prognosis	
...
P_281.png	E	68	...	0	7.33	0	0	1	MILD	
P_544.png	F	72	...	0	9.6	0	0	1	SEVERE	
P_657.png	C	83	...	1	9	0	0	1	SEVERE	
P_1_93.png	F	66	...	0	10	0	0	0	SEVERE	
P_73.png	A	48	...	1	9.13	0	0	0	MILD	
...

Ora che si presenta il dataset completo, si procede col trasformare tutti i dati nello stesso formato, ovvero in valori binari [4]. 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 MLP

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.

4.3 Creazione rete composta

Per poter usare contemporaneamente le due reti neurali create, ovvero la CNN e la 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 previsioni 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.
immagine rete completa 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.

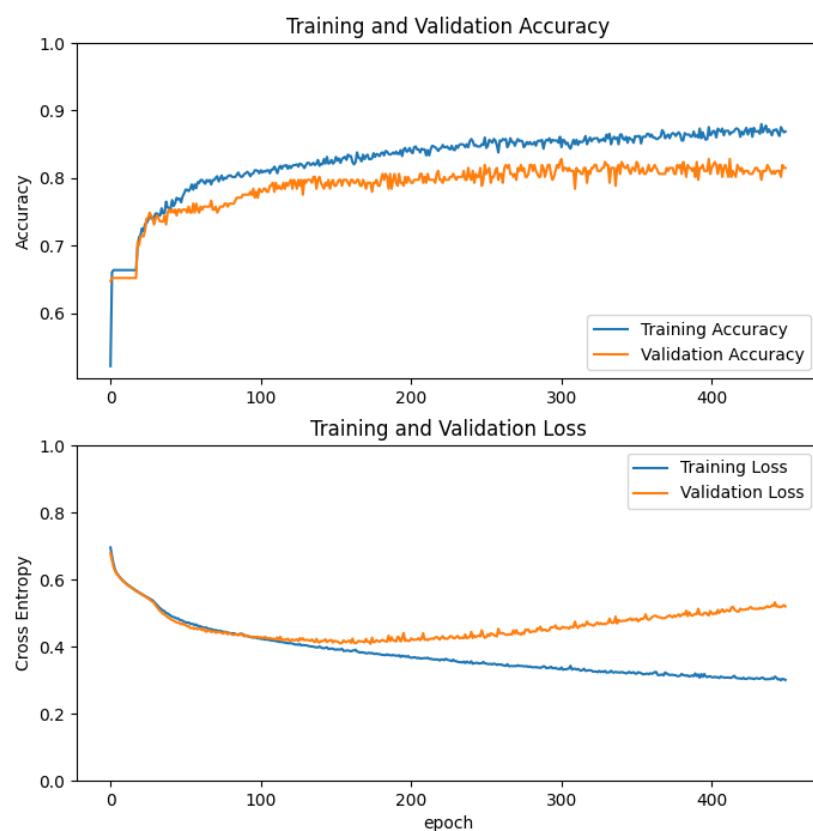


Figura 4.1: Test effettuato usando come learning rate 10^{-3}

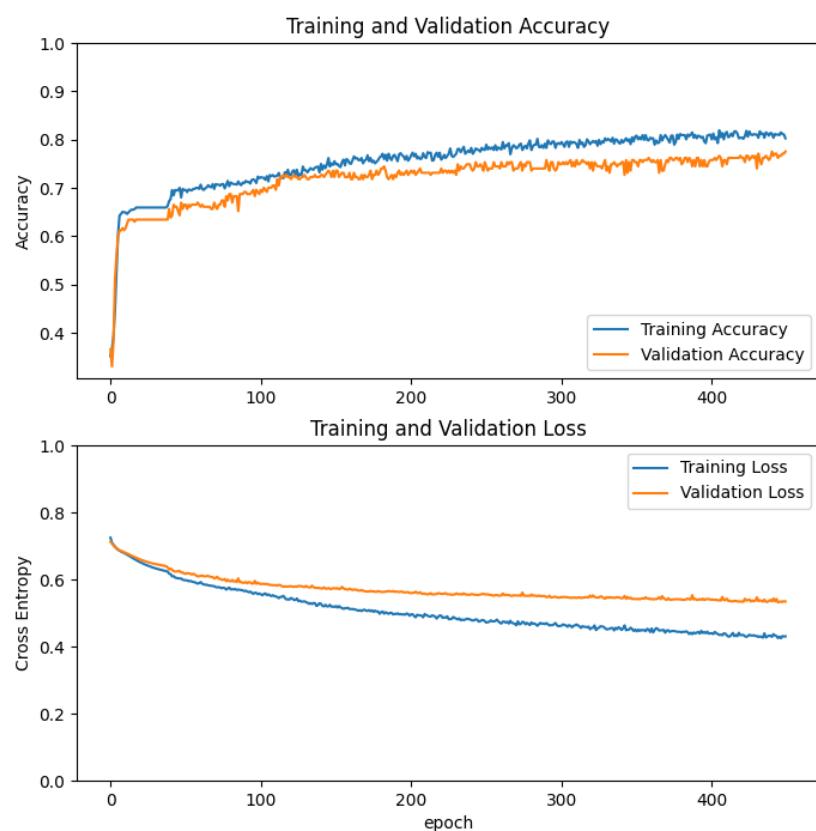


Figura 4.2: Test effettuato usando come learning rate 10^{-4}

5 | Risultati

5.1 Risultati ottenuti

Conclusioni

In questo progetto di tesi si è focalizzata l'attenzione sull'uso dei reti neurali convoluzionali e di percepitrone multistrato (MLP) per analizzare ed effettuare previsioni usando un dataset fornito. Il dataset si può suddividere nelle immagini, raffiguranti radiografie del busto, e nei metadati relativi alle immagini. Per comprendere come creare una rete neurale in grado di prendere in ingresso sia le immagini che i metadati relativi, si è partito studiando i due casi in maniera separata.

Lo studio delle immagini e di come la relativa rete le processa risulta più complesso, per tale motivo si è partiti con l'affrontare tale problematica. In primo luogo si è compreso la composizione del dataset e 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. Tali passaggi sono :

- Verificare se l'immagine presentava fosse stata posta in negativo,
- Correggere la luminosità dell'immagine e equalizzare i colori usando opencv,
- Ridimensionare l'immagine aggiungendo il padding per evitare di modificare l'aspect-ratio dell'immagine.

Alla fine di tali passaggi si è ottenuto un dataset migliore. Per procedere con il training della rete convoluzionale, si è dovuto prima effettuare un altro passaggio riguardante al creazione di immagini in cui si vedono solamente i polmoni. Tale passaggio è composto da:

- Creazione delle maschere mediante l'uso della U-Net,
- Creazione delle bounding-box usando le maschere trovate,
- Ridimensionamento ed eventuale padding finale dell'immagine.

La rete usata come U-Net è EfficientNet e sono stati usati dei pesi preallenati per accellerare il processo e per ottenere maschere migliori.

Una volta termimmato questo studio sulle immagini si è passato al training vero e proprio della rete convoluzionale. La rete usata è stata composta usando come base MobileNetV2. Si è deciso anche qui di usare dei pesi pre-allenati, ma in questo caso si è dovuto effettuare un pasasggio di Fine tuning per adattare i pesi al dataset usato. A tale rete sono stati aggiunti layer per ridimensionare correttamente l'output ed ottenere dei valori binari. Per allenare tali layer si è dovuto effettuare il 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 un insieme di 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 sottinsieme completo del dataset, poiché erano presenti dati mancanti. Oltre a ciò, essendo il dataset suddiviso in training e test set, si è dovuto scegliere un sottinsieme comune ai due dataset.

Ottenuto il dataset finale si è proceduto col normalizzare le varie tipologie di dati presenti in dati binari, ad esempio trasformando dati categorici, con più di due categorie, in dati binari, oppure trasformando dati interi in dati binari.

A seguito della creazione dell'MLP si è proceduto, in fine, con la creazio-ne di una rete unica, composta dalle precedenti in modo che possa prendere in ingresso sia immagini che metadati e restituire un risultato più accurato. Sono stati, anche in questo caso, effettuati dei training e valutati i risultati.

5.2 Sviluppi Futuri

Bibliografia

- [1] google. «TensorFlow.» (2022), indirizzo: <https://www.tensorflow.org/>.
- [2] «ai4covid.» (2022), indirizzo: <https://ai4covid-hackathon.it/>.
- [3] Valerio Guerrasi e Paolo Soda. «Covid CXR Hackathon.» (2022), indirizzo: <https://github.com/cosbidev/COVIDCXRChallenge>.
- [4] A. Rosebrock, «Keras: Multiple Inputs and Mixed Data,» 2021. indirizzo: <https://pyimagesearch.com/2019/02/04/keras-multiple-inputs-and-mixed-data/>.