Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Informatica

X-ray per la diagnosi di Covid19 con Deep Convolutional Neural Network

Relatore:
Chiar.mo Prof.
Maurizio Gabbrielli

Presentata da:
Daniele Ventura

III Sessione
Anno Accademico 2019/2020

a mia madre,
che non vorrà mai leggere questa tesi,
ma che ha fatto di tutto per farmela scrivere . . .

Abstract

Questa tesi di laurea compie uno studio sull' utilizzo di reti neurali convoluzionali per la diagnosi di covid-19 attraverso l' utilizzo di radiografie. Dopo una breve introduzione sul deep learning, sul funzionamento delle reti convoluzionali e del loro attuale impiego in ambito medico verrà progettato e implementato un modello basato sull' architettura ResNet50 e adattato per renderlo maggiormente funzionale al task di binary classification proposto, questo semplice modello viene poi allenato e testato sul dataset più esteso ed aggiornato esistente e infine confrontato con lo stato dell' arte delle reti che affrontano lo stesso problema in letteratura.

Verrà poi costruita e allenata una rete from scratch da confrontare a quella usata nel tentativo di coglierne similarità e differenze. In ultima istanza viene condotta un' analisi dell' approccio usato evidenziandone le ragioni d'essere (dalle reti convoluzionali all' explainability dei risultati) e i problemi derivati (quantità di dati limitata, bias etc.).

Indice

Abstract **i**

Introduzione **v**

1 Deep Learning **1**

1.1 Storia 1

1.2 Fondamenti di Machine Learning 2

1.2.1 Loss Function 3

1.2.2 Loss Optimization 4

1.2.3 Overfitting 6

1.3 Scheletro di una Rete Neurale: il perceptrone 7

1.4 Elementi principali di una rete neurale 8

1.4.1 Activation Function 8

1.4.2 Backpropagation 9

1.4.3 Regularization 10

1.4.4 Hyperparameters 11

2 Computer Vision **13**

2.1 Introduzione 13

2.2 Storia 13

2.3 Task tipici 15

3 Convolutional Neural Network **17**

3.1 Dettaglio blocchi 18

3.1.1 Convolutional Layer 18
3.1.2 Pooling Layer 19
3.1.3 Fully Connected Layer 20
3.2 Residual Learning e Resnet 22
3.3 Transfer Learning 23

4 Computer Vision e diagnosi mediche 25
4.1 Possibilità 25
4.2 Limiti 26

5 Case study: diagnosi di covid da radiografie 29
5.1 Metodo 29
5.1.1 Dataset e split 30
5.1.2 Data preprocessing 31
5.1.3 Training 32
5.2 Implementazione basata su Resnet50 33
5.3 Implementazione from scratch 35
5.4 Iperparametri 38
5.5 Comparazione Risultati 38
5.5.1 Custom ResNet50 40
5.5.2 Scratch Convolutional Neural Net 41

6 Black box problem ed Explainability 43
6.1 Grad-CAM 43
6.2 Semantic Segmentation 45

Conclusioni 49

Bibliografia 51

Introduzione

Il SARS-CoV2, comunemente detto COVID-19, è un nuovo virus riportato per la prima volta a fine 2019 a Wuhan, Cina. Esso è tristemente famoso in quanto ha causato una pandemia globale [1] e ha costretto molti stati a prendere provvedimenti da stato d' emergenza. Le conseguenti e necessarie chiusure delle varie filiere produttive hanno causato una crisi sociale ed economica che va aggiungendosi a quella sanitaria, aggravando ulteriormente il bilancio degli effetti del virus. Da un punto di vista clinico, l' Istituto Superiore della Sanità riporta che complessivamente il 20% dei casi è grave/critico e necessita dunque di ricovero in ospedale, mentre la letalità si attesta intorno al 3% [2]. A complicare ulteriormente il quadro generale si aggiungono le difficoltà relative alla diagnosi di questa patologia in quanto i sintomi sono spesso indistinguibili da quelli di piú comuni e ben piú innocue influenze virali. Attualmente lo standard per la diagnosi è la tecnica della reazione a catena della polimerasi (rt-PCR) ma essa è costosa, richiede tempo ed è soggetta a errori[3]. Grande attenzione dunque è stata rivolta nell' individuare tecniche alternative alla rt-PCR e nell'ultimo anno sono stati pubblicati molti studi sull' utilizzo di reti neurali convoluzionali (CNN) per la classificazione, data una radiografia polmonare, della positività o meno al virus (alcuni fra i tanti lavori publicati in [4-7]). In questa tesi verranno dapprima analizzati i più influenti nonché recenti studi in letteratura e confrontati i risultati da essi ottenuti, dopodichè basandoci sull' architettura rivelatasi piú promettente, cioè Resnet50, verrà proposta una soluzione custom e dai risultati in linea con lo stato dell' arte. Verrà in seguito effettuato un approfondimento

sulle CNN e sul loro funzionamento e, come collegamento al lavoro svolto precedentemente, verrà implementata una CNN from scratch e paragonata a Resnet50. Infine, si prenderanno in analisi le problematiche relative all' utilizzo di tali tecniche in ambito medico e verranno proposte alcune soluzioni per limitarne gli effetti.

Capitolo 1

Deep Learning

Sebbene vi sia confusione su cosa sia rigorosamente Deep Learning e cosa invece no [8], esso potrebbe essere definito brevemente come "un insieme di tecniche basate su reti neurali artificiali organizzate in diversi strati" [30]. Da questa definizione si evince che il cardine è proprio la nozione di *rete neurale* che pertanto viene esaminata da un punto di vista storico e formale nelle sezioni seguenti.

1.1 Storia

Le origini dell' apprendimento automatico risalgono al 1943 quando Warren McCulloch e Walter Pitts pubblicarono un lavoro sul possibile funzionamento dei neuroni e modellarono una semplice rete neurale con dei circuiti elettrici. Nel 1949 Donald Hebb nel suo lavoro *The organization of behavior* formula invece un principio sul funzionamento delle connessioni fra neuroni ancora oggi in auge e rappresentato analiticamente dall' idea di avere connessioni *pesate* fra neuroni. Il successivo sviluppo degno di nota nell' evoluzione dell' apprendimento automatico si ebbe nel 1958 con l' introduzione del Perceptron, il primo modello di apprendimento supervisionato. Esso era troppo primitivo e semplice per poter essere di una qualche utilità (non era in grado di calcolare uno XOR) e venne duramente criticato, dieci anni dopo, da

Minsky e Papert. Fu solo negli anni 80, infatti, che si ebbero i primi sviluppi decisivi in questo campo ad opera di Hinton, Rumelhart e Williams che, nel celeberrimo articolo sulla backpropagation [9], descrissero la tecnica in uso ancora oggi per *allenare* le reti. Come visto dunque gli sviluppi teorici sulle reti neurali hanno radici profonde e lontane nel tempo ma è solo negli ultimi anni che il campo è esploso grazie alla disponibilità di hardware adeguato (CPU con possibilità di calcolo parallelo) e all' enorme reperibilità di dati rispetto al passato.

1.2 Fondamenti di Machine Learning

L'apprendimento automatico o machine learning è quell'ambito della ricerca che si occupa della costruzione di algoritmi che abbiano la capacità di apprendere dall'esperienza senza essere esplicitamente programmati a priori. Esso si basa su alcuni punti fondamentali:

- Un *modello* per il task da affrontare, dipendente da un set di parametri Θ
- Una valutazione di performance, ovvero una misura dell' errore per valutare il modello
- L' ottimizzazione dei parametri Θ per minimizzare l' errore.

Questa tecnica prende il nome di *learning* perchè l' ottimizzazione di Θ si basa sull' osservazione dei dati e sull' uso di tecniche iterative. Vi sono tre grandi paradigmi di apprendimento automatico:

- **supervised learning**: Per supervised learning si intende quella branca del machine learning nella quale al modello viene fornito un insieme di dati input con le corrispettive etichette o output, dette *labels*. Lo scopo del modello è quindi quello di imparare una correlazione fra questi due insiemi in modo da poter effettuare predizioni su nuovi dati. Possiamo riconoscere due sottoclassi di problemi a seconda che l' ouput sia continuo o discreto:

  - problemi di regressione: laddove lo spazio delle predizioni è tutto $\mathbb{R}$, ovvero il campo dei numeri reali.
  - problemi di classificazione: in questo caso invece l' obiettivo è quello di fare predizioni su uno spazio discreto, questo si traduce con il determinare l' appartenenza dell' input a un numero finito di *classi*. Nell' ambito di questa tesi verrà usato questo tipo di apprendimento per dividere le radiografie in due classi: positività o negatività al Covid19.

- **unsupervised learning**: Per quanto riguarda l' unsupervised learning al modello vengono invece forniti solo dati di input senza i corrispettivi *label*, esso deve quindi imparare correlazioni solo in base alla struttura dei dati. Gli algoritmi più utilizzati in questo campo sono quelli di clustering, component analysis o anomaly detection.

- **reinforcement learning**: Questo ultimo paradigma è invece basato sui concetti di *azione* e *ricompensa*. Esso punta a realizzare agenti autonomi in grado di prendere decizioni sequenziali, in cui l'azione da compiere dipende dallo stato attuale del sistema e ne determina quello futuro.

1.2.1 Loss Function

La *Loss function* è una funzione che misura l' errore delle predizioni non corrette. Il suo calcolo, come vedremo nella prossima sezione, intuitivamente guida l' apprendimento della rete. Tramite essa infatti è possibile calcolare in modo rigoroso la bontà delle predizioni del nostro modello. Per i problemi di classificazione binaria la Loss function più usata è quella detta *Binary Cross Entropy Loss* che possiamo definire come segue: Dati y = label, $p(y)$ = label predetto dalla rete , N = numero di input

$$H(X) = -\frac{1}{N}\sum_{i=1}^{N} y_i log(p(y_i)) + (1 - y_i) log(1 - p(y_i)) \tag{1.1}$$

Nei problemi di regressione invece la Loss function più comunemente usata è quella detta *Mean Squared Error* definita come:

$$MSE = \frac{\sum_{i=1}^{N}(y_i - p(y_i))^2}{N} \tag{1.2}$$

L' obiettivo sarà quello di minimizzare questa funzione facendo variare i parametri Θ presenti nel modello, il problema si configura quindi un problema di ottimizzazione:

$$min\Theta J(\Theta), \Theta \in \mathcal{R}^N \tag{1.3}$$

1.2.2 Loss Optimization

Come anticipato nella sezione precedente, la fase di apprendimento di una rete neurale può essere vista come un problema di ottimizzazione e cioè la minimizzazione della Loss function.

Discesa del gradiente: è la tecnica principe per questo tipo di ottimizzazione ed ha lo scopo di trovare i punti di minimo di una funzione a più variabili. Nel dettaglio, come descritto in [10], la tecnica della discesa del gradiente è un modo di minimizzare una funzione di costo $J(\Theta)$ parametrizzata dai parametri $\Theta \in \mathcal{R}^N$ del modello attraverso l' aggiornamento degli stessi parametri nella direzione opposta al gradiente della funzione J. Il *Learning Rate* α determina la misura del passo fatto per raggiungere il minimo locale. In altre parole si segue, verso il basso, la pendenza della superficie creata dalla funzione di costo fino a raggiungere il minimo.

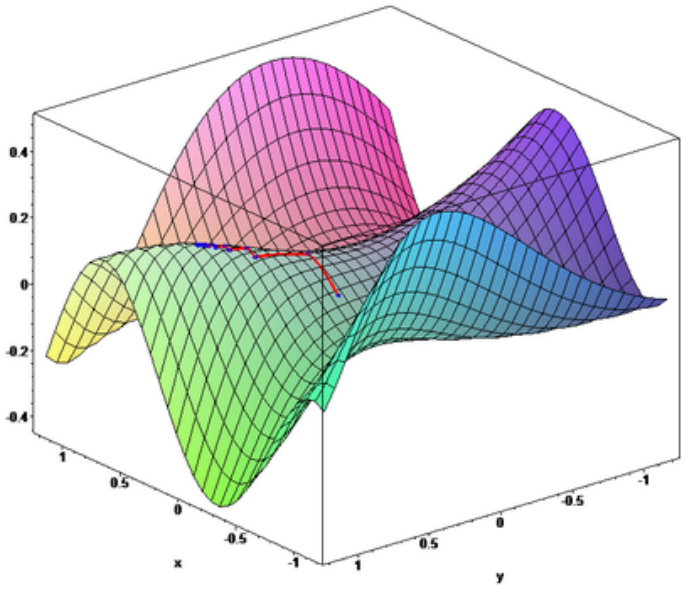


Figura 1.1: Ricerca del minimo di una funzione attraverso la discesa del gradiente

Esistono tre varianti nell' utilizzo di questa tecnica:

- Stochastic gradient descent;
- Batch gradient descent;
- Mini-Batch gradient descent.

La più comune, utilizzata anche in questa tesi, è l' ultima che prevede l' aggiornamento dei parametri ogni *batchsize* dati di input. Ad alto livello essa può essere descritta come:

```
for i in range(nb_epochs ):
    for  batch  in  get_batches(data , batch_size):
        params_grad = eval_gradient(loss_f, batch, params)
        params = params - learning_rate * params_grad
```

1.2.3 Overfitting

In statistica e in informatica, si parla di *overfitting* quando un modello statistico molto complesso si adatta ai dati osservati (il campione) perchè ha un numero eccessivo di parametri rispetto al numero di osservazioni. In modo informale si potrebbe dire che un algoritmo di apprendimento presenta overfitting rispetto a un altro più semplice se è più accurato nel fare predizioni su dati conosciuti ma ha performance peggiori nell' effettuare predizioni su nuovi dati. Il modello in questo caso viola i principi del *rasoio di Occam* e, non essendo capace di generalizzare, è di poca utilità.

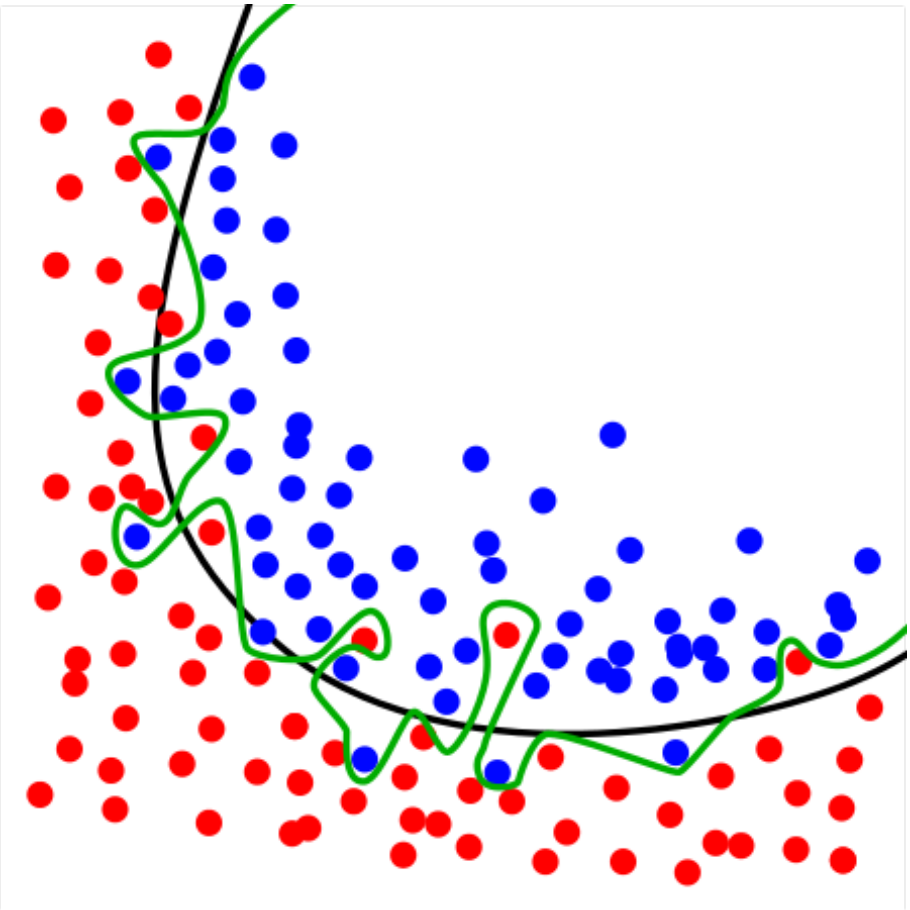


Figura 1.2: linea verde: modello con overfitting, linea nera: modello regolarizzato

1.3 Scheletro di una Rete Neurale: il perceptrone

Analizziamo ora più in dettaglio il funzionamento di una rete neurale. Come elemento costitutivo fondamentale possiamo considerare il perceptrone (visto però in un' accezione moderna diversa da quella di Rosenblatt) ovvero un singolo neurone di una rete neurale.

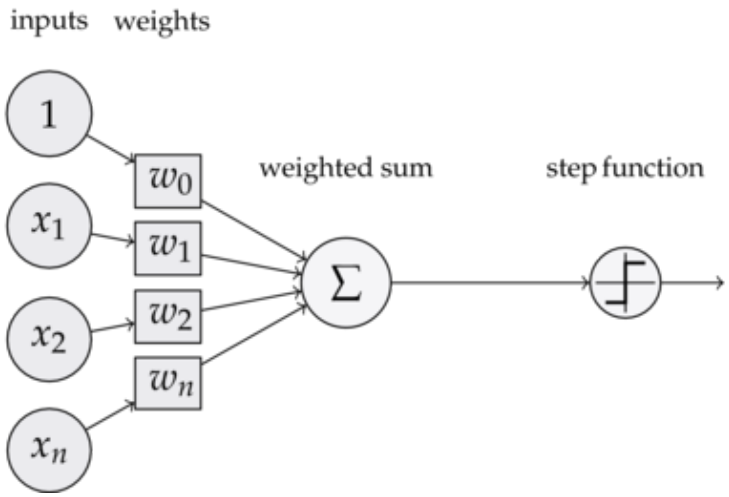


Figura 1.3: schema del funzionamento di un percettrone

Dove:

- $1, \ldots, x_m$ sono gli input;
- $w_0, \ldots, w_m$ sono i *pesi* associati alla connessione;
- $stepfunction$ è la funzione di attivazione non lineare (che verrà descritta in seguito);

Possiamo quindi riassumere il funzionamento di un singolo neurone con la formula:

$$y = f(b + \sum_{i=1}^{n} x_i w_i) \tag{1.4}$$

Dove b è un vettore di n elementi detto *bias*.

1.4 Elementi principali di una rete neurale

Per capire il funzionamento di una rete neurale, dopo averne visto l' elemento strutturale, è necessario andare a considerare alcune delle intuizioni matematiche che ne stanno alla base.

1.4.1 Activation Function

Per Activation Function intendiamo una classe di funzioni con lo scopo di introdurre *non linearità* nel modello. Infatti, senza l' utilizzo di funzioni di attivazione il modello potrebbe solo effettuare separazioni lineari che mal si adattano a rappresentare fenomeni complessi. Usando il paragone biologico coi neuroni esse modellano l' attivazione o l' inibizione di una data connessione.

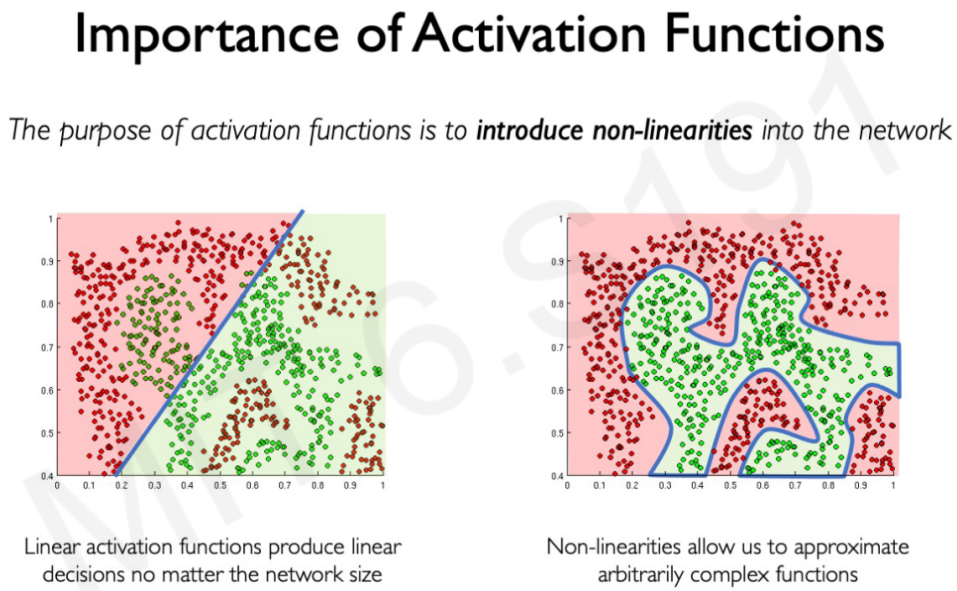


Figura 1.4: scopo delle funzioni di attivazione

Fra le funzioni di attivazione più comuni possiamo elencare:

- Sigmoid function;
- Hyperbolic tangent function;
- ReLU or Rectified Linear Unit (quella usata in questa tesi).

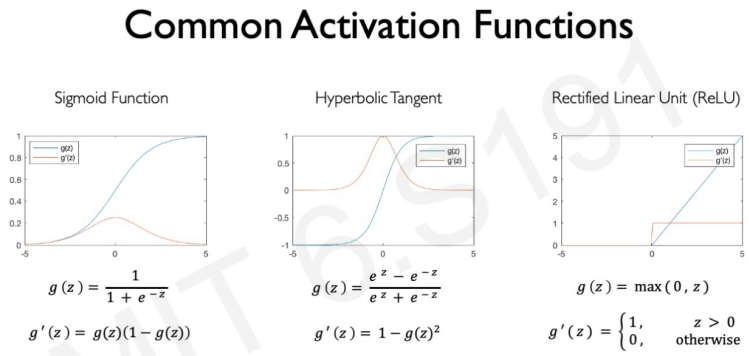


Figura 1.5: le più usate funzioni di attivazione

1.4.2 Backpropagation

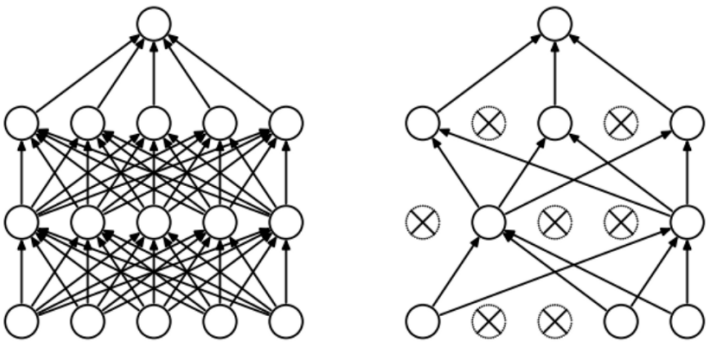
Come già accennato precedentemente, questa tecnica venne scoperta negli anni '80 da Hinton, Rumelhart e Williams [9] e diede nuova linfa al campo dell' intelligenza artificiale. Per backward propagation intendiamo un algoritmo che permette di conciliare l'idea di discesa del gradiente con la struttura multi-livello delle reti neurali artificiali allo scopo di permettere un calcolo efficiente rispetto ai vari parametri della rete. È chiamato backward propagation perchè il gradiente della loss function viene calcolato rispetto ai pesi dell'ultimo layer, per poi aggiornare a ritroso i parametri dei layer precedenti fino a raggiugnere il primo. L' intuizione matematica alla base di questa tecnica è l' utilizzo della *regola della catena* per il calcolo delle derivate a ritroso, ovvero dall' ultimo layer al primo.

1.4.3 Regularization

Con il termine *regularization* si intende un insieme di tecniche atte a minimizzare la complessità del modello allo scopo di ridurre l' overfitting.

- **L2 Regularization**: La forma di regularization più comunemente usata è quella detta *L2 Regularization*: Un tipo di regolarizzazione che penalizza i pesi in proporzione alla somma del loro quadrato. Essa può essere calcolata aggiungendo alla già nota funzione di costo un ulteriore termine: $\lambda \sum_{i=1}^{M} W_j^2$ dove M è il numero dei pesi. Questo accorgimento tende a far decadere il valore dei pesi intorno a 0 permettendo al modello di ridurre l' importanza di valori *fuori norma*, in modo tale da aumentare la capacità di generalizzazione del modello.

- **Dropout**: Con il termine *Dropout* invece si intende una tecnica di regolarizzazione definita in [13] secondo la quale alcune unità della rete vengono stocasticamente *gettate via* (da qui il termine drop out) insieme alle loro connessioni in input e output. È una tecnica di semplice implementazione ma che per la sua efficacia ha riscosso un grande successo per la regolarizzazione di reti molto profonde.



(a) Standard Neural Net (b) After applying dropout.

Figura 1.6: esempio di dropout

1.4.4 Hyperparameters

Gli *hyperparameters*, o iperparametri, sono le variabili che determinano la struttura della rete (ad esempio il numero di *hidden layer*) e le variabili che determinano *come* effettuare l' apprendimento della rete. Gli iperparametri vengono impostati prima della fase di training e ne regolano vari aspetti, di seguito un elenco dei più significativi:

- Percentuale di Dropout;
- Activation function usata;
- Learning rate;
- Batch size;
- Numero di epoche.

Per quanto riguarda le reti convoluzionali, cambiando la struttura e la funzione dei layer, ne entrano in gioco altri che verranno descritti in seguito.

Capitolo 2

Computer Vision

2.1 Introduzione

La Visione Articiale (da qui in poi Computer Vision) è quel campo dell' intelligenza artificiale che mira a replicare le capacità visive umane. È importante sottolineare che per *capacità visive* non si intende solamente la facoltà di di vedere contorni e immagini ma anche quella di imitare la *percezione*, ovvero la capacità di conferire senso a ciò che si vede. Questo capitolo introduttivo è essenziale ai fini della comprensione delle reti neurali convoluzionali e per avere una visione d'insieme di un'area attualmente al centro della ricerca scientifica.

2.2 Storia

Pur essendo finito sotto l' attenzione generale solo di recente (si vedrà in seguito esattamente quando) il campo della Computer Vision, così come l' intelligenza artificiale in generale, non è sicuramente nuovo. Infatti la sua storia ha molto in comune con quella già vista del Machine Learning: i primi sviluppi si ebbero intorno agni anni '50 del secolo scorso e nel 1959 grazie a Kirsch si ebbe la prima immagine digitale della storia.

Figura 2.1: il figlio di Kirsch, una delle prime immagini digitali create

Altro momento importante fu la pubblicazione nel 1963 da parte di Roberts di un lavoro dal titolo *Machine perception of three-dimensional solids* [14] considerato come il vero punto di partenza della ricerca accademica in questo campo. Gli anni seguenti invece furono rilevanti più per i loro fallimenti che per i successi tanto che fino agli anni '80 si parla di *inverno dell' intelligenza artificiale*. Nel 1980 uno scienziato giapponese, Kunihiko Fukushima, creò quello che è considerato il primo esempio di rete *deep*, ovvero

formata da una moltitudine di layer, chiamata *Neocognitron.*[15] Il successivo passo fondamentale si ebbe nel 1989 quando Yann LeCun (che ha conseguito il premio Turing nel 2018 per i suoi studi in questo campo, insieme a Yoshua Bengio e Geoffrey Hinton) applicò l'algoritmo di backpropagation alla rete di Fukushima creando LeNet-5, la prima rete convoluzionale moderna. Nei venti anni seguenti alla nascita di LeNet, sebbene la ricerca sia continuata e abbia progredito, non si ebbero breakthrough di fondamentale importanza e l' interesse per il campo prese a scemare. Fu solo nel 2012 con la creazione di AlexNet da parte di Alex Krizhevsky che rinacque l' interesse da parte del mondo accademico (e non solo). Questa celeberrima rete, basata a sua volta su quella di LeCun, può essere considerata come l' archetipo di tutte le reti convoluzionali moderne.

2.3 Task tipici

Il campo della computer vision è alquanto vasto così come innumerevoli e disparate sono le sue applicazioni, tuttavia è possibile tracciare un contorno generale di quelli che sono i principali task che un modello di intelligenza artificiale deve compiere.

- **Classificazione di immagini**: È probabilmente l' applicazione più comune, si propone l' obiettivo di classificare appunto l' immagine in input come appartenente a una determinata categoria, o classe, presa da un insieme dato e predeterminato in precedenza

- **Localizzazione**: Obiettivo della localizzazione è quello di trovare la locazione di un singolo oggetto all' interno di un' immagine. Tipicamente ciò viene compiuto disegnando un contorno intorno all' oggetto detto *bounding box.*

- **Rilevamento di oggetti**: Simile al task precedente con la differenza che il numero di oggetti da localizzare è indefinito a priori e variabile da zero a molti.

- **Segmentazione semantica**: Per segmentazione semantica, o *semantic segmentation*, si intende il processo di assegnare a ogni pixel di un immagine un'etichetta tale che pixel aventi le stesse caratteristiche (i.e. appartenenti allo stesso oggetto o stessa classe) abbiano la stessa etichetta.

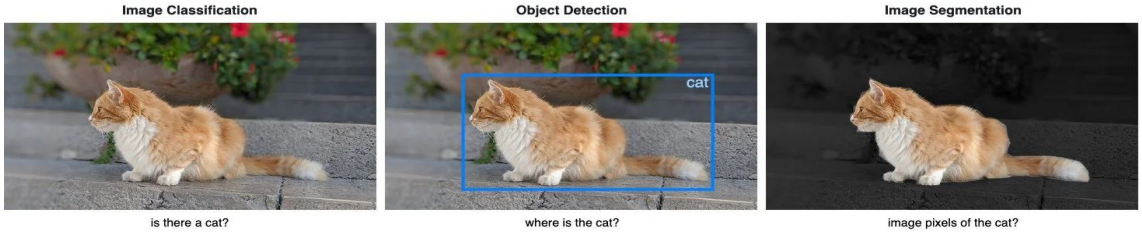


Figura 2.2: differenze fra i vari task

Capitolo 3

Convolutional Neural Network

Come introdotto nel capitolo precedente, si possono considerare le reti neurali convoluzionali moderne come evoluzioni di LeNet-5. Prendiamo ora in esame il perchè tali reti sono diventate dominanti nel campo della computer vision e cosa le differenzia dalle normali reti feed-forward. Una rete neurale convoluzionale è un tipo di rete neurale artificiale in cui la connettività tra neuroni è ispirata dall'organizzazione della corteccia visiva animale. L' idea originale infatti fu quella di simulare processi biologici. [16] Da un punto di vista architetturale, come ogni rete neurale, una CNN consiste di un input layer, vari hidden layers e un output layer. Tuttavia, in una CNN, gli strati nascosti effettuano operazioni dette *convoluzioni*, che le rendono particolarmente adatte nell' analisi di immagini. Per costruire una rete convoluzionale vengono usati principalmente tre tipi di *blocchi*: Convolutional, Pooling e Fully-Connected, i quali vengono combinati in vari modi per formare l' architettura completa. Il pattern più comune con cui questi blocchi vengono combinati è generalmente di questa forma:

$INPUT -> [[CONV -> RELU] * N -> POOL?] * M -> [FC -> RELU] * K -> FC$

Dove $*$ indica la ripetizione e il punto interrogativo denota l' opzionalità del layer. [18] Per capire come si faccia ad arrivare da un immagine all' ouptut desiderato è necessario vedere in dettaglio ciascuno di questi blocchi.

3.1 Dettaglio blocchi

3.1.1 Convolutional Layer

Il layer convoluzionale è ciò che distingue una CNN da una rete neurale *standard*. L' elemento principale è chiamato filtro o *kernel* e la sua dimensione è un iperparametro della rete. Ogni filtro è piuttosto limitato nelle dimensioni (tipicamente 3x3 o 5x5) per quanto riguarda lunghezza e larghezza ma si estende per l' intera profondità dell' input. (1 per immagini in bianco e nero, 3 per RGB etc.) Durante l' esecuzione dell' algoritmo (forward-pass) ogni filtro viene fatto scivolare (*convolvere* appunto) lungo tutto l' input e viene calcolato il prodotto scalare fra i valori presenti nel filtro e la porzione di input presa in esame. Ripetendo questa operazione per tutta la lunghezza e larghezza dell' immagine si ottiene in output una *activation map* a 2 dimensioni che restituisce la "risposta" dell' applicazione del filtro all' immagine. La rete impara così a distinguere caratteristiche dell' input quali contorni, particolari insiemi di colori, forme etc. Procedendo in profondità nella rete le caratteristiche imparate dai filtri saranno sempre di livello più alto fino a giungere a patterni complessi quali ad esempio volti o ruote di veicoli. È bene notare che in ogni blocco convoluzionale vengono applicati più filtri, per ognuno di questi viene prodotta un' activation map e la dimensione finale dell' output per quel blocco sarà data dal numero di filtri usati(per quanto concerne la profondità).[18]

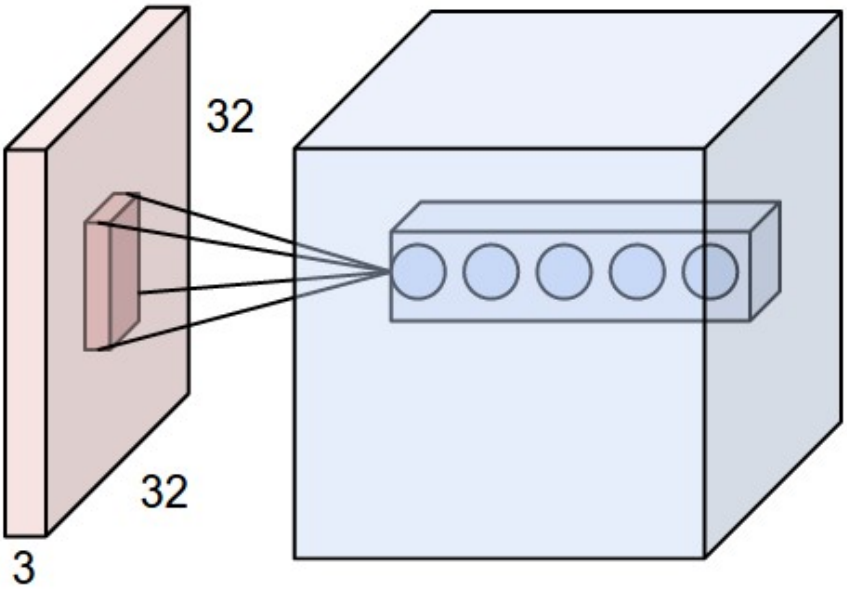


Figura 3.1: applicazione di 5 filtri, l' output sarà di dimensione 32x32x5 (senza considerare downsampling) [18]

Tuttavia se si facesse convolvere il filtro in questo modo, senza utilizzare particolari accorgimenti o tecniche, si andrebbe a ridurre la dimensione dell' output in ogni layer, perdendo troppo velocemente informazione. Infatti, se l' immagine in input ha dimensioni di $n \times n$ e si utilizza un filtro $f \times f$ allora l' output avrà dimensione $(n - f + 1) \times (n - f + 1)$. Per ovviare a questo problema si utilizza una tecnica chiamata *padding* che consiste nell' aggiungere pixel ai bordi (generalmente pixel dal valore 0, da cui il nome *zero padding*) in modo da preservare la dimensione dell' input.

3.1.2 Pooling Layer

Scopo del Pooling layer è quello di ridurre la dimensione dell' input senza perdere informazioni siginificative. Applicando ripetutamente questo tipo di blocchi infatti è possibile ridurre il numero complessivo di parametri e pertanto di computazioni che la rete deve calcolare, rendendola più efficiente

e meno soggetta a overfitting. Il pooling viene effettuato selezionando una porzione dell' input e applicando una funzione (solitamente MAX o AVG, si parla quindi di MAX-Pooling o AVG-Pooling) in modo da riportare in ouput solo il risultato dell' operazione sull' input. [18]

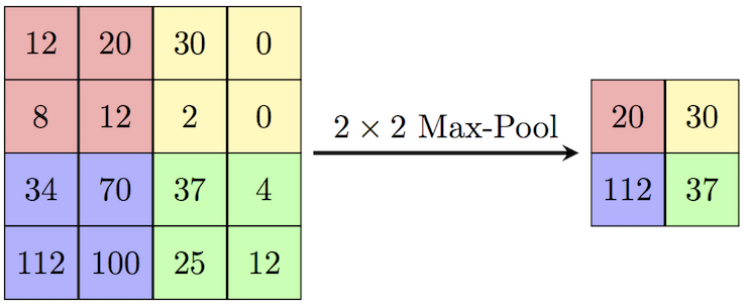


Figura 3.2: esempio di MAX-Pooling

Il pooling è la tecnica principe per ridurre le dimensioni dell' input, come visto nella sezione precedente infatti non è l' unico modo per ottenere questo risultato. Si potrebbe ad esempio anche non usare padding in modo da ridurre le dimensioni ad ogni layer convoluzionale ma così facendo si perderebbero informazioni troppo velocemente rendendo l'apprendimento poco efficiente.

3.1.3 Fully Connected Layer

Il livello denso non presenta invece differenze con una normale rete neurale, dunque come tutti i fully connected layer ogni neurone al livello n è *completamente connesso* con tutti i neuroni al livello $n-1$. La funzione di attivazione viene pertanto calcolata in modo standard con una moltiplicazione fra matrici seguita dalla somma del bias.

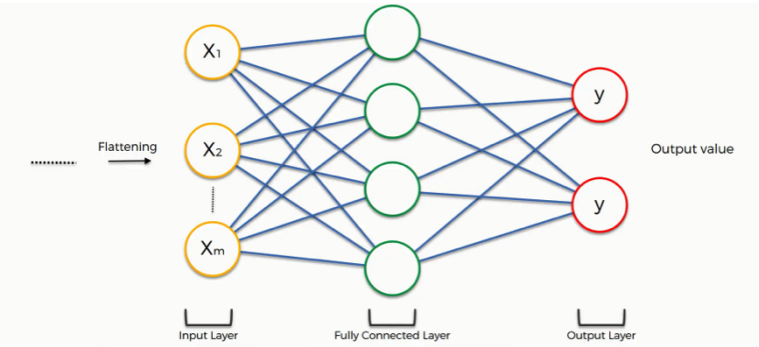


Figura 3.3: layer completamente connesso in una cnn

3.2 Residual Learning e Resnet

Il concetto di Residual Learning è piuttosto nuovo essendo nato nel 2015 dopo l' influente articolo *Deep Residual Learning for Image Recognition* di He et al. [19], la loro rete (ResNet) infatti vinse le maggiori competizioni di computer vision del 2015 e da allora è diventata lo standard anche per l' utilizzo al di fuori dell' accademia. Una rete neurale viene detta *residuale* quando presenta blocchi, detti *identity connections*, che presentano *scorciatoie*, ovvero salti fra le connessioni. Questa tecnica permette di poter costruire reti più profonde aggirando i limiti dell' epoca (i.e. scomparsa del gradiente) permettendo così di ottenere risultati nettamente migliori.

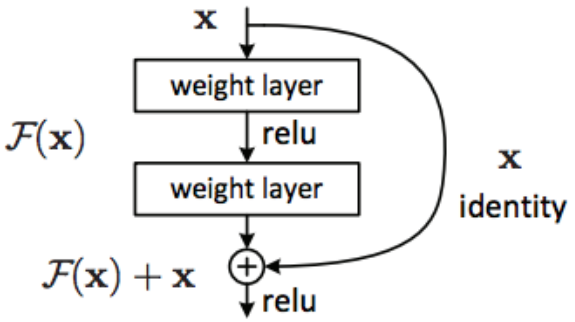


Figure 2. Residual learning: a building block.

Figura 3.4: esempio di identity block

3.3 Transfer Learning

Le moderne architetture di reti convoluzionali sono spesso molto profonde e pertanto presentano milioni di parametri i quali, per essere propriamente allenati, necessitano di un enorme quantità di dati. Tuttavia, nella maggior parte dei casi, dataset delle dimensioni necessarie al *training from scratch* di una rete, ad esempio ResNet, non esistono dunque per ovviare a questo problema si sfrutta una metodologia di allenamento detta Transfer Learning. Questo metodo consiste nell' allenare la rete usando dati provenienti da un dominio diverso da quello *target* (ma quanto più simile) sfruttando così la grande quantità di dati di vario genere presenti in rete, per poi allenare nuovamente alcuni layer, gli ultimi, con i dati del dominio target. Facendo ciò si riesce a sfruttare la capacità delle reti di apprendere feature rilevanti e "generali" per poi trasferire questa conoscenza al task originario. Il numero di dati necessari per l' utilizzo di reti molto deep si riduce significativamente rendendo possibile l' utilizzo di reti quali ResNet o VGG con dataset di poche migliaia di elementi.

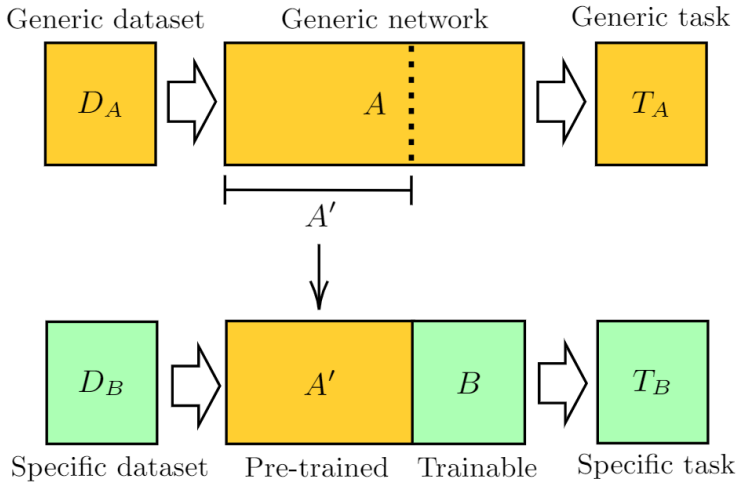


Figura 3.5: funzionamento del transfer learning

Capitolo 4

Computer Vision e diagnosi mediche

4.1 Possibilità

In tutto il mondo stanno diffondendosi startup con l' obiettivo di conciliare il campo dell' intelligenza artificiale a quello medico e i risultati sembrano sostenere questo ottimismo. Anche l'idea di questa tesi nasce sull' onda dei recenti sviluppi nel campo delle analisi di immagini e sulla possibilità di applicarne i risultati nel campo della diagnosi. La pandemia in atto ha fatto da motore per la ricerca in quest'area e nel corso del 2020 sono stati compiuti numerosi sforzi coronati da risultati importanti. Sono decine, se non centinaia, le pubblicazioni a riguardo (fra cui ricordiamo [4-7][29]) che concordano sull' affidabilità nonchè grande precisione delle reti convoluzionali nella diagnosi di Covid19. L' *accuracy* dei modelli in letteratura raggiunge percentuali che lasciano poco spazio alle interpretazioni e in modo simile ciò si ripete nelle altre metriche di valutazione. Questi risultati fanno dunque intendere un' effettiva possibilità di utilizzo di questo strumento nella lotta alla pandemia. Come visto, infatti, una volta che le reti convoluzionali imparano dei *pattern* relativi a un task esse sono in grado di classificare con grande precisione nuovi dati. Ormai da anni queste reti venogno studiate in

modo approfondito e l' attenzione è stata guidata proprio dai risultati sperimentali ottenuti (AlexNet nel 2012 principalmente, ma anche ResNet nel 2015). Le predizioni di questi modelli potrebbero essere usate come tecnica automatizzata per la diagnosi in modo tale da coadiuvare il test rt-PCR essendo più veloci, meno costose e più facilmente reperibili. è inoltre verificato che l' rt-PCR ha un *miss rate*, ovvero il numero di falsi negativi, pari al 30% (ben maggiore dei modelli in esame) quindi il bilancio fra test attualmente disponibili e l' utilizzo di reti convoluzionali sembra far propendere verso un' effettiva e concreta possibilità di utilizzo di queste ultime.

4.2 Limiti

Dopo aver introdotto le motivazioni che hanno spinto molti ricercatori a occuparsi della questione in esame, prendiamo ora in considerazione i problemi e i limiti legati a tale approccio. In [20] viene primariamente considerato il parere degli esperti e cioè l' opinione su queste tecniche delle varie associazioni nazionali di radiologia. Detta opinione è, appunto, unanime nel bocciare senza riserve questo approccio [21-23]. Ma non è il solo argomento contrario, sempre [20] evidenza alcune problematiche di natura metodologica che invalidano i risultati finora ottenuti dalle intelligenze artificiali. Prima di tutto, proprio a causa della natura del task, si può evidenziare un grave *selection bias*, ovvero il sottoinsieme dei casi testati è significativamente diverso da quello a cui si vuole applicare la tecnica (cioè l' intera popolazione). Questo è evidente in quanto sia gli studi che hanno costruito il proprio dataset [24] sia la maggior parte degli altri che hanno utilizzato vari dataset liberamente disponibili in rete, utilizzano CT o radiografie di persone *già* in ospedale per un motivo o per l' altro e questo è ancora più siginificativo per quanto riguarda le immagini di pazienti positivi. Esse infatti venendo ricavate da pazienti ricoverati sono di per sè fortemente condizionate e pertanto inapplicabili in un contesto di screening generale della popolazione. Infine, vi è anche il problema costituito dal fatto che queste reti neurali vengono considerate delle

black box relativamente inaffidabili dato che, avendo in taluni casi miliardi di parametri, è difficile dire con esattezza in base a che cosa è stato prodotto un certo output. Questo problema verrà approfondito nell' ultimo capitolo.

Capitolo 5

Case study: diagnosi di covid da radiografie

5.1 Metodo

In questo lavoro di tesi è stato assunto come modello metodologico quello più usato in letteratura e, in generale, nei task di computer vision. Esso prevede innanzitutto la scelta di un dataset riproducibile e pubblicamente disponibile. Questo è fondamentale in quanto se il dataset utilizzato nell' esperimento non dovesse avere queste caratteristiche l' intero esperimento non avrebbe alcuna valenza scientifica. Dopo aver selezionato, o costruito, il dataset si procede a una fase detta *data preprocessing* che consiste nell' effettuare operazioni sui dati in modo da renderli fruibili al modello che si vuole utilizzare, ovvero ResNet in questo caso. A questo punto si è pronti a separare il dataset in tre sottoinsiemi, *train*, *validation* e *test* che verranno usati uno per la fase di training della rete, uno per cercare la migliore regolazione degli iperparametri e l' ultimo, ovvero test, per la conferma dei risultati. Sebbene esistano varie e più raffinate tecniche di separazione dei dati (i.e. n-fold cross etc.) in questo lavoro ci si è limitati alla suddivisione descritta sopra poichè interessati principalmente all' apprendimento della rete convoluzionale in sé e ad un indagine sulla effettiva capacità di questi

strumenti in task del genere, cioè la diagnosi medica. Ultimate le operazioni sui dati si comincia a progettare la rete vera e propria. In questa tesi verranno prese in considerazione due reti: una basata su ResNet50 e l' altra costruita *from scratch*, le quali verranno poi confrontate con le opportune metriche. Il framework di riferimento è PyTorch per entrambe le architetture. Infine, si comparano i risultati ottenuti con quelli presenti in letteratura.

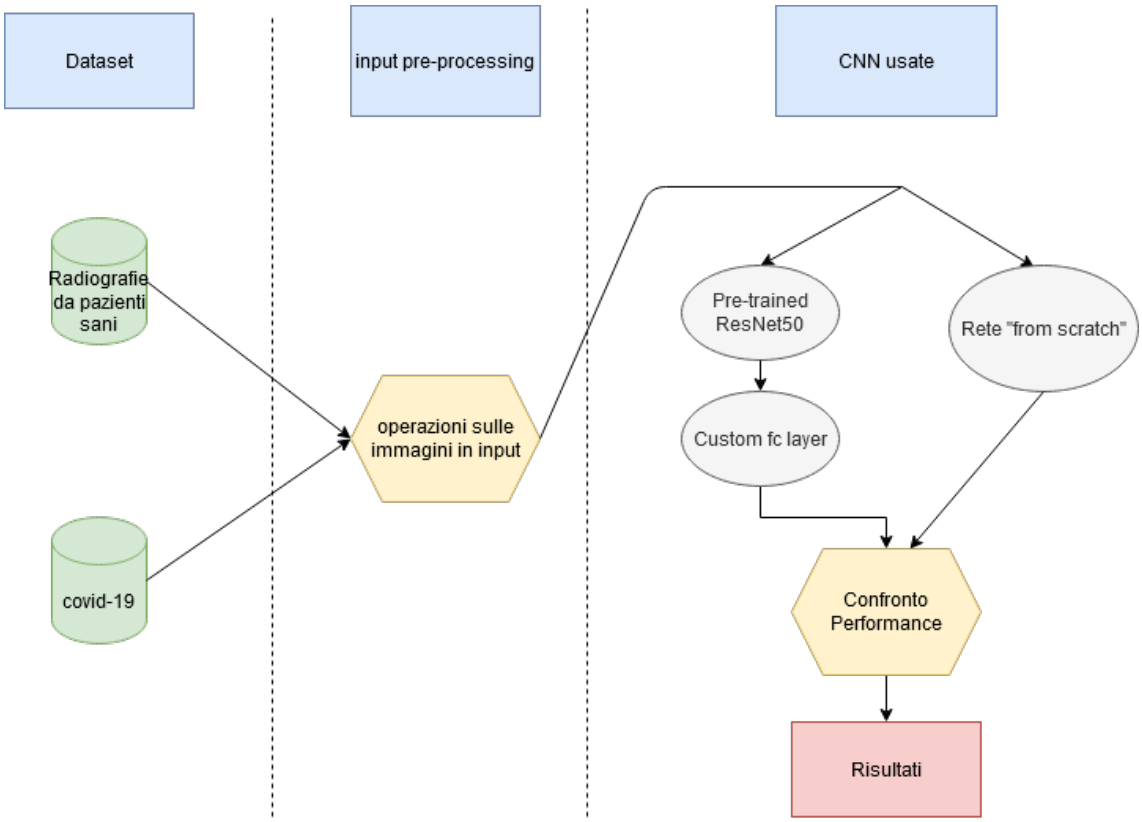


Figura 5.1: pipeline delle operazioni

5.1.1 Dataset e split

Seguendo quanto fatto in [7] il dataset utilizzato è stato QaTa-COV19 Dataset, (reperibile presso https://www.kaggle.com/aysendegerli/qatacov19-dataset) il quale è il più esteso dataset reperibile in rete. Esso consta infatti di 4603

radiografie di soggetti positivi al Covid19 e di molte migliaia di radiografie "non positive", a loro volta suddivise in due *control groups*, uno composto da radiografie effettuate a pazienti sani e l'altro da quelle effettuate a pazienti affetti da altre patologie polmonari. Per evitare problemi di *class imbalance* (i.e. una sproporzione rilevante fra le due classi oggetto dell' analisi) sono state scelte in modo casuale lo stesso numero di immagini prese dal primo Control Group (pazienti sani) in modo tale da operare su un dataset finale di circa 9200 radiografie. Per quanto riguarda lo split il dataset è stato diviso, secondo l' uso corrente, in train (80%), val (10%) e test (10%). Dove train e val sono stati utilizzati per l'allenamento e il tuning degli iperparametri sulle diverse reti provate mentre test è stato utilizzato esclusivamente nella fase finale di analisi della performance.

5.1.2 Data preprocessing

Per rendere le radiografie in input più effettive e fruibili dal modello sono state eseguite delle operazioni di processamento sui dati. Lo scopo di questa fase è quello di preparare le immagini a essere usate come input per la rete e per far ciò sono necessarie alcune elaborazioni tipiche. In questa tesi dette operazioni sono eseguite in modo sequenziale utilizzando semplici funzioni di libreria:

```
#Per ogni input x ->
     transforms.Compose([
        transforms.Resize(256),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ])
```

Dove la prima operazione non fa altro che ridimensionare l' input a 256x256 pixel e la seconda ritaglia il quadrato centrale di 224x224 pixel (misura necessaria per l' utilizzo di ResNet). La terza è una conversione in *tensore*,

la struttura dati d'elezione per compiere le operazioni tipiche di una rete neurale, mentre l' ultima si occupa di normalizzare l' immagine utilizzando come primo parametro la media e come secondo la deviazione standard.

5.1.3 Training

Dopo aver terminato le fasi preliminari di data processing giungiamo alla definizione dell' algoritmo di training della rete. Esso è una parte fondamentale dell' intera architettura anche se il procedimento è pressochè standard per quanto riguarda le reti convoluzionali.

```
for epoch in range(num_epochs):
   # Ogni epoca ha una fase di validazione e di training
   for phase in ['train', 'val']:
       if phase == 'train':
           model.train()  # setta il modello in modalita' training
       else:
           model.eval()   # setta il modello in validazione

       # Itera sui dati
       #e li sposta sul 'device', tipicamente una gpu, preferita alla cpu
       #in quanto capace di effettuare operazioni floating point in parallelo
       for inputs, labels in dataloaders[phase]:
           inputs = inputs.to(device)
           labels = labels.to(device)

           # azzera i gradienti ad ogni batch
           optimizer.zero_grad()

           # forward pass: effettua predizioni e calcola la loss
           with torch.set_grad_enabled(phase == 'train'):
               outputs = model(inputs)
```

```
                _, preds = torch.max(outputs, 1)
                loss = criterion(outputs, labels)

                # backward propagation solo in fase di training
                if phase == 'train':
                    loss.backward()
                    optimizer.step()

    if phase == 'train':
        scheduler.step()
```

Come è possibile vedere l' allenamento della rete è implementabile in pochissime istruzioni grazie ai diffusissimi framework (generalmente open-source) presenti.

5.2 Implementazione basata su Resnet50

L'architettura chiamata ResNet è stata già introdotta da un punto di vista storico dunque ripetiamo solamente che essa viene considerata una delle reti più performanti e pertanto è quella più comunemente usata [18]. Utilizzando pytorch per istanziare una nuova ResNet basta una semplice istruzione:

```
(from torchvision import models)
modello = models.resnet50(pretrained=True)
```

A questo punto si ha una rete pre-allenata con il dataset *ImageNet* e di conseguenza il layer completamente connesso è basato su questo dataset, sia come "capacità" di giudizio" (i.e. pesi e bias) sia come morfologia e cioè il numero di neuroni che esso contiene. ImageNet infatti contiene immagini prese da un insieme di 1000 classi, quindi il layer finale pre-trained avrà 1000 neuroni/output.

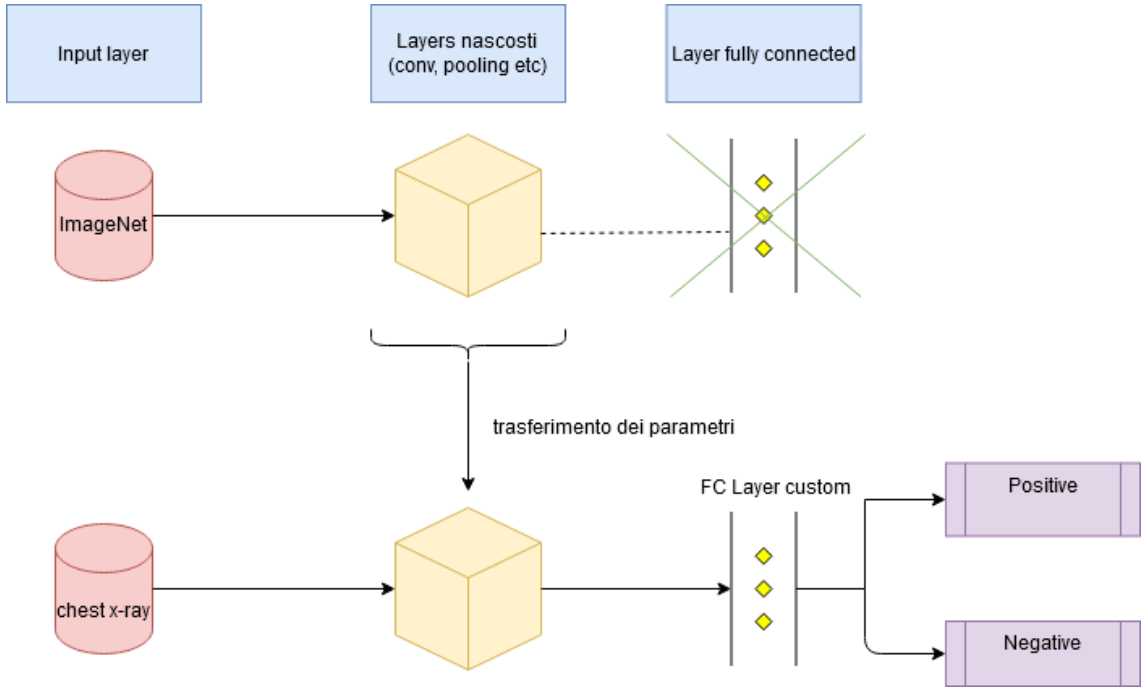


Figura 5.2: transfer-learning

È necessario quindi modificare questo layer in modo da adattarlo al task in esame, in questo caso trattandosi di classificazione binaria si è scelto di strutturarlo in questo modo:

```
modello.fc = nn.Sequential(
nn.Dropout(0.1),
nn.Linear(num_ftrs, 1024),
nn.ReLU(),
nn.Dropout(0.1),
nn.Linear(1024, 512),
nn.ReLU(),
nn.Dropout(0.1),
nn.Linear(512,2)
)
```

dove num_ftrs è uguale alla dimensione del vettore in input (2048 in questo caso). Si ricorda infatti che le operazioni tipiche di una rete convoluzionale modificano la struttura dell' input trasformandolo da immagine (e.g. 224x224x3) in vettore/array unidimensionale. A questo punto, scelta una Loss Function e un ottimizzatore per l' algoritmo di discesa del gradiente (in questo caso Adam), la rete è pronta per essere utilizzata.

5.3 Implementazione from scratch

Grazie ai framework disponibili anche implementare una rete from scratch non è un compito particolarmente arduo. Creiamo la classe in questo modo:

```
class ScratchCNN(nn.Module):
def __init__(self):
    super(ScratchCNN, self).__init__()

    self.model = nn.Sequential(
        nn.Conv2d(3, 16, kernel_size=3),
        nn.ReLU(),
        nn.Conv2d(16, 16, kernel_size=3),
        nn.ReLU(),
        nn.MaxPool2d(2,2),

        nn.Conv2d(16, 32, kernel_size=3),
        nn.ReLU(),
        nn.Conv2d(32, 32, kernel_size=3),
        nn.ReLU(),
        nn.MaxPool2d(2,2),

        nn.Conv2d(32, 64, kernel_size=3),
        nn.ReLU(),
        nn.Conv2d(64, 64, kernel_size=3),
```

```
            nn.ReLU(),
            nn.MaxPool2d(2,2),

            nn.Conv2d(64, 128, kernel_size=3),
            nn.ReLU(),
            nn.Conv2d(128, 128, kernel_size=3),
            nn.ReLU(),
            nn.MaxPool2d(2,2),

            nn.Conv2d(128, 256, kernel_size=3),
            nn.ReLU(),
            nn.Conv2d(256, 256, kernel_size=3),
            nn.ReLU(),
            nn.MaxPool2d(2,2),

        ).to(device)

        self.classifier = nn.Sequential(
            nn.Flatten(),
            nn.Dropout(0.25),
            nn.Linear(4096, 256),
            nn.ReLU(),
            nn.Dropout(0.25),
            nn.Linear(256, 2)
        ).to(device)

    def forward(self, x):
        x = self.model(x)
        x = self.classifier(x)
        return x
```

dove, seguendo la già vista formula

$INPUT- > [[CONV- > RELU] * N- > POOL?] * M- > [FC- > RELU] * K- > FC$

realizziamo la rete utilizzando 5 blocchi identici (cambiando solo le dimensioni in modo opportuno) ognuno costituito dalla successione: Convoluzione, Relu, Convoluzione, Relu, Pooling. Infine viene aggiunti il layer denso similmente a quanto fatto precedentemente. Possiamo vedere un sommario grazie al comando *summary(model)*

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 16, 254, 254]             448
              ReLU-2         [-1, 16, 254, 254]               0
            Conv2d-3         [-1, 16, 252, 252]           2,320
              ReLU-4         [-1, 16, 252, 252]               0
         MaxPool2d-5         [-1, 16, 126, 126]               0
            Conv2d-6         [-1, 32, 124, 124]           4,640
              ReLU-7         [-1, 32, 124, 124]               0
            Conv2d-8         [-1, 32, 122, 122]           9,248
              ReLU-9         [-1, 32, 122, 122]               0
        MaxPool2d-10           [-1, 32, 61, 61]               0
           Conv2d-11           [-1, 64, 59, 59]          18,496
             ReLU-12           [-1, 64, 59, 59]               0
           Conv2d-13           [-1, 64, 57, 57]          36,928
             ReLU-14           [-1, 64, 57, 57]               0
        MaxPool2d-15           [-1, 64, 28, 28]               0
           Conv2d-16          [-1, 128, 26, 26]          73,856
             ReLU-17          [-1, 128, 26, 26]               0
           Conv2d-18          [-1, 128, 24, 24]         147,584
             ReLU-19          [-1, 128, 24, 24]               0
        MaxPool2d-20          [-1, 128, 12, 12]               0
           Conv2d-21          [-1, 256, 10, 10]         295,168
             ReLU-22          [-1, 256, 10, 10]               0
           Conv2d-23           [-1, 256, 8, 8]          590,080
             ReLU-24           [-1, 256, 8, 8]               0
        MaxPool2d-25           [-1, 256, 4, 4]               0
          Flatten-26                [-1, 4096]               0
          Dropout-27                [-1, 4096]               0
           Linear-28                 [-1, 256]       1,048,832
             ReLU-29                 [-1, 256]               0
          Dropout-30                 [-1, 256]               0
           Linear-31                   [-1, 2]             514
================================================================
Total params: 2,228,114
Trainable params: 2,228,114
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.75
Forward/backward pass size (MB): 59.16
Params size (MB): 8.50
Estimated Total Size (MB): 68.41
----------------------------------------------------------------
```

Figura 5.3: output di summary(model)

Il metodo *forward(self,x)* va invece a definire come suggerito dal nome la forward function ovvero la funzione che si occupa di mappare il tensore in input in quello di ouput.

5.4 Iperparametri

Essendo una caratteristica fondamentale e determinante di una rete neurale convoluzionale, è necessario riportare gli iperparametri utilizzati per l' esperimento proposto. Per quanto riguarda la rete basata su ResNet è stata usata come loss function la funzione Binary Cross Entropy Loss già descritta precedentemente, batch size = 16, mentre come optimizer è stato scelto Adam [31] con learning rate pari a 0.00001. La rete è stata infine allenata per 10 epoche. Invece, per la rete costruita from scratch, laddove sono rimasti invariati la loss function, il batch size e l' optimizer, la rete è stata allenata per 50 epoche.

5.5 Comparazione Risultati

Per comparare i risultati delle due reti descritte in questo lavoro ci si è limitati all'utilizzo di una *confusion matrix*. Sebbene esistano altre metriche più usate e (magari) indicative, ottenibili con semplici funzioni di libreria, si è preferito l' utilizzo della confusion matrix per la sua immediatezza di comprensione anche per i non esperti del settore. Nel campo del machine learning e specialmente dell' apprendimento supervisionato si definisce *confusion matrix* una specifica tabella che permette la visualizzazione delle performance di un algoritmo [25]. In questo caso, trattandosi di classificazione binaria, la matrice ha dimensioni 2×2 e sulle quattro celle sono riportati i risultati per quanto riguarda: veri positivi (true positive, TP), veri negativi (true negative, TN), falsi positivi (FP) e falsi negativi (FN).

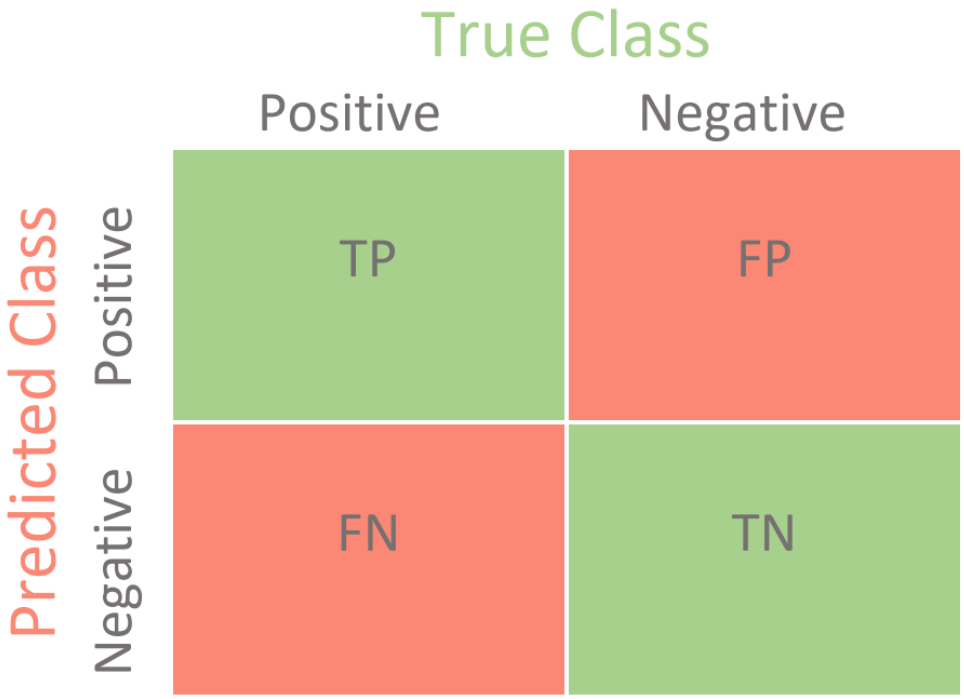


Figura 5.4: struttura della confusion matrix

Dopo aver terminato la fase di training delle due reti artificiali si fornisce loro come input, per la prima volta, il sottoinsieme di test e dai risultati ottenuti si calcolano le rispettive confusion matrices.

È evidente che sebbene siano una metrica piuttosto semplice la loro comparazione riesce a dare un' idea intuitiva ma chiara delle performance dei due modelli. Una volta calcolate le matrici, andiamo a definire le metriche come segue:

- **accuracy**: $\frac{TP+TN}{TP+TN+FP+FN}$
- **precision**: $\frac{TP}{TP+FP}$
- **recall**: $\frac{TP}{TP+FN}$

5.5.1 Custom ResNet50

La rete basata su ResNet50, utilizzando transfer learning su ImageNet, raggiunge un' accuracy pari a 99.8% , recall del 99,6% e precision del 100%:

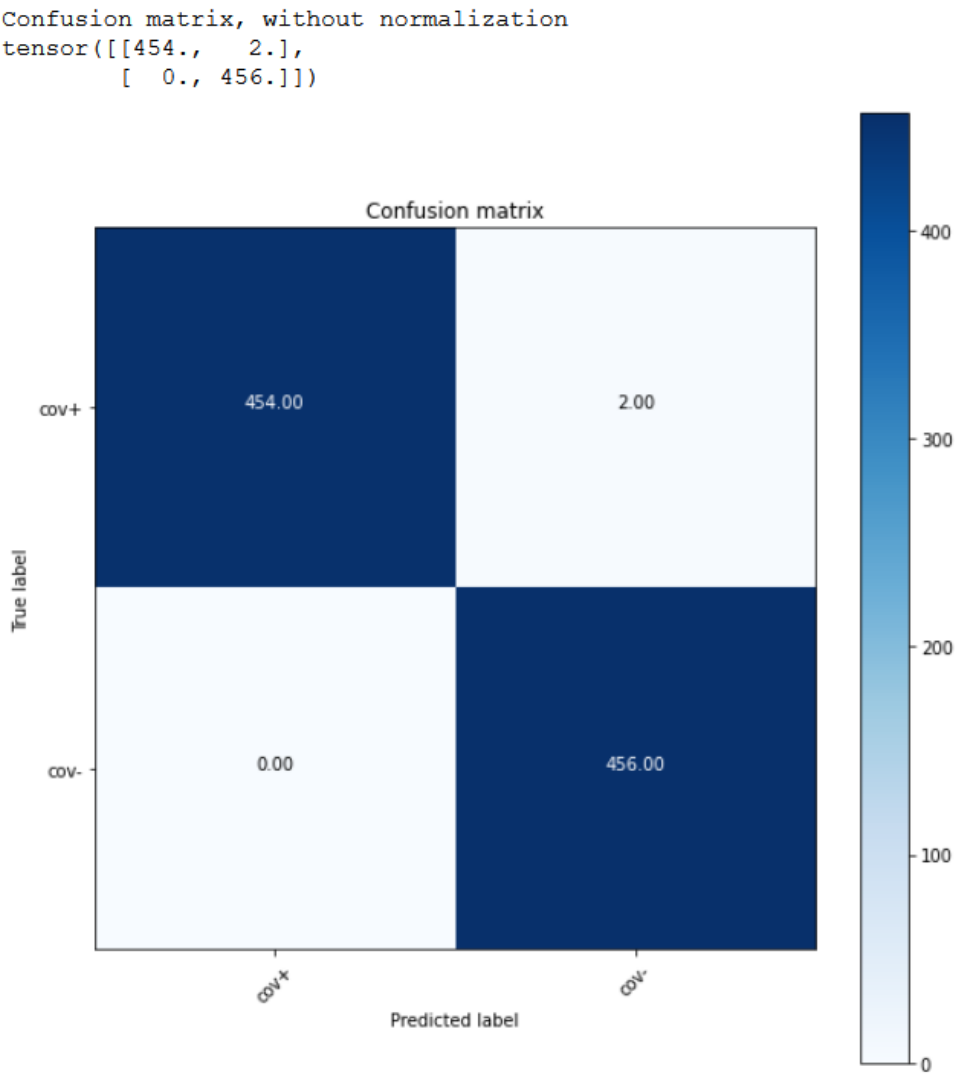


Figura 5.5: confusion matrix per la rete basata su ResNet50

5.5.2 Scratch Convolutional Neural Net

La rete costruita from scratch invece, senza utilizzare transfer learning, raggiunge un' accuracy pari a 93% , precision 94.3% e recall 91.4% .

```
Confusion matrix, without normalization
tensor([[417.,  39.],
        [ 25., 431.]])
```

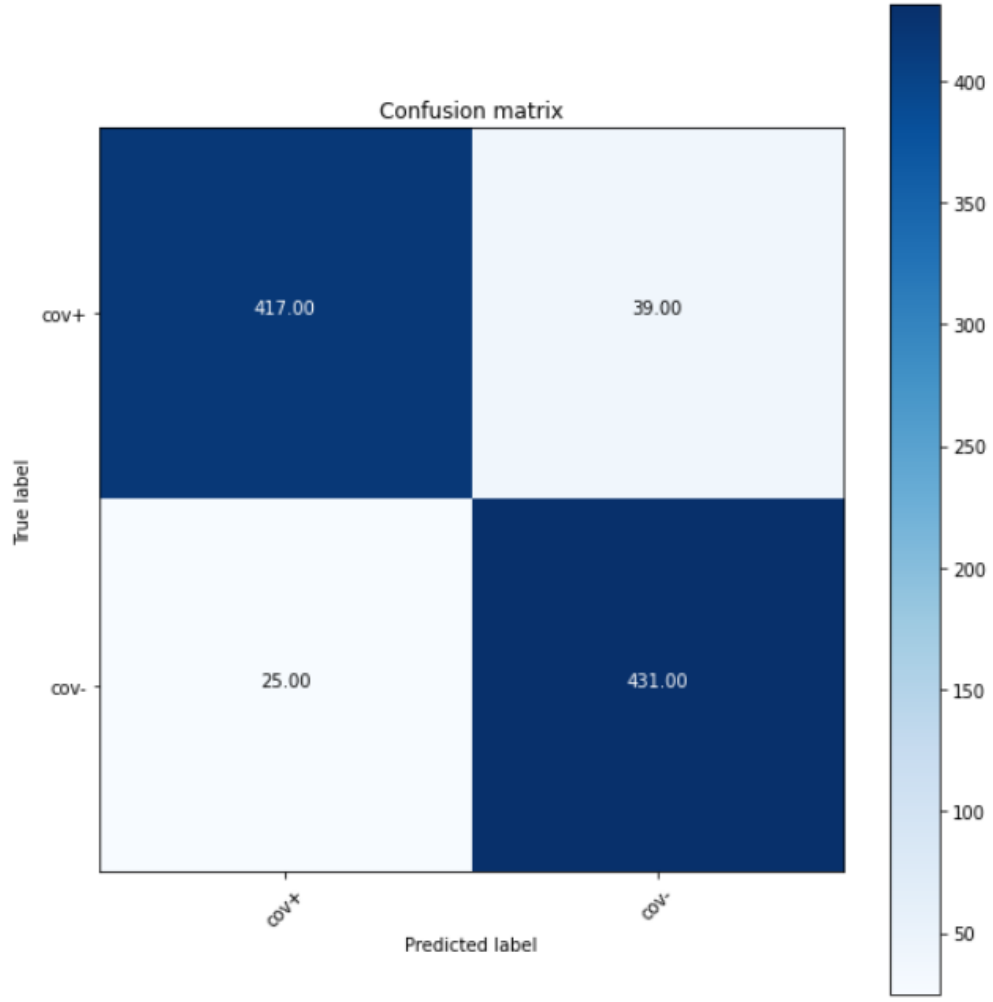


Figura 5.6: confusion matrix per la rete from scratch

Da questi risultati, in linea con quelli presenti in letteratura [4-7, 17], si evince chiaramente la straordinaria capacità di questi modelli di classificare

correttamente l'input. È altresì evidente che una rete semplice composta da ben pochi layer, come quella presentata in questo lavoro, sia comunque capace di giungere a percentuali molto alte. Ricordiamo infatti che per molteplici ragioni una rete *semplice* è preferibile a una uguale ma più complessa.

Capitolo 6

Black box problem ed Explainability

Andando a riprendere alcune delle problematiche evidenziate nel capitolo 4, si prova ora a dare una spiegazione sul funzionamento di questi modelli. Esistono diverse tecniche e diversi approcci per tale compito, in questo lavoro si esamina dapprima *Grad-CAM* [27], per poi fare una breve panoramica sulla *semantic segmentation.*

6.1 Grad-CAM

Come descritto dagli autori in [27] Grad-CAM è una tecnica per dare una "spiegazione visiva" delle decisioni prese da una rete neurale convoluzionale. L' acronimo sta per Gradient-weighted Class Activation Mapping e, come suggerito, usa i gradienti che dall' input "scorrono" fino al layer fully connected per produrre una grossolana mappa localizzante le regioni che più hanno influito sulla classificazione. Sebbene sia una tecnica elegante e adatta a molti tipi di problemi, per il task in esame in questo lavoro si rivela, come dimostrato da [7] e altri, inaccurata e piuttosto fallimentare. Per alcuni input infatti l' *activation map* prodotta da Grad-CAM è estremamente accurata mentre per altri (la maggior parte) essa è totalmente errata indicando aree

esterne ai polmoni o comunque notevolmente discordanti dai ground thruths. Prendendo come esempio alcune immagini annotate da radiologi, possiamo vedere come in alcuni casi le activation map siano molto accurate:

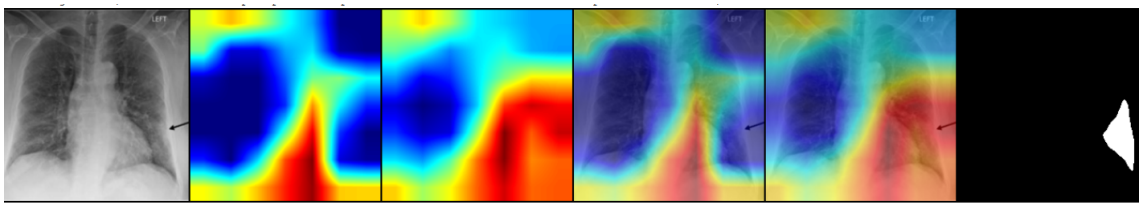


Figura 6.1: esempio di heatmap corretta, a destra ground truth dell' infezione

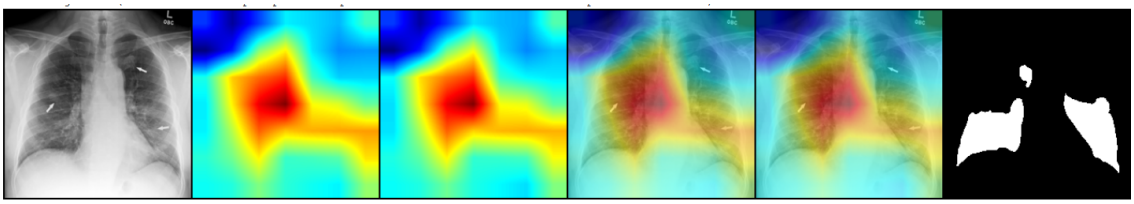


Figura 6.2: altro esempio di heatmap approssimativamente corretta

Mentre in altri esempi, sempre scelti fra le immagini annotate, è evidente la performance inaffidabile di questa tecnica:

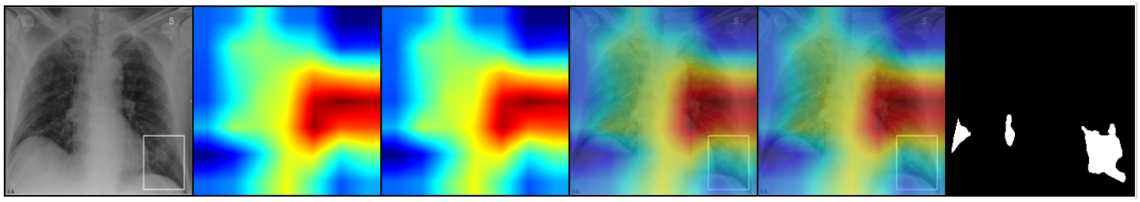


Figura 6.3: esempio di heatmap sbagliata

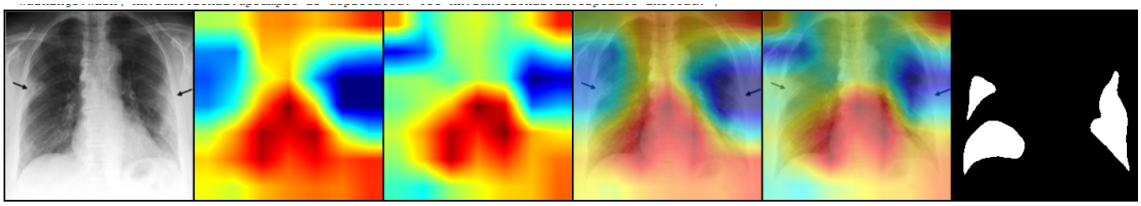


Figura 6.4: altro esempio di heatmap scorretta

6.2 Semantic Segmentation

La tecnica nota come semantic segmentation, ricordiamo, consiste nel classificare ogni pixel dell' immagine in input come appartenente ad una certa classe, può dunque essere vista come una problema di classificazione *pixel-wise*. Esistono numerose architetture nate per affrontare questo task ma la più nota è sicuramente UNET [28] e anche nelle prove sperimentali effettuate in questo lavoro verrà usata un' architettura basata su di essa. In diversi articoli viene dimostrata la capacità di questa rete di effettuare segmentazioni corrette partendo da tomografie computerizzate ma l' analisi di radiografie rimane un' area ancora da esplorare e sono ancora molto pochi i lavori pubblicati a riguardo. A titolo di esempio viene riportata un' immagine presa dai risultati di [29] che ben evidenzia i risultati sinora ottenuti in questo campo:

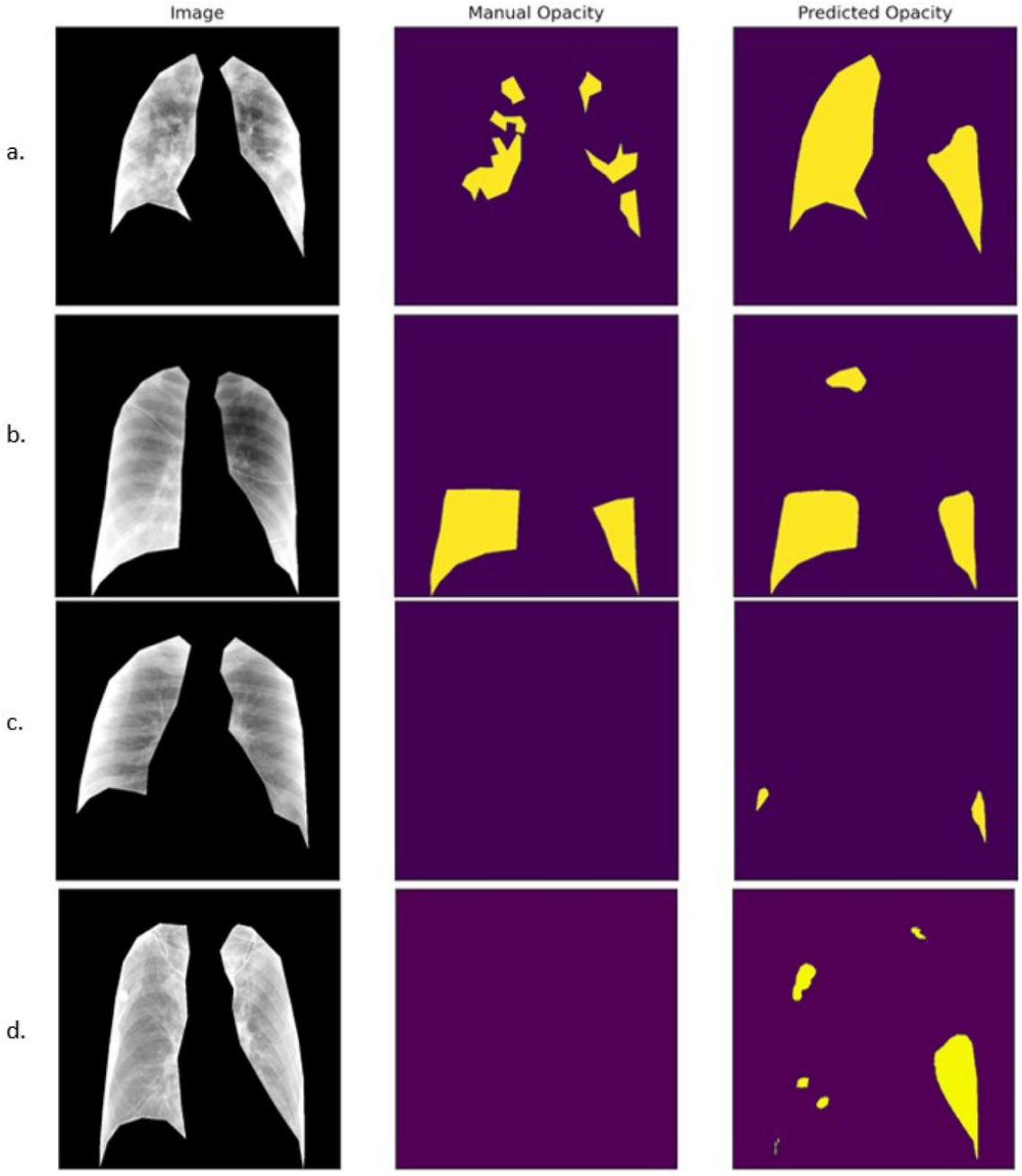


Figura 6.5: alcuni risultati di [29]

Mentre per quanto riguarda le prove sperimentali effettuate in questo lavoro come già detto precedentemente è stata usata una architettura basata

su UNet la quale è stata trainata per 30 epoche con 2360 coppie (radiografia - ground truth), mentre sono state usate per la fase di validation 591 di queste coppie. La Loss Function usata è stata la Dice Loss con learning rate di $0,0001$ e batch-size pari a 16. Vengono qui riportati alcune immagini prese dai risultati ottenuti:

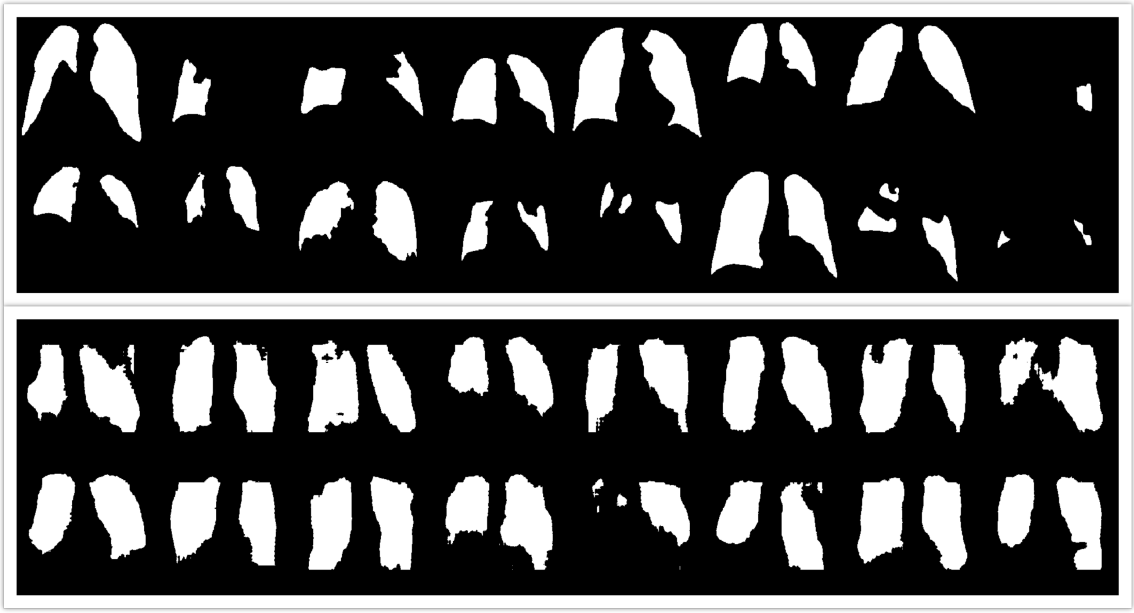


Figura 6.6: in alto ground truth per un batch random di validation, in basso le predizioni del moello

Senza entrare nei dettagli circa l' utilizzo della Dice Loss, i risultati ottenuti vengono generalmente ritenuti "buoni" con un *dice score* > 0.5 mentre quello qui ottenuto è di 0.43.

Seguendo quanto riportato in [29] e dai risultati prodotti in questa tesi, si evidenzia che questo task è estremamente più complesso della *semplice* classificazione, infatti i contorni delle regioni aventi opacità sono molto difficili da identificare chiaramente dalle radiografie e ciò può essere dovuto a un gran numero di problemi come la qualità dell' immagine, l' anatomia del paziente, lo stato della malattia etc. In secondo luogo, alcune strutture anatomiche

potrebbero compromettere le predizioni del modello: ad esempio una sovrapposizione di tessuti polmonari potrebbe facilmente essere scambiata per un' opacità o un ispessimento della membrana causato da infezioni.

Da queste prove sperimentali si può dedurre, come evidenziato in [26], che è facile incappare in qualche forma di bias lavorando con questi strumenti e pertanto è necessaria grande attenzione nonchè conoscenza di questi problemi durante la preparazione degli esperimenti. Tuttavia, pur non raggiungendo i risultati auspicati, queste tecniche di visualizzazione possono essere un valido strumento di aiuto per demistificare i risultati ottenuti.

Conclusioni

L' argomento trattato in questa tesi è estremamente vasto e comprende una molteplicità di discipline anche molto diverse fra loro. Si è cercato di riassumere al meglio lo sforzo duale (teorico e pratico) stante dietro a questo lavoro cominciando da una panoramica storica che possa far comprendere l' importanza di questi argomenti al di là del loro essere di moda in questi anni. Infatti, se da un lato i risultati hanno un che di miracoloso, è bene tenere presente la vastità nonché la complessità della teoria che fa loro da fondamenta. È stato inoltre ritenuto fondamentale, come evidenziato nell' ultimo capitolo, riportarne i limiti (attuali) e porre un significativo accento sulla possibilità di spiegare i risultati ottenuti. Il focus di questa tesi, tuttavia, non è e non poteva essere il campo della *semantic segmentation* la quale è un'area ai confini della ricerca scientifica e dove molto lavoro viene fatto ogni giorno, ma anzi l' obiettivo è stato quello di dimostrare, in modo quanto più formale possibile per quanto riguarda la teoria e corredando il tutto con le prove sperimentali, che queste nuove tecnologie, anzi veri e propri paradigmi, possono senz' altro costituire un elemento fortemente innovativo in moltissimi campi come appunto quello della diagnosi medica. Le reti neurali convoluzionali infatti sono uno strumento alquanto versatile e dalle grandi potenzialità, la matematica alle fondamenta è solida e ormai consolidata nel tempo dalle numerose sfide vinte ma bisogna però utilizzarle con cautela e con la giusta attenzione, con le giuste metriche e con un dataset affidabile e comprovato. Perchè laddove questi accorgimenti vengano dimenticati, come viene spesso ripetuto nel mondo dell' informatica, è facile scambiare le *cnn*

per un martello e allora tutto può sembrare un chiodo, ma la realtà mal si presta a queste banalizzazioni e la sua complessità non è riducibile a *una* formula, per quanto complessa e articolata questa possa essere. In altre parole si potrebbe dire che anche se il Deep Learning, è bene ripeterlo, ha capacità straordinarie, citando Fred Brooks "There's no Silver Bullet".

Tutto il codice utilizzato in questa tesi, insieme a molte altre prove sperimentali effettuate ma non riportate, è disponibile al seguente indirizzo: https://github.com/Orasz/CNN4COVID19.

Bibliografia

[1] World Health Organization, Coronavirus disease 2019 (covid-19): situation report, 88, 2020

[2] https://www.iss.it consultato il 10/02/2021

[3] Y. Li, L. Yao, J. Li, L. Chen, Y. Song, Z. Cai, and C. Yang, âStability issues of rt-pcr testing of sars-cov-2 for hospitalized patients clinically diagnosed with covid-19,âJournal of medical virology, 2020

[4] de Sousa, P.M., Carneiro, P.C., Oliveira, M.M. et al. COVID-19 classification in X-ray chest images using a new convolutional neural network: CNN-COVID. Res. Biomed. Eng. (2021).

[5] Gonzales, Bustos et al. UMLS-ChestNet: A deep convolutional neural network for radiological findings, differential diagnoses and localizations of COVID-19 in chestx-rays. https://arxiv.org/pdf/2006.05274.pdf

[6] Duran-Lopez, L.; Dominguez-Morales, J.P.; Corral-Jaime, J.; Vicente-Diaz, S.; Linares-Barranco, A. COVID-XNet: A Custom Deep Learning System to Diagnose and Locate COVID-19 in Chest X-ray Images. Appl. Sci. 2020, 10, 5683. https://doi.org/10.3390/app10165683

[7] Degerli, Aishali et al. COVID-19 Infection Map Generation and Detection from Chest X-Ray Images. https://arxiv.org/pdf/2009.12698.pdf

[8] W. J. Zhang, G. Yang, Y. Lin, C. Ji and M. M. Gupta, "On Definition of Deep Learning," 2018 World Automation Congress (WAC), Stevenson, WA, 2018, pp. 1-5, doi: 10.23919/WAC.2018.8430387.

[9] Rumelhart, D., Hinton, G. Williams, R. Learning representations by back-propagating errors. Nature 323, 533â536 (1986). https://doi.org/10.1038/323533a

[10] Ruder s. An overview of gradient descent optimization algorithms.https://arxiv.org/pdf/1609.04747.pdf

[11] Andrew Ng. Lecture notes from 'machine learning' course on coursera.org.

[12] Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. (1998). "Gradient-based learning applied to document recognition" (PDF). Proceedings of the IEEE. 86 (11): 2278â2324. doi:10.1109/5.726791.

[13] Srivastasa N., Hinton J. et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. https://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf

[14] Roberts, Lawrence. (1963). Machine Perception of Three-Dimensional Solids.

[15] Fukushima, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biol. Cybernetics 36, 193â202 (1980). https://doi.org/10.1007/BF00344251

[16] Hubel DH, Wiesel TN. Receptive fields and functional architecture of monkey striate cortex. J Physiol. 1968;195(1):215-243. doi:10.1113/jphysiol.1968.sp008455

[17] Ali Narin, Ceren Kaya, Ziynet Pamuk, Automatic Detection of Coronavirus Disease (COVID-19) Using X-ray Images and Deep Convolutional Neural Networks, https://arxiv.org/abs/2003.10849

[18] Lecture notes from Stanford CS231n: Convolutional Neural Networks for Visual Recognition.

[19] He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing; Sun, Jian (2015-12-10). "Deep Residual Learning for Image Recognition". [arXiv](https://en.wikipedia.org/wiki/ArXiv_(identifier)):[1512.03385](https://arxiv.org/abs/1

[20] https://lukeoakdenrayner.wordpress.com/2020/03/23/ct-scanning-is-just-awful-for-diagnosing-covid-19/Â consultato il 18/02/2021

[21] https://car.ca/news/canadian-society-of-thoracic-radiology-and-canadian-association-of-radiologists-statement-on-covid-19 consultato il 18/02/2021

[22] https://www.acr.org/Advocacy-and-Economics/ACR-Position-Statements/Recommendations-for-Chest-Radiography-and-CT-for-Suspected-COVID19-Infection consultato il 18/02/2021

[23] https://www.rcr.ac.uk/college/coronavirus-covid-19-what-rcr-doing/clinical-information/role-ct-chest/role-ct-patientsÂ consultato il 18/02/2021

[24] Tao Ai, Zhenlu Yang, Hongyan Hou, Chenao Zhan, Chong Chen, Wenzhi Lv, Qian Tao, Ziyong Sun, and Liming Xia Correlation of Chest CT and RT-PCR Testing for Coronavirus Disease 2019 (COVID-19) in China: A Report of 1014 Cases

[25] https://en.wikipedia.org/wiki/Confusion_matrix

[26] Maguolo G., Nanni L., A Critic Evaluation of Methods for COVID-19 Automatic Detection from X-Ray Images, https://arxiv.org/abs/2004.12823

[27] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization," 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 2017, pp. 618-626, doi: 10.1109/ICCV.2017.74.

[28] Ronneberger, Olaf; Fischer, Philipp; Brox, Thomas (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation". arXiv:1505.04597 [cs.CV].

[29] Segmentation model of the opacity regions in the chest X-rays of the Covid-19 patients in the us rural areas and the application to the disease severity Haiming Tang, Nanfei Sun, Yi Li medRxiv 2020.10.19.20215483; doi: https://doi.org/10.1101/2020.10.19.20215483

[30] https://it.wikipedia.org/wiki/Apprendimento_profondo

[31] Diederik P. Kingma, Jimmy Ba, Adam: A Method for Stochastic Optimization, https://arxiv.org/abs/1412.6980

Ringraziamenti

Ringrazio mia madre, che ha compiuto molti più sforzi di me affinchè io potessi arrivare a scrivere queste righe. Ringrazio i miei amici che hanno camminato insieme a me in tutti questi anni, in particolare una menzione va a Francesco Codicè senza il quale starei ancora litigando con la guardia del ciclo *for*. Ringrazio i miei professori del liceo, ringrazio il dottor Zingaro che mi ha guidato e accompagnato in questo lavoro e infine ringrazio mio fratello, che sta combattendo una battaglia più dura di qualche esame.