

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. Giuseppe Notarstefano

Correlatore:

Name Surname

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	6
2.3 Metadati	6
3 Implementazione: CNN	7
3.1 Creazione rete neurale	7
3.2 Transfer Learning	8
3.3 Fine Tuning	10
3.4 Data Augmentation	12
4 Gestione dati eterogenei con MLP	13
4.1 Gestione dei dati	13
4.2 Struttura della rete	15
5 Risultati	16
5.1 Rete finale	16
5.2 Risultati ottenuti	16
A Appendice	17
Bibliografia	17

Abstract

L'uso di tecniche di machine learnig è molto

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.

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

inserire link all'hackaton e immagini dataset

Il dataset usato per effettuare lo studio è stato preso da un hackaton 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.

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 categoria, dati che possono essere considerati come booleani e dati interi, come ad esempio l'età.

2.2 Immagini

2.3 Metadati

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 preallentati 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 preallentati
- 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 preallentati 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

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.

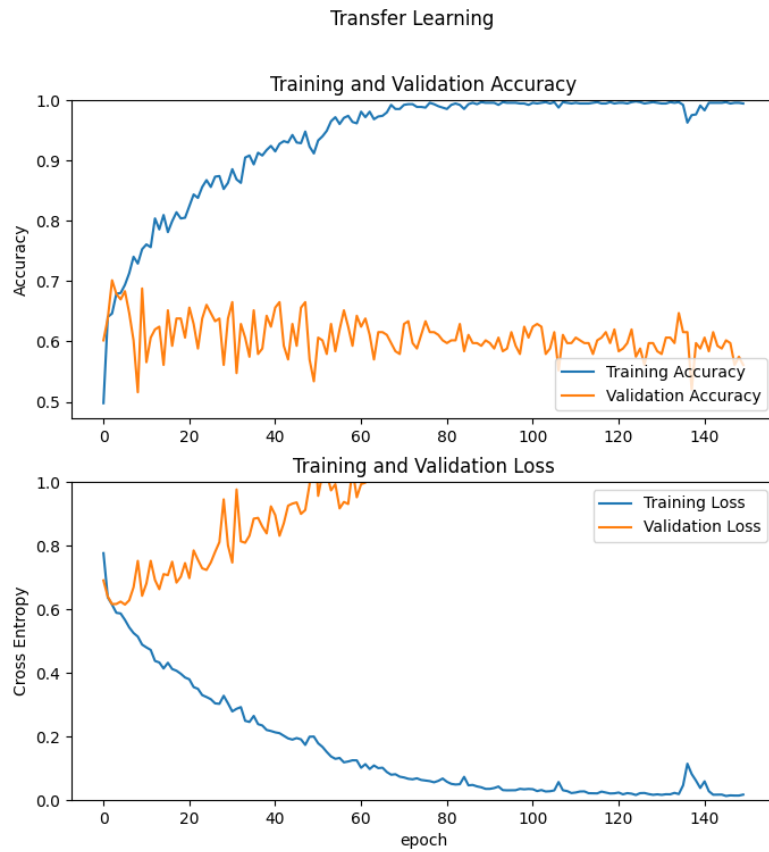


Figura 3.1: Test effettuato usando come learning rate 10^{-3}

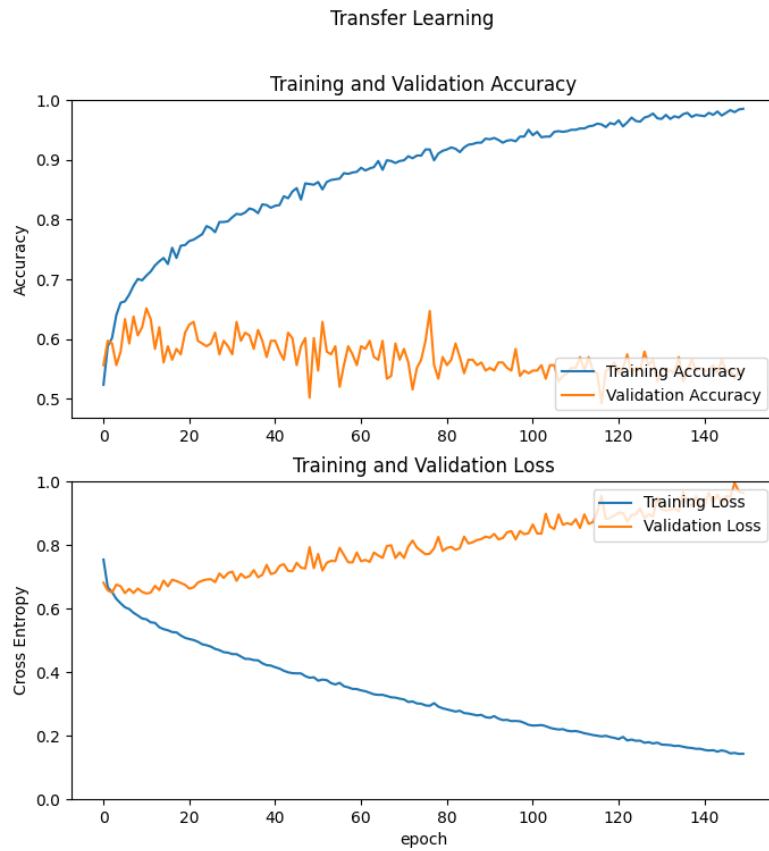


Figura 3.2: Test effettuato usando come learning rate 10^{-4}

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.

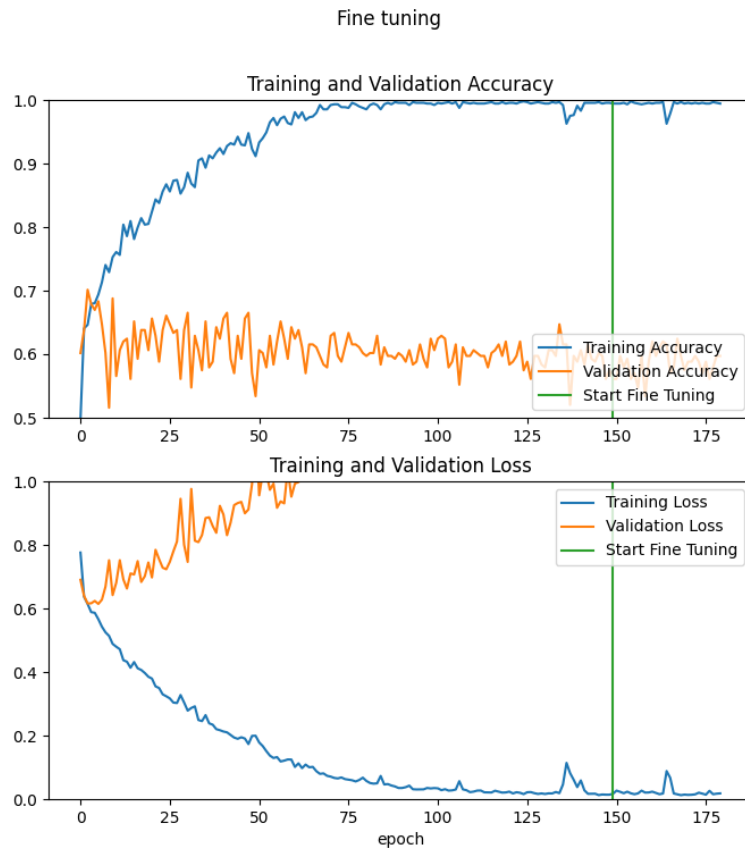


Figura 3.3: Test effettuato usando come learning rate 10^{-3}

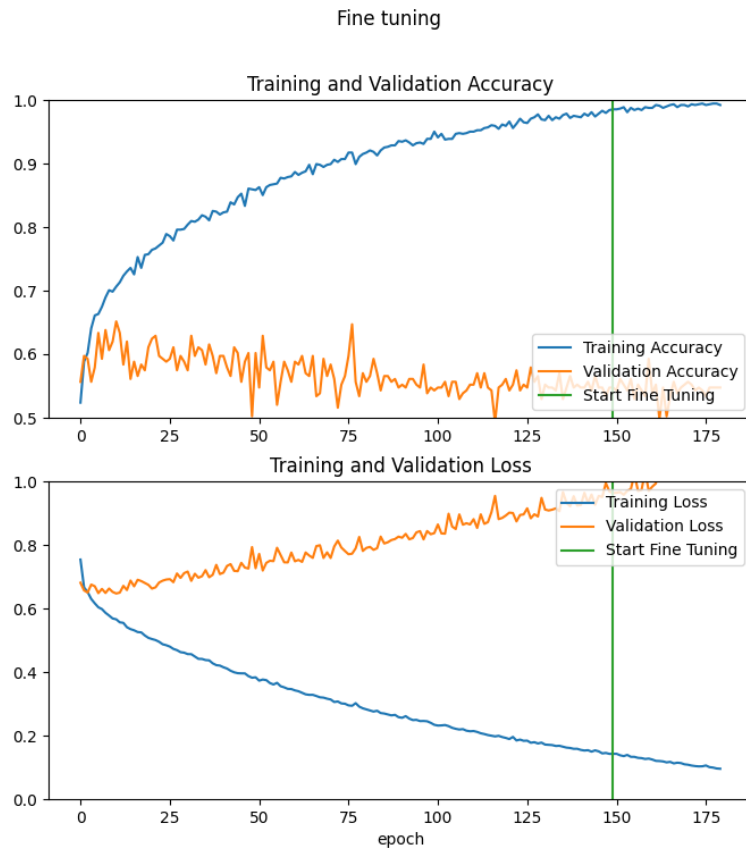


Figura 3.4: Test effettuato usando come learning rate 10^{-4}

3.4 Data Augmentation

4 | Gestione dati eterogenei con MLP

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 maggiorparte 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.

Ora che si presenta il dataset completo, si procede col trasformare tutti i dati nello stesso formato, ovvero in valori binari. 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 necessita di soli due bit per rappresentarla, ovvero una sarà rappresentata da 10 e l'altra da 01. 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:

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

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

I valori binari infine sono rimasti invariati.

4.2 Struttura della rete

5 | Risultati

5.1 Rete finale

5.2 Risultati ottenuti

A | Appendice