



ALMA MATER STUDIORUM  
UNIVERSITÀ DI BOLOGNA

DIPARTIMENTO DI INGEGNERIA DELL'ENERGIA ELETTRICA E DELL'INFORMAZIONE "GUGLIELMO MARCONI" - DEI

CORSO DI LAUREA IN INGEGNERIA DELL'AUTOMAZIONE

---

**RETI NEURALI  
APPLICATE ALLA DIAGNOSI DELLA  
SCLEROSI MULTIPLA**

Tesi di laurea in Controlli Automatici T-1

**Relatore**

**Prof. Roberto Diversi**

**Presentata da**

**Sofia Benati**

---

Sessione Dicembre 2025

Anno Accademico 2024/2025

# Indice

<b>1 Introduzione</b>	<b>7</b>
1.1 Reti neurali artificiali . . . . .	8
1.1.1 La storia . . . . .	8
1.1.2 I neuroni a confronto . . . . .	9
1.1.3 Struttura generale di una rete neurale . . . . .	10
1.2 Sclerosi multipla . . . . .	11
1.2.1 I sintomi . . . . .	11
1.2.2 La diagnosi . . . . .	12
1.3 Stato dell'arte . . . . .	13
<b>2 Architettura delle reti neurali</b>	<b>14</b>
2.1 Un singolo neurone . . . . .	14
2.1.1 Funzione a gradino . . . . .	15
2.1.2 Funzione sigmoide . . . . .	15
2.1.3 Funzione tangente iperbolica . . . . .	16
2.1.4 Funzione ReLU . . . . .	16
2.1.5 Funzione softmax . . . . .	18
2.2 Uno strato di neuroni - Hidden Layer . . . . .	19
2.3 Rete neurale profonda - Multi Layer Perceptron . . . . .	19
2.4 Universalità delle reti neurali . . . . .	20
2.4.1 Universalità con un solo strato nascosto . . . . .	20
2.5 Rete neurale convoluzionale - CNN . . . . .	21
2.5.1 Architettura delle reti neurali convoluzionali . . . . .	22
2.5.2 Strato convoluzionale . . . . .	22
2.5.3 Strato di pooling . . . . .	23
2.5.4 AlexNet . . . . .	23
2.6 U-Net . . . . .	24
2.6.1 Architettura della U-Net . . . . .	25
<b>3 Allenamento delle reti neurali</b>	<b>26</b>
3.1 Funzione di costo e ottimizzazione . . . . .	26

3.1.1	Cross-entropy e Dice loss function . . . . .	26
3.1.2	Discesa del gradiente . . . . .	27
3.1.3	Backpropagation algorithm . . . . .	29
3.1.4	Suddivisione del dataset per il calcolo del gradiente . . . . .	30
3.1.5	Varianti della discesa del gradiente . . . . .	30
3.2	Parametri e iperparametri . . . . .	32
3.3	Possibili problematiche dell’allenamento . . . . .	33
3.3.1	Overfitting e underfitting . . . . .	33
3.3.2	Internal covariate shift . . . . .	34
3.3.3	Vanishing gradient . . . . .	35
3.3.4	Scarsità di dati disponibili . . . . .	35
<b>4</b>	<b>Applicazione ad un caso reale</b>	<b>36</b>
4.1	Analisi e preparazione del dataset . . . . .	36
4.1.1	Struttura del dataset . . . . .	36
4.1.2	Preparazione del dataset . . . . .	38
4.2	Implementazione del modello . . . . .	38
4.2.1	Struttura della U-Net 2D . . . . .	38
4.3	Allenamento della U-Net . . . . .	40
4.3.1	Metriche utilizzate . . . . .	41
4.4	Validation e test . . . . .	42
4.4.1	Valutazione basata sulle metriche - Validation . . . . .	42
4.4.2	Valutazione basata sui plot - Test . . . . .	43
4.5	Studi rilevanti sulla segmentazione . . . . .	46
4.5.1	GAU U-Net . . . . .	46
4.5.2	A dense residual U-Net . . . . .	47
4.5.3	A modified U-Net attention . . . . .	48
4.5.4	Confronto con gli studi riportati . . . . .	50
<b>5</b>	<b>Conclusioni</b>	<b>52</b>

# Elenco delle figure

1.1	Confronto tra neurone biologico e neurone artificiale . . . . .	9
1.2	Rete neurale [8] . . . . .	10
1.3	Immagini di risonanze magnetiche, le lesioni visibili sono di colore bianco .	12
1.4	Stesse lesioni delle immagini precedenti, viste dal piano frontale . . . . .	12
2.1	Funzione a gradino[15] . . . . .	15
2.2	Funzione sigmoide[16] . . . . .	16
2.3	Funzione tangente iperbolica[17] . . . . .	16
2.4	Funzione ReLU[18] . . . . .	17
2.5	Funzione Leaky ReLU[19] . . . . .	18
2.6	Funzione Softmax[21] . . . . .	18
2.7	Approssimazione di una funzione[24] . . . . .	20
2.8	Approssimazione di una funzione con 2 input[25] . . . . .	20
2.9	Rete Neurale Convoluzionale [27] . . . . .	21
2.10	Rete Neurale Convoluzionale U-Net[32] . . . . .	24
3.1	Discesa del gradiente[34] . . . . .	28
3.2	Dimensione del passo[35] . . . . .	28
3.3	Underfitting e overfitting[38] . . . . .	34
4.1	Immagine di risonanza magnetica di tipo FLAIR del paziente n°6 con la rispettiva maschera. A sinistra è visibile l'immagine originale senza maschera applicata, in cui le zone più chiare del cervello corrispondono alle lesioni. Al centro è visibile la maschera che indica la posizione delle lesioni, a destra la sovrapposizione della maschera sull'immagine. . . . .	37
4.2	Immagine di risonanza magnetica di tipo T1 del paziente n°6 con la rispettiva maschera. A sinistra è mostrata l'immagine originale di tipo T1 senza maschera applicata, si nota il fatto che l'immagine sia più scura rispetto alla precedente e quindi le lesioni siano meno visibili, la struttura del plot al centro e a destra è identica alla precedente. . . . .	37

4.3	Risonanza magnetica di tipo T2 del paziente n°6 con la rispettiva maschera. Anche in questo caso la struttura è come nei casi precedenti, si nota la maggiore luminosità dell'immagine a sinistra, dovuta al segnale dei fluidi. .	37
4.4	Andamenti della funzione di costo Dice loss e del Dice coefficient . . . . .	42
4.5	Risonanza magnetica orientata sul piano sagittale . . . . .	43
4.6	Risonanza magnetica orientata sul piano trasversale . . . . .	43
4.7	. . . . .	44
4.8	. . . . .	45
4.9	GAU U-Net [43] . . . . .	46
4.10	Dense residual U-Net [44] . . . . .	48

# Elenco delle tabelle

4.1	Architettura del modello U-Net utilizzato . . . . .	40
4.2	Valori assunti dalle metriche durante il validation . . . . .	42
4.3	Confronto tra le prestazioni delle architetture utilizzate . . . . .	47
4.4	Valori del Dice per diversi modelli utilizzati . . . . .	48
4.5	Valori del Dice utilizzando la cross data validation . . . . .	48
4.6	Casi di studio . . . . .	49
4.7	Dice, Sensitivity e IoU per rete e tipo di dato. . . . .	49

# Capitolo 1

## Introduzione

Lo scopo di questa tesi è quello di utilizzare le reti neurali per la segmentazione delle lesioni cerebrali in pazienti affetti da sclerosi multipla.

In particolare, si intende impiegare reti neurali convoluzionali per l'elaborazione di immagini di risonanza magnetica (RM), con l'obiettivo di identificare e delimitare le aree del cervello in cui si manifestano le lesioni tipiche della patologia.

La segmentazione delle lesioni consente di ottenere una valutazione oggettiva dell'evoluzione della malattia, migliorando l'efficienza del processo diagnostico; infatti, permette di individuare anche lesioni di piccole dimensioni difficili da rilevare manualmente e di ridurre i tempi rispetto ad una segmentazione manuale.

Per affrontare in modo completo il tema, in questo capitolo verranno introdotte le basi teoriche delle reti neurali artificiali tradizionali (ANN) e verrà fornita una panoramica sulla sclerosi multipla. Nei capitoli successivi, si approfondirà la struttura, il funzionamento delle ANN e si passerà allo studio delle reti neurali convoluzionali (CNN) per trattare l'architettura delle reti U-Net, che verranno applicate ad un caso di studio reale.

## 1.1 Reti neurali artificiali

Una rete neurale artificiale (ANN) è un modello computazionale ispirato alla rete neurale biologica [1]. Nonostante è noto che il cervello sia molto più complesso di una ANN, queste sono le principali caratteristiche in comune:

- Apprendimento e adattamento
- Generalizzazione
- Parallelismo su larga scala
- Robustezza
- Memorizzazione associativa delle informazioni
- Elaborazione di informazioni spazio temporali

### 1.1.1 La storia

Il primo modello del neurone artificiale fu proposto da McCulloch e Pitts nel 1943 i quali descrissero un combinatore lineare a soglia in grado di elaborare più dati binari in entrata e di restituire un solo dato binario in uscita. Collegando vari di questi elementi si ottiene una rete in grado di calcolare semplici funzioni logiche e booleane.

Le prime ipotesi di apprendimento delle reti neurali furono introdotte da Donald Hebb nel 1949, secondo il quale la forza di connessione tra due neuroni aumenta quando essi si attivano simultaneamente.

Nel 1958 Frank Rosenblatt introdusse il perceptron, il primo modello di rete neurale per il riconoscimento e la classificazione delle forme. Inoltre, propose il primo modello di apprendimento supervisionato, decisamente innovativo rispetto al modello binario grazie alla possibilità di modificare i pesi sinaptici, rendendo così il perceptron in grado di apprendere.

I risultati ottenuti da Rosenblatt stimolarono una grande quantità di ricerche per un decennio, fino a quando Minsky e Papert non mostrarono i limiti operativi delle reti a due strati basate sul perceptron, in particolare l'incapacità del perceptron di risolvere problemi non linearmente separabili. Questo determinò un temporaneo declino dell'interesse verso le reti neurali.

Soltanto nel 1982 Jhon Hopfield propose un nuovo modello di rete neurale capace di rappresentare stati stabili e processi associativi, dando così nuovamente il via alle ricerche sulle reti neurali.

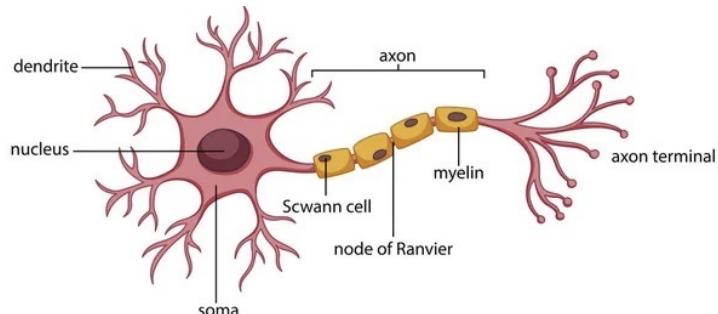
Pochi anni dopo, nel 1986, Rumelhart, Hinton e Williams introdussero il backpropagation error algorithm, che permise di superare i limiti del perceptron grazie all'addestramento efficace di reti multistrato [2].

### 1.1.2 I neuroni a confronto

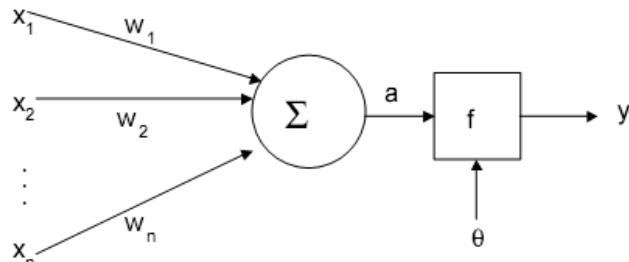
Il neurone biologico è l'unità del sistema nervoso in grado di ricevere, elaborare e trasmettere impulsi nervosi [3]. È costituito da tre parti: il soma, che è la parte centrale del neurone, i dendriti, che sono destinati alla recezione degli impulsi e l'assone, che si occupa della trasmissione. Inoltre è presente la sinapsi, che è il collegamento tra l'assone del neurone precedente e i dendriti del neurone successivo [4]. Quando il segnale raggiunge una soglia limite il neurone genera un segnale di output verso gli altri neuroni [5].

I neuroni artificiali, che sono una semplificazione del neurone biologico, possono essere così descritti [1]:

- Connessioni di input:  $x_1, x_2, \dots, x_n$  ad ognuna delle quali è associato un peso  $w_1, w_2, \dots, w_n$ . Un ingresso particolare è il bias, che ha un valore costante pari a 1
- Funzione di input  $a$ : calcola il segnale netto aggregato all'ingresso del neurone  $a = f(x, w)$ , tipicamente è  $a = \sum_{i=1}^n x_i \cdot w_i$
- Funzione di attivazione: calcola il livello di attivazione del neurone  $f = s(u)$
- Funzione di output: calcola il valore del segnale di uscita, di norma tale segnale viene assunto uguale al livello di attivazione del neurone



(a) Neurone biologico [6]



(b) Neurone artificiale [7]

Figura 1.1: Confronto tra neurone biologico e neurone artificiale

### 1.1.3 Struttura generale di una rete neurale

Le reti neurali si descrivono generalmente in base alle seguenti caratteristiche:

1. Tipo di neurone, che viene determinato dalla funzione di attivazione
2. L'architettura, che può essere descritta in base al numero di neuroni di ingresso e uscita e in base al numero di strati presenti:
  - (a) **Autoassociativa:** i neuroni di ingresso coincidono con i neuroni di uscita
  - (b) **Eteroassociativa:** i neuroni di ingresso e uscita sono distinti, come nella rete multistrato

Inoltre, in base alla presenza di connessioni in retroazione dai neuroni di uscita a quelli di ingresso si distinguono altri due tipi di architetture:

- (a) **Architettura feedforward:** non sono presenti retroazioni, perciò la rete non conserva memoria dei precedenti valori di uscita e degli stati di attivazione dei neuroni
  - (b) **Architettura feedback:** sono presenti retroazioni, perciò lo stato successivo della rete dipende sia dai segnali di ingresso correnti che dagli stati precedenti
3. Algoritmo di apprendimento: algoritmo che consente l'apprendimento della rete
  4. Algoritmo di recall: algoritmo mediante il quale le conoscenze apprese vengono richiamate dalla rete

Altri aspetti importanti che caratterizzano le reti neurali sono: **capacità di autoapprendimento**, ad esempio durante l'implementazione del riconoscimento di immagini la rete impara a riconoscere quelle simili tra loro, **memoria associativa**, le reti con architettura feedback sono in grado di ricordare un'informazione a partire da un segnale incompleto e infine **alta velocità di elaborazione**.

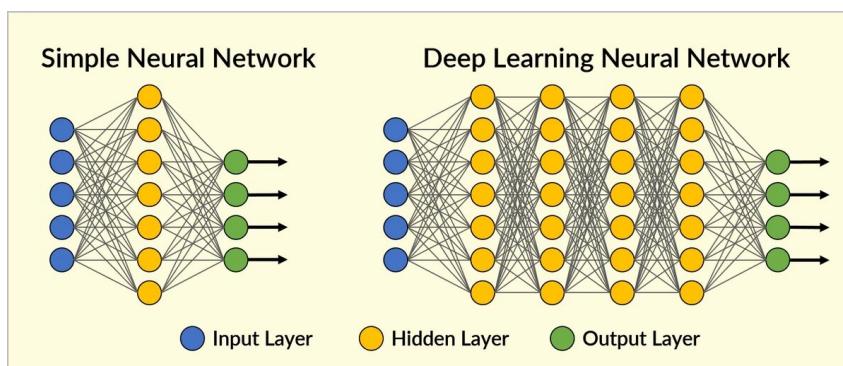


Figura 1.2: Rete neurale [8]

## 1.2 Sclerosi multipla

La sclerosi multipla è una malattia neurodegenerativa che colpisce il sistema nervoso centrale, caratterizzata da una reazione anomala delle difese immunitarie che attaccano alcuni componenti del sistema nervoso centrale scambiandoli per agenti estranei, per questo rientra tra le patologie autoimmuni. [9]

L'infiammazione scatenata dal sistema immunitario, può danneggiare sia la mielina, che è la guaina che ricopre e isola le fibre nervose, sia le cellule specializzate nella sua produzione, sia le fibre nervose stesse.

Questo processo, definito demielinizzazione, può provocare aree di perdita o lesione della mielina, che vengono definite placche o lesioni. Possono presentarsi ovunque nel sistema nervoso centrale, in particolare nei nervi ottici, cervelletto e midollo spinale.

Le lesioni possono evolvere da una fase infiammatoria iniziale a una fase cronica, in cui assumono caratteristiche simili a cicatrici, dette sclerosi. Nelle Figure 1.3 e 1.4 sono riportate alcune immagini di risonanze magnetiche dove sono visibili le lesioni.

### 1.2.1 I sintomi

I sintomi della sclerosi multipla si presentano in modo diverso da persona a persona, in base alla diversa localizzazione delle lesioni nel sistema nervoso, che non è prevedibile.

I sintomi più comuni sono:

- Fatica: mancanza di energia fisica o mentale rispetto alla norma in rapporto all'attività svolta
- Disturbi visivi: neurite ottica, sdoppiamento della vista
- Disturbi della sensibilità: calo della sensibilità, sensazioni alterate o dolorose
- Disturbi cognitivi: disturbi della concentrazione o della memoria, difficoltà a mantenere la concentrazione, sensazione di mente annebbiata o confusa
- Disturbi della cordinazione: alterata fluidità dei movimenti o disturbi dell'equilibrio
- Disturbi intestinali e vescicali
- Disturbi del linguaggio: debolezza e mancanza di cordinazione della lingua, della muscolatura orale e facciale
- Dolore: serie di sensazioni fisiche spiacevoli o lancinanti, passeggiere o croniche, quali bruciori, fitte acute, sofferenza muscoloscheletrica, sensazioni di tensione alla schiena, al petto o allo stomaco, dolore al volto

## 1.2.2 La diagnosi

Attualmente non esiste un singolo test per poter diagnosticare la sclerosi multipla, il neurologo considera tre elementi:

1. Storia clinica: informazioni raccolte durante il colloquio riguardanti la situazione clinica, i sintomi, i precedenti disturbi e tutte le informazioni che possono servire ad avere un quadro generale
2. Esame neurologico: visita effettuata dal neurologo per analizzare i movimenti, la reazione agli stimoli visivi e i riflessi, può prevedere diverse prove
3. Esami specifici strumentali e biologici: risonanza magnetica, potenziali evocati, esami del sangue e del liquido cerebrospinale.

Per poter confermare la diagnosi è necessario che le lesioni siano diffuse in diverse aree del sistema nervoso centrale.

Diagnosticare tempestivamente la sclerosi multipla, è fondamentale per poter agire prima che le condizioni della malattia si aggravino e garantire alle persone un'alta qualità di vita.

Di seguito vengono riportate alcune immagini di risonanze magnetiche, in cui sono visibili le lesioni dovute alla patologia.

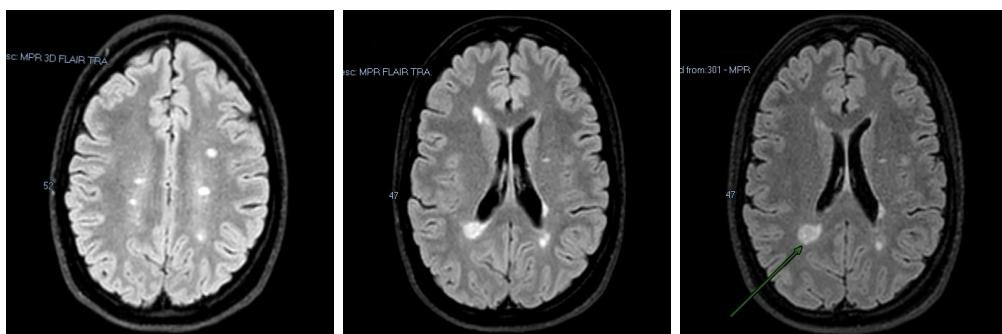


Figura 1.3: Immagini di risonanze magnetiche, le lesioni visibili sono di colore bianco



Figura 1.4: Stesse lesioni delle immagini precedenti, viste dal piano frontale

## 1.3 Stato dell'arte

Negli ultimi anni, l'applicazione delle reti neurali, ha mostrato un grande potenziale nel supporto alla diagnosi della sclerosi multipla [10].

Studi hanno dimostrato l'efficacia delle reti nell'automatizzare la segmentazione e la quantificazione delle lesioni, facilitando una diagnosi più accurata e riducendo significativamente lo sforzo e il tempo richiesto a radiologi e neurologi per analizzare manualmente le immagini.

L'analisi delle immagini delle risonanze magnetiche è fondamentale per monitorare la progressione della malattia e valutare l'efficacia delle terapie, a tale scopo l'utilizzo delle reti neurali facilita la valutazione del carico volumetrico delle lesioni e la loro evoluzione nel corso del tempo.

Storicamente l'attenzione era rivolta principalmente alle immagini trasversali, ma ciò è cambiato nel 2017 con la revisione dei criteri di McDonald, che ha evidenziato la necessità di valutare la progressione della malattia sia dal punto di vista spaziale che temporale [11]. L'introduzione dei dataset di Open MS Data, di ISBI e del dataset della challenge MSSEG-2 ha contribuito a standardizzare i benchmark, includendo immagini di risonanze provenienti da più centri e scanner diversi per sviluppare modelli più robusti e generalizzabili. Questo ha permesso ai ricercatori di valutare e confrontare in modo oggettivo le prestazioni dei diversi modelli di reti neurali, migliorando la riproducibilità dei risultati. La maggior parte dei lavori presentati alla challenge MSSEG-2 ha impiegato le CNN, con architetture U-Net particolarmente diffuse. Specialmente la nnU-Net v2, con cui si è dimostrato di poter migliorare ulteriormente la capacità di segmentare strutture complesse, consentendo quindi una maggiore precisione nell'identificazione delle lesioni.

I risultati degli studi esaminati evidenziano come le reti neurali rappresentino uno strumento promettente per migliorare il processo diagnostico e il monitoraggio della SM.

Nonostante ciò, la complessità della malattia e la variabilità di essa tra i pazienti rendono ancora necessario lo sviluppo di approcci più complessi e robusti, in grado di adattarsi a scenari clinici diversi.

# Capitolo 2

## Architettura delle reti neurali

L'architettura delle reti neurali si basa sulle connessioni tra i neuroni, è necessario quindi capire il funzionamento di un singolo neurone per poter comprendere la struttura e il funzionamento delle reti neurali.

### 2.1 Un singolo neurone

Considerando un solo neurone a cui non è applicata nessuna funzione di attivazione, esso prende in ingresso diversi input binari e ad ogni input viene associato un peso, un numero reale che esprime l'importanza dell'input rispetto all'output. L'output viene determinato dalla funzione  $a = \sum_{i=1}^n x_i \cdot w_i$ , se è maggiore di una certa soglia l'output è 1, altrimenti 0 [12].

Per semplificare la descrizione di un neurone, la definizione di soglia può essere sostituita con quella di bias, una misura della facilità con cui l'output del neurone arriva a 1.

Un bias negativo significa che sarà difficile ottenere l'output uguale a 1, viceversa con un bias grande, l'output perciò viene determinato dalla funzione  $a = \sum_{i=1}^n x_i \cdot w_i + b$ .

Un neurone a cui non vi è applicata nessuna funzione di attivazione ha principalmente due limiti non trascurabili:

- Grande sensibilità dell'output a piccole variazioni dei pesi o del bias
- Impossibilità di risolvere problemi complessi con un solo strato

Al fine di superare tali limiti, viene introdotta la funzione di attivazione, fondamentale per l'utilizzo delle reti neurali in quanto determina l'attivazione dei neuroni.

Senza tale funzione infatti l'output dei neuroni è sempre una trasformazione lineare degli input e il modello risulta inadeguato per affrontare problemi complessi [13].

In seguito vengono mostrate le principali funzioni di attivazione.

### 2.1.1 Funzione a gradino

Con l'utilizzo della funzione a gradino, l'attivazione del neurone avviene al superamento di una certa soglia, di conseguenza gli stati possibili del neurone sono soltanto due, attivo o non attivo.

Questo tipo di funzione di attivazione è utilizzabile nel dominio dei problemi lineari, ad esempio le funzioni logiche, nelle quali una singola retta può essere utilizzata per separare le classi 0 e 1.

Tuttavia la funzione a gradino non è flessibile, in quanto presenta numerose difficoltà durante l'allenamento e può essere utilizzata soltanto come funzione di attivazione per reti a singolo output a classificazione binaria [14].

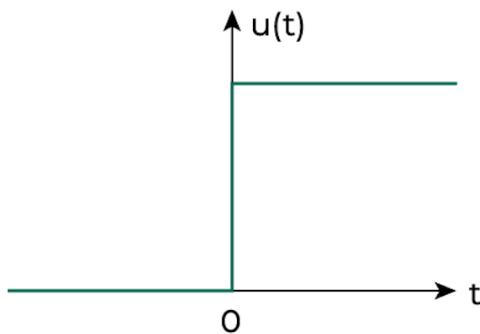


Figura 2.1: Funzione a gradino[15]

### 2.1.2 Funzione sigmoide

La funzione sigmoide è non lineare, perciò consente un'attivazione non lineare del neurone. Dall'equazione che descrive la funzione:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

possiamo osservare che i valori di  $y$  variano rapidamente al variare di  $x$  quando  $x$  si trova nell'intervallo

$$-2 \leq x \leq +2$$

Ciò fa sì che la funzione tenda a spingere i valori di  $y$  verso uno dei due estremi della curva, migliorando così la capacità di distinguere chiaramente tra le classi. Questo comportamento la rende particolarmente indicata per reti neurali poco profonde, come nel caso della simulazione di funzioni logiche.

La problematica principale della funzione di attivazione sigmoide riguarda l'apprendimento nelle regioni quasi orizzontali della curva, in tali regioni la rete non è in grado di aggiornare efficacemente i pesi e di conseguenza apprende molto lentamente.

La sigmoide è di particolare interesse in questa tesi in quanto verrà utilizzata proprio nello strato finale della rete neurale per la segmentazione delle lesioni da sclerosi multipla.

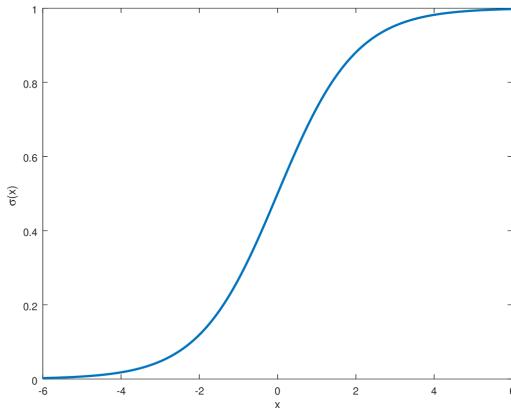


Figura 2.2: Funzione sigmoide[16]

### 2.1.3 Funzione tangente iperbolica

La funzione di attivazione tangente iperbolica fa parte delle funzioni non lineari, come la funzione sigmoide, ed è definita come:

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Il vantaggio rispetto alla sigmoide è che input negativi si mappano su output negativi, questa caratteristica permette un migliore addestramento della rete.

Anche questa funzione è utilizzata nella classificazione binaria.

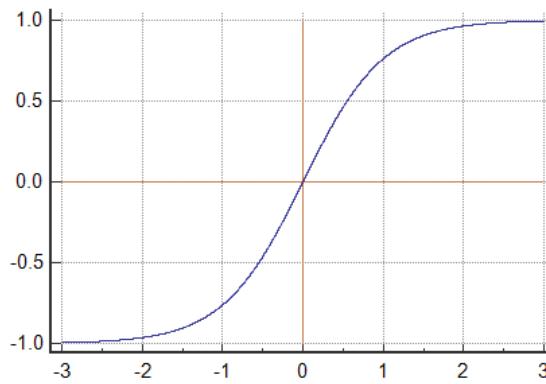


Figura 2.3: Funzione tangente iperbolica[17]

### 2.1.4 Funzione ReLU

La funzione Rectified Linear Unit (ReLU), è una funzione lineare a tratti che restituisce un output positivo in caso di input positivo, zero in caso contrario.

## Relu di base

La ReLU è una funzione a rampa descritta dalla seguente equazione:

$$f(x) = x^+ = \max(0, x)$$

È utilizzata soprattutto nelle reti neurali profonde ed ha un apprendimento molto più veloce rispetto alle altre funzioni di attivazione.

È inoltre molto efficace nei problemi di classificazione, mentre per quanto riguarda la rappresentazione di funzioni smooth richiede più strati di quanti ne richiederebbe una funzione tanh, in quanto servono molte funzioni lineari per rappresentarle.

Il problema principale della ReLU, si presenta quando durante l'allenamento un neurone restituisce sempre zero, ciò causa l'impossibilità di aggiornare i pesi e di conseguenza la "morte" del neurone (Dying ReLU).

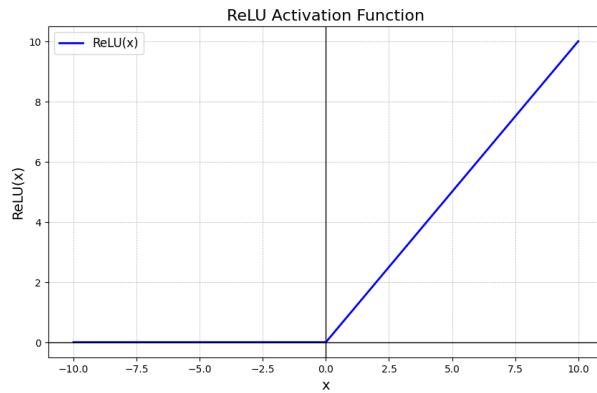


Figura 2.4: Funzione ReLU[18]

## Leaky ReLU

Per evitare il problema dying ReLU, viene introdotta la funzione di attivazione leaky ReLU. Ciò consiste nell'inserire un moltiplicatore che inclina la parte di grafico negativa, prevenendo così la tendenza alla "morte" dei neuroni.

La funzione è definita come:

$$f(x) = \begin{cases} \alpha x, & \text{se } x \leq 0 \\ x, & \text{se } x > 0 \end{cases}$$

Per scegliere il moltiplicatore viene usato il neurone maxout, che calcola il valore massimo tra due insiemi indipendenti di pesi:

$$f(x) = \max(w_1x + b_1, w_2x + b_2)$$

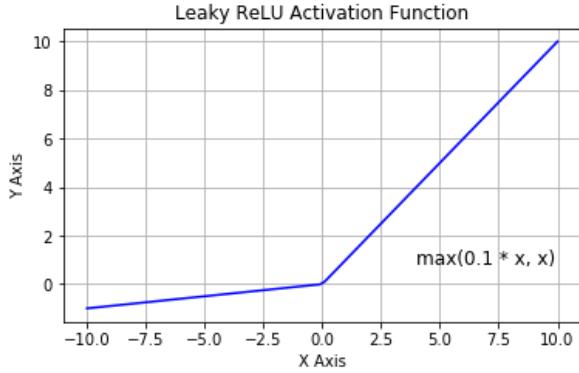


Figura 2.5: Funzione Leaky ReLU[19]

### 2.1.5 Funzione softmax

La funzione softmax, è spesso utilizzata nello strato finale di una rete neurale quando si affrontano problemi di classificazione multiclass [20].

Tale funzione riceve come input un insieme di valori numerici e li trasforma in probabilità (e quindi sommano a 1). Prende in ingresso un vettore di numeri reali e produce un altro vettore della stessa dimensione come output. La funzione è descritta dalla seguente equazione:

$$f(x) = \frac{e^x}{\sum_i e^{x_i}}$$

Per ogni input  $x$ , la funzione softmax calcola l'esponenziale, poi normalizza il valore ottenuto dividendolo per la somma di tutti gli esponenziali.

La classe con la probabilità più alta viene quindi predetta come classe di output finale. Le problematiche principali riguardano la sensibilità ai valori anomali dei dati di input, che possono causare risultati inaffidabili, e l'assunzione che ogni classe sia ugualmente importante, il che implica che la rete tenderà a imparare meglio la classe che si presenta più frequentemente.

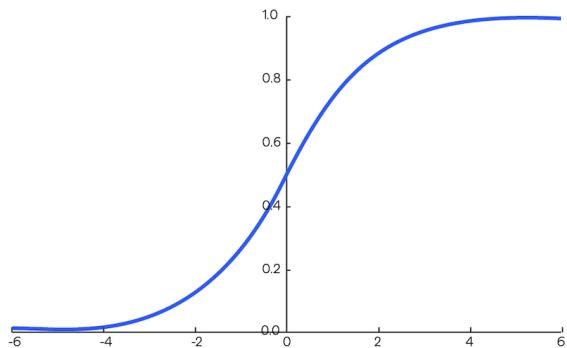


Figura 2.6: Funzione Softmax[21]

## 2.2 Uno strato di neuroni - Hidden Layer

Una rete neurale costituita da un singolo strato nascosto di neuroni, è composta da tre elementi: unità di input, unità nascoste e unità di output.

Ogni neurone prende in ingresso un vettore di input  $\mathbf{x}$ , uno di pesi  $\mathbf{w}$  e un singolo bias  $b$ . Se consideriamo uno strato composto da  $n$  neuroni, allora i vettori dei pesi possono essere rappresentati in una matrice  $\mathbf{W}$  di dimensione  $m \times n$ , dove  $m$  è il numero di neuroni e  $n$  è il numero degli input, ogni elemento  $w_{ij}$  della matrice è il peso del input j-esimo rispetto al neurone i-esimo [22].

Il calcolo che viene effettuato dallo strato della rete può essere quindi descritto come il prodotto tra la matrice dei pesi  $\mathbf{W}$  e il vettore degli input  $\mathbf{x}$ , la somma del vettore risultante con il vettore dei bias  $\mathbf{b}$  e infine l'applicazione della funzione di attivazione al vettore risultante, consentendo alla funzione di agire su ogni elemento  $f(\mathbf{W}\mathbf{x} + \mathbf{b})$ .

$$\begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m1} & w_{m2} & \cdots & w_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

## 2.3 Rete neurale profonda - Multi Layer Perceptron

Le reti neurali profonde sono composte da più strati nascosti di neuroni, la più semplice è chiamata rete neurale feedforward: l'output delle unità di ogni strato viene passato ai neuroni dello strato successivo, senza che ci siano retroazioni all'interno della rete.

Usando apici tra parentesi quadre per indicare lo strato e partendo da 0 per lo strato di input, possiamo definire:  $\mathbf{W}^{[i]}$  la matrice dei pesi dello strato i-esimo nascosto e  $\mathbf{b}^{[i]}$  il vettore dei bias.

Chiamando  $a^{[i]}$  l'uscita dello strato i-esimo e  $z^{[i]}$  la combinazione lineare dell'uscita, otteniamo:

$$\mathbf{z}^{[i]} = \mathbf{W}^{[i]} \mathbf{a}^{[i-1]} + \mathbf{b}^{[i]}$$

Le equazioni hanno sempre la stessa forma per ogni strato, perciò l'algoritmo per il calcolo dell'output per una rete feedforward diventa:

$$\text{per } i = 1, \dots, n : \begin{cases} \mathbf{z}^{[i]} = \mathbf{W}^{[i]} \mathbf{a}^{[i-1]} + \mathbf{b}^{[i]} \\ \mathbf{a}^{[i]} = g^{[i]}(\mathbf{z}^{[i]}) \end{cases} \quad \hat{\mathbf{y}} = \mathbf{a}^{[n]}$$

È importante aggiungere ora una funzione di attivazione non lineare, perchè senza di essa la rete risulterebbe uguale ad una con un solo strato, in quanto sarebbe solo una variazione di notazione e un diverso set di pesi.

## 2.4 Universalità delle reti neurali

Supponendo di avere una funzione complicata e irregolare, esiste sempre una rete neurale che, indipendentemente dalla complessità della funzione, è in grado di restituire, per ogni valore di input  $x$ , il corrispondente valore  $f(x)$  oppure un'approssimazione molto vicina ad esso [23].

### 2.4.1 Universalità con un solo strato nascosto

Consideriamo una rete composta da un singolo strato nascosto, con un input e un output. Se, ad ogni neurone dello strato nascosto, associamo una funzione di attivazione sigmoide e regoliamo opportunamente pesi e bias per approssimarla a una funzione a gradino, allora, aumentando il numero di neuroni, è possibile riprodurre con buona precisione l'andamento di una funzione arbitraria.

In questo caso la funzione approssimabile è bidimensionale.

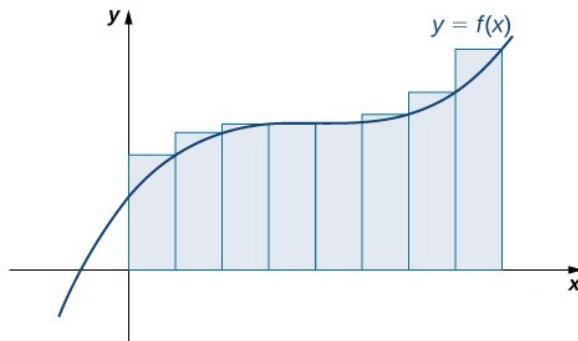


Figura 2.7: Approssimazione di una funzione[24]

Aggiungendo delle variabili di input  $(x_1, x_2, \dots, x_n)$  la rete può approssimare dei gradini multidimensionali nel dominio di  $\mathbb{R}^n$ .

Nel caso di due input possiamo rappresentare una superficie tridimensionale.

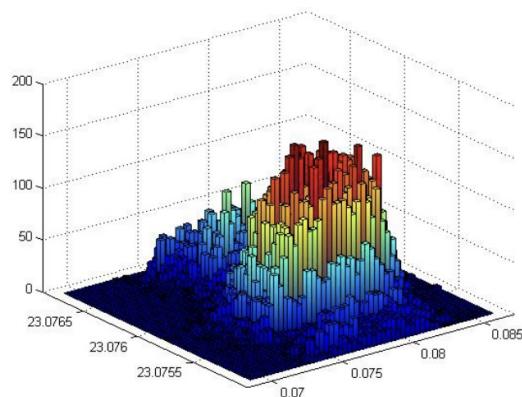


Figura 2.8: Approssimazione di una funzione con 2 input[25]

In sintesi, anche reti con un solo strato nascosto sono in grado di approssimare funzioni arbitrariamente complesse, estendendo la loro capacità a domini multidimensionali e superfici di elevata complessità.

Tuttavia, utilizzare reti più profonde richiede meno neuroni per ottenere la stessa capacità di approssimazione, risultando quindi più efficienti nell'apprendere strutture complesse e nel generalizzare a nuovi dati.

## 2.5 Rete neurale convoluzionale - CNN

Le reti neurali convoluzionali sono analoghe alle reti neurali artificiali tradizionali, ogni neurone continua a ricevere un input e ad eseguire un'operazione e l'intera rete rappresenta comunque una singola funzione.

L'ultimo strato contiene le funzioni di costo associate alle diverse classi, e tutti i principi sviluppati per le reti neurali tradizionali si applicano ancora. La principale differenza con le ANN, è che le reti neurali convoluzionali vengono impiegate nel campo del riconoscimento di pattern all'interno delle immagini [26].

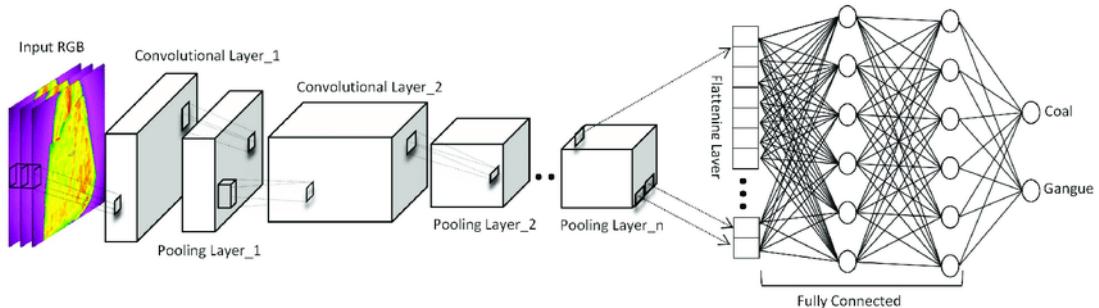


Figura 2.9: Rete Neurale Convoluzionale [27]

### Cos'è la convoluzione?

La convoluzione è un'operazione matematica che combina due funzioni per descrivere la sovrapposizione tra loro. Prende due funzioni e le fa scorrere l'una sull'altra, moltiplicando i valori della funzione in ogni punto di sovrapposizione e aggiungendo i prodotti per creare una nuova funzione [28].

Di seguito è riportata la formula della convoluzione utilizzata nelle reti neurali convoluzionali:

$$y_{i_l+1, j_l+1, d}^{l+1} = \sum_{i=0}^H \sum_{j=0}^W \sum_{d_l=0}^{D_l} f_{i,j,d_l,d}^l \times x_{i_l+i, j_l+j, d_l}^l.$$

dove  $y_{i,j,d}^{l+1}$  è l'output,  $f_{m,n,c,d}^l$  è il kernel di convoluzione,  $x_{i+m, j+n, c}^l$  è l'input,  $b_d$  è il bias, e  $H, W, D_l$  rappresentano altezza, larghezza e profondità del filtro.

### 2.5.1 Architettura delle reti neurali convoluzionali

I neuroni di una CNN sono organizzati in tre dimensioni: altezza, larghezza e profondità. Per profondità si intende la terza dimensione del volume di attivazione, cioè il numero di canali o feature map [26].

La rete neurale convoluzionale è strutturata su quattro aree principali:

1. Strato di input: contiene i valori dei pixel dell'immagine in ingresso
2. Strato convoluzionale: calcola l'uscita dei neuroni connessi a regioni locali dell'input, alla quale viene poi applicata la funzione di attivazione ReLU
3. Strato di pooling: esegue un downsampling riducendo ulteriormente il numero di parametri di attivazione
4. Strati fully connected: sono delle ANN standard e vengono utilizzati per la classificazione

L'archittettura standard per una CNN, consiste in strati convoluzionali sovrapposti, seguiti da strati di pooling e ripetuti più volte prima di passare agli strati fully connected.

### 2.5.2 Strato convoluzionale

I parametri su cui si basa lo strato convoluzionale sono i kernel (filtri), tramite i quali si effettua la convoluzione, e i bias.

I kernel sono delle matrici di pesi, che hanno lo scopo di rilevare caratteristiche specifiche nell'immagine di input, sono piccoli in altezza e larghezza ma si estendono lungo tutta la profondità dell'input. Ciò significa che i kernel hanno 3 canali (RGB).

Quando i dati raggiungono uno strato convoluzionale, lo strato applica la convoluzione di ogni filtro lungo la dimensione spaziale dell'input per produrre una mappa di attivazione (feature map), cioè calcola con un prodotto scalare dei valori numerici in ogni posizione. Quindi la rete imparerà kernel che si attivano quando rilevano una caratteristica specifica in una determinata posizione spaziale dell'input, questi output sono noti come attivazioni. Se addestrassimo le ANN su immagini produrrebbero modelli troppo grandi per essere addestrati efficacemente, questo è dovuto ai neuroni fully connected delle ANN standard. Per evitare questo problema, ogni neurone in uno strato convoluzionale è connesso solo a una piccola regione del volume di input. La dimensione di questa regione è chiamata dimensione del campo recettivo del neurone.

#### Campo recettivo del neurone

Il campo recettivo si sposta lungo la dimensione spaziale dell'input, tale spostamento è chiamato **stride**, impostando uno stride piccolo i campi recettivi saranno molto sovrapposti e di conseguenza l'output produrrà attivazioni grandi, la dimensione dello stride

influisce sulla dimensione dell'output.

Un altro metodo per controllare la dimensione spaziale dell'output è lo **zero-padding**, cioè l'aggiunta di zeri ai bordi dell'input.

Oltre alla dimensione spaziale, anche la profondità dell'output dipende da quella dell'input, infatti riducendo il numero di neuroni applicati alla stessa regione di input si riduce di conseguenza la profondità dell'output, con lo svantaggio di una possibile riduzione della capacità di riconoscimento dei pattern.

### Parameter sharing

Per ridurre ulteriormente il numero di parametri si utilizza il parameter sharing.

All'interno dello stesso strato vengono condivisi i parametri, basandosi sull'ipotesi che se una caratteristica è rilevante in una certa regione dell'immagine lo sarà anche in altre, utilizzando così per ogni mappa gli stessi pesi e bias.

### 2.5.3 Strato di pooling

L'obiettivo dello strato di pooling è di ridurre il numero di parametri e la complessità computazionale del modello.

Nella maggior parte delle reti convoluzionali si utilizzano strati di max-pooling, prendendo il valore massimo nella finestra e utilizzando kernel di dimensione 2x2 applicati con uno stride di 2.

In alternativa è possibile utilizzare l'overlapping pooling, in cui lo stride è impostato su 2 e la dimensione del kernel su 3, questo significa che le regioni di pooling si sovrappongono parzialmente tra loro. Tuttavia, a causa della natura distruttiva del pooling, utilizzare kernel di dimensione superiore a 3 comporta un grosso peggioramento delle prestazioni del modello.

Le CNN possono includere anche il general pooling. Gli strati di general pooling sono costituiti da neuroni in grado di eseguire la normalizzazione e l'average pooling.

### 2.5.4 AlexNet

Studiando le reti neurali convoluzionali è importante citare AlexNet, una delle più importanti reti neurali convoluzionali. La sua descrizione e i risultati furono pubblicati nel 2012 in uno degli articoli di ricerca più influenti nella storia del deep learning da Krizhevsky et al. (2012) [29]. Si aggiudicò il primo posto alla ImageNet Large Scale Visual Recognition Challenge del 2012, con un errore del 15,3%, rispetto al 26,2% del secondo classificato [30].

La rete utilizzò 60 milioni di parametri e 650 mila neuroni, l'allenamento fu eseguito su due GPU in parallelo, al fine di suddividere i calcoli e la memoria.

Inoltre per migliorare la capacità di generalizzazione, furono utilizzate tecniche di data augmentation e una regolarizzazione basata su weight decay con coefficiente 0.0005.

## Architettura

L'architettura della rete AlexNet è costituita da:

- 8 strati totali, di cui i primi 5 convoluzionali e gli ultimi 3 fully connected
- Funzione di attivazione ReLU, che consente un'apprendimento veloce ed evita la saturazione del gradiente
- Max pooling 3x3 e stride 2
- Local Response Normalization, che in caso di attivazione molto forte di un neurone inibisce le attivazioni dei neuroni vicini permettendo alla rete di considerare i pattern più significativi
- Dropout negli strati fully connected per ridurre l'overfitting, di cui verrà spiegato il significato in seguito

## 2.6 U-Net

La U-Net è un tipo di rete neurale convoluzionale, proposta per la prima volta nel 2015 per la segmentazione di immagini biomediche. Questa rete si ottiene dalla concatenazione di una rete contrattiva (encoder) e di una espansiva (decoder), senza la presenza di strati fully connected [31].

È di particolare rilevanza in questa tesi in quanto verrà utilizzata per la segmentazione delle immagini delle risonanze magnetiche di pazienti con sclerosi multipla.

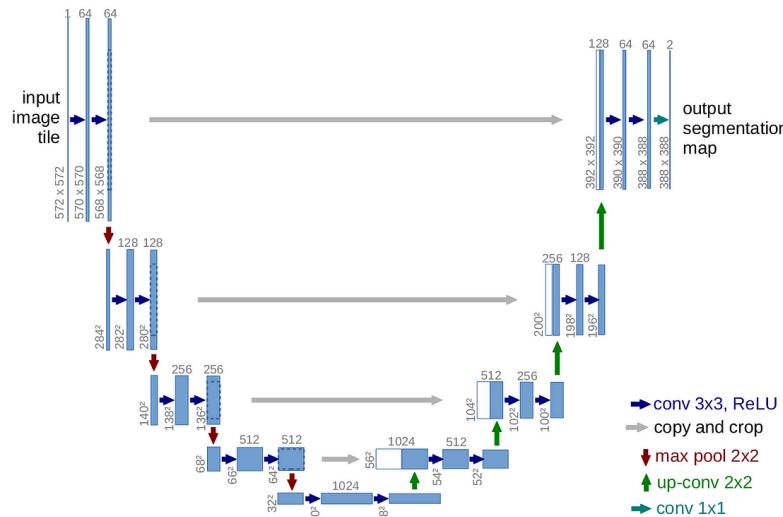


Figura 2.10: Rete Neurale Convoluzionale U-Net[32]

### 2.6.1 Architettura della U-Net

Come anticipato, la rete è strettamente strutturata sulla concatenazione di un encoder e di un decoder, tra i quali è presente il bottleneck, lo strato più profondo della rete, in cui la rappresentazione delle feature è altamente astratta.

Si può notare nella Figura 2.10 che l'architettura è simmetrica, ogni passaggio nell'encoder ha un corrispettivo nel decoder, ciò permette il bilanciamento tra l'estrazione delle feature e la successiva ricostruzione.

La concatenazione si effettua attraverso le skip connections, ovvero l'unione di due o più feature map lungo i canali.

#### Skip connections

Le skip connections saltano uno o più strati della rete neurale, inviando l'output di uno strato come input a uno strato successivo, non necessariamente adiacente [33].

Esistono due tipi di configurazioni in cui vengono impiegate: short skip connections e long skip connections.

Nel caso della U-Net si parla di long skip connections, esse vengono impiegate per trasferire le feature dal percorso di encoder a quello di decoder, al fine di recuperare le informazioni spaziali perse durante il processo di downsampling.

#### Rete contrattiva - encoder

Lo scopo della rete contrattiva è ridurre la dimensione spaziale dell'immagine tramite strati convoluzionali e di pooling, estraendo le feature più importanti [31].

L'architettura segue quella di una rete convoluzionale, consiste cioè nell'applicazione di due strati convoluzionali 3x3, ciascuno seguito da una ReLU e da un'operazione di max pooling 2x2 con stride 2 per il downsampling.

Ad ogni passo di downsampling vengono raddoppiati i canali di feature, al fine di rappresentare strutture più complesse e mantenere la stessa quantità di contenuto informativo che la rete deve elaborare, infatti la qualità dell'immagine si riduce ma aumentano i canali.

#### Rete espansiva - decoder

Nella parte espansiva, invece di effettuare ulteriori riduzioni, si aumenta la risoluzione concatenando l'output con le feature ad alta risoluzione della rete contrattiva.

La rete consiste quindi in un upsampling seguito da uno strato convoluzionale 2x2 che dimezza il numero di canali di feature, una concatenazione con la feature map corrispondente del percorso contrattivo e due strati convoluzionali 3x3 seguiti da una ReLU.

Nell'ultimo strato viene usata una convoluzione 1x1 per mappare ciascun vettore di feature a 64 componenti nel numero desiderato di classi.

# Capitolo 3

## Allenamento delle reti neurali

In generale, l’allenamento di una rete neurale è il processo attraverso il quale la rete apprende l’esecuzione di un compito specifico, scegliendo pesi e bias in modo da minimizzare l’errore tra l’output prodotto e quello desiderato.

Per valutare l’accuratezza di tale approssimazione si introduce una funzione di costo, che quantifica l’errore commesso.

### 3.1 Funzione di costo e ottimizzazione

La funzione di costo esprime, rispetto ad un ingresso  $x$ , quanto l’output della rete  $\hat{y}$  sia vicino al valore corretto di  $y$  [22].

#### 3.1.1 Cross-entropy e Dice loss function

La funzione di costo più utilizzata per la classificazione è costruita in modo che favorisca gli output corretti delle etichette dell’allenamento, rendendole più probabili. Questo approccio è chiamato maximum likelihood estimation e la funzione è la cross-entropy loss, vengono scelti pesi e bias in modo da massimizzare la probabilità logaritmica delle etichette  $y$  corrette nei dati di addestramento rispetto agli input  $x$ .

Questa funzione di costo ci assicura che man mano che la probabilità dell’esito corretto aumenta, la probabilità dell’esito errato diminuisce.

$$L_{CE}(\hat{y}, y) = -\log p(y | x) = -[y \log \hat{y} + (1 - y) \log(1 - \hat{y})]$$

#### Dice Loss function

Al fine di questa tesi si utilizza la funzione Dice Loss, impiegata principalmente nelle reti neurali convoluzionali in ambito biomedico. La Dice Loss deriva dalla metrica Dice similarity coefficient, che misura la sovrapposizione di due insiemi  $A$  e  $B$ .

Il Dice coefficient è definito come:

$$Dice = \frac{2|A \cap B|}{|A| + |B|}$$

dove  $A$  è l'insieme dei pixel previsti come appartenenti alla classe positiva,  $B$  è l'insieme dei pixel reali appartenenti alla classe positiva,  $|A \cap B|$  rappresenta il numero di pixel correttamente classificati come positivi.

Il coefficiente assume valori compresi tra 0 e 1, dove 1 indica una perfetta sovrapposizione tra previsione e ground truth, 0 indica l'assenza totale di sovrapposizione.

La funzione Dice Loss è così definita:

$$DiceLoss = 1 - Dice$$

Utilizzando ora i valori di probabilità predetti per ciascun pixel, la funzione può essere espressa come:

$$DiceLoss = 1 - \frac{2 \sum_i p_i g_i + \epsilon}{\sum_i p_i + \sum_i g_i + \epsilon}$$

dove  $p_i$  è la probabilità prevista per il pixel  $i$ ,  $g_i$  è il valore reale del pixel (0 oppure 1),  $\epsilon$  è un termine introdotto per evitare divisioni per zero.

### 3.1.2 Discesa del gradiente

Lo scopo è quindi minimizzare la funzione di costo, trovando i pesi ottimali che riducono al minimo il valore medio della funzione rispetto a tutti i pesi del modello.

Per minimizzarla si utilizza il metodo della discesa del gradiente, basato sulla determinazione della direzione in cui la funzione cresce più rapidamente e sul successivo spostamento nella direzione opposta, in modo da ridurne il valore. Di seguito viene riportata la formula che descrive il problema di minimizzazione della funzione di costo, da risolvere attraverso la discesa del gradiente e la cui formula verrà riportata successivamente.

$$\hat{\theta} = \arg \min_{\theta} \frac{1}{m} \sum_{i=1}^m C_{CE}(f(x^{(i)}; \theta), y^{(i)})$$

dove  $\hat{\theta}$  rappresenta i parametri ottimali del modello,  $\arg \min_{\theta}$  indica la ricerca del valore di  $\theta$  che minimizza la funzione di costo,  $\frac{1}{m} \sum_{i=1}^m$  è la media dei costi sui  $m$  esempi,  $C$  è una generica funzione di costo,  $f(x^{(i)}; \theta)$  è la previsione del modello per l'esempio  $i$ -esimo, e  $y^{(i)}$  è l'output reale corrispondente.

Nella maggior parte dei casi non è possibile trovare il minimo globale ma soltanto un minimo locale, in quanto le funzioni di costo delle reti multistrato non sono convesse e quindi presentano più di un minimo.

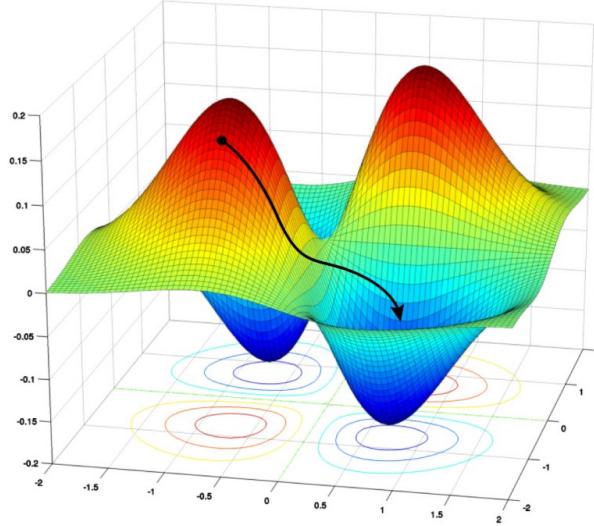


Figura 3.1: Discesa del gradiente[34]

### Dimensione del passo - Learning rate

L'ampiezza dello spostamento verso il minimo della funzione, viene deciso attraverso il learning rate. Il learning rate è un fattore di scala, indica di quanto ci si sposta nella direzione opposta al gradiente per aggiornare i pesi. Quando è alto significa che il passo è grande e si rischia di andare oltre al minimo, in questo caso la rete impara velocemente ma è poco precisa. Se invece il learning rate è basso allora il passo è piccolo e la rete è precisa ma più lenta. L'algoritmo è descritto dalla seguente equazione:

$$\theta_{t+1} = \theta_t - \eta \nabla C(f(x; \theta), y)$$

dove  $\theta_t$  è il vettore dei parametri attuali, mentre  $w_{t+1}$  è il vettore dei nuovi parametri dopo l'aggiornamento,  $\eta$  è il learning rate,  $\nabla C(f(x; \theta), y)$  è il gradiente della funzione di costo,  $f(x; \theta)$  rappresenta il modello di previsione e  $y$  è il valore reale che il modello cerca di predire.

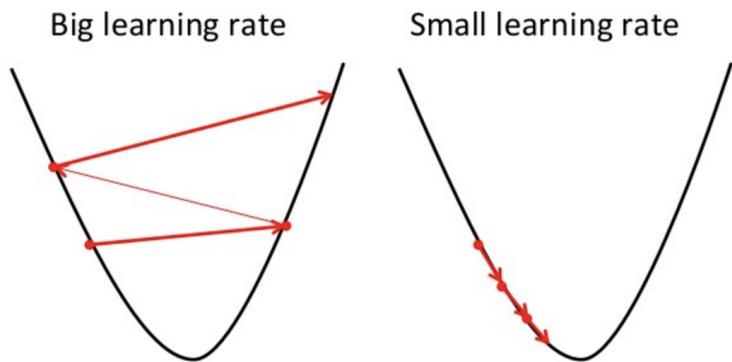


Figura 3.2: Dimensione del passo[35]

### 3.1.3 Backpropagation algorithm

Il gradiente viene calcolato utilizzando l'algoritmo di backpropagation, il quale consente all'informazione proveniente dal costo di fluire all'indietro attraverso la rete [36]. Lo scopo è calcolare come ciascun peso della rete contribuisce all'errore e aggiornarlo nella direzione che riduce il costo. Tale algoritmo si riferisce soltanto al calcolo del gradiente, per l'apprendimento effettivo si utilizza un altro algoritmo, ad esempio Adam.

Per fare ciò si utilizza la regola della catena, la quale permette di calcolare derivate di funzioni composte di cui si conoscono le derivate. La funzione di costo non dipende direttamente dai pesi, perciò la regola della catena è utile per scomporre la derivata. Nel caso di strati composti da singoli neuroni:

$$\frac{\partial C_0}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C_0}{\partial a^{(L)}}$$

Questa prima equazione indica che il gradiente si ottiene moltiplicando le derivate parziali lungo la catena delle dipendenze: da  $w^{(L)}$  a  $z^{(L)}$ , da  $z^{(L)}$  a  $a^{(L)}$ , e infine da  $a^{(L)}$  a  $C_0$ .

$$C_0 = (a^{(L)} - y)^2 \rightarrow \frac{\partial C_0}{\partial a^{(L)}} = 2(a^{(L)} - y)$$

La derivata della funzione di costo rispetto all'attivazione fornisce una misura dell'errore tra l'uscita della rete  $a^{(L)}$  e il valore dell'output  $y$ .

$$a^{(L)} = \sigma(z^{(L)}) \rightarrow \frac{\partial a^{(L)}}{\partial z^{(L)}} = \sigma'(z^{(L)})$$

La derivata della funzione di attivazione descrive come varia l'uscita del neurone rispetto alla sua somma pesata degli input.

$$z^{(L)} = w^{(L)}a^{(L-1)} + b^{(L)} \rightarrow \frac{\partial z^{(L)}}{\partial w^{(L)}} = a^{(L-1)}$$

Infine la derivata di  $z^{(L)}$  mostra che il gradiente rispetto ai pesi è proporzionale all'input dello strato precedente.

Generalizzando il caso ad una rete multistrato, con più di un singolo neurone per strato, possiamo scrivere la seguente espressione, che permette di ottenere di ottenere ogni componente del gradiente:

$$\begin{aligned} \frac{\partial C}{\partial w_{jk}^{(l)}} &= a_k^{(l-1)} \sigma'(z_j^{(l)}) \frac{\partial C}{\partial a_j^{(l)}} \\ \frac{\partial C}{\partial a_j^{(l)}} &= \sum_{j=0}^{n_{l+1}-1} w_{jk}^{(l+1)} \sigma'(z_j^{(l+1)}) \frac{\partial C}{\partial a_j^{(l+1)}} \quad \text{or} \quad 2(a_j^{(L)} - y_j) \end{aligned}$$

### 3.1.4 Suddivisione del dataset per il calcolo del gradiente

#### Discesa stocastica del gradiente - SGD

La discesa stocastica del gradiente è un algoritmo che minimizza la funzione di costo, calcolando il gradiente dopo ogni esempio durante l'allenamento. È definita stocastica perché sceglie casualmente un singolo esempio alla volta.

Questo metodo migliora le prestazioni su un unico step dell'algoritmo, ma non permette un'apprendimento regolare in quanto causa molte oscillazioni nel percorso.

#### Full-batch

Con il full-batch il gradiente viene calcolato sull'intero dataset, il modello infatti vede tutti gli esempi contemporaneamente e ciò permette una stima molto precisa della direzione in cui aggiornare i pesi.

Tuttavia ha un costo di elaborazione molto elevato, perché richiede di elaborare ogni singolo esempio del set ad ogni epoca. Per epoca si intende un passaggio completo sull'intero dataset di allenamento.

#### Mini-batch

Utilizzando il mini-batch il dataset per l'allenamento viene suddiviso in gruppi, per ognuno dei quali viene calcolato il gradiente medio della funzione di costo e vengono aggiornati i pesi di conseguenza.

Ha il vantaggio di essere più stabile rispetto alla discesa stocastica del gradiente, ed è più efficiente perché i gruppi possono essere elaborati in parallelo. Inoltre, permette una stima abbastanza accurata del gradiente con un costo di elaborazione contenuto e una velocità elevata.

### 3.1.5 Varianti della discesa del gradiente

Di seguito sono riportate diverse modalità di calcolo del gradiente.

#### Momentum

Momentum è un metodo utilizzato per migliorare le prestazioni della SGD, permette di accelerare la discesa nella direzione rilevante e ridurre le oscillazioni [37].

Il vettore di aggiornamento è calcolato come segue:

$$v_t = \gamma v_{t-1} + \eta \nabla_{\theta} C(\theta), \quad \theta = \theta - v_t$$

dove  $\theta$  è il vettore dei parametri del modello,  $C(\theta)$  è la funzione di costo,  $\nabla_{\theta} C(\theta)$  è il gradiente della funzione di costo rispetto ai parametri,  $\eta$  è il learning rate,  $v_t$  è il vettore

di aggiornamento con momentum al passo  $t$ ,  $\gamma$  è il termine di momentum, tipicamente impostato a 0.9.

Tale termine aumenta per le dimensioni i cui gradienti puntano nella stessa direzione e riduce gli aggiornamenti per le dimensioni i cui gradienti cambiano direzione. Di conseguenza, otteniamo una convergenza più rapida e oscillazioni ridotte.

## Adagrad

Adaptive gradient (Adagrad), è un algoritmo di ottimizzazione basato sull'adattamento del learning rate ai parametri, che viene effettuato eseguendo aggiornamenti maggiori per i parametri più rari e aggiornamenti più piccoli per parametri più frequenti. Ciò permette di migliorare significativamente la robustezza della SGD.

$$\theta_{t+1} = \theta_t - \frac{\eta \odot g_t}{\sqrt{G_t} + \varepsilon}$$

dove  $\theta_t$  indica il vettore dei parametri al passo  $t$ ,  $\theta_{t+1}$  rappresenta i parametri aggiornati al passo successivo,  $g_t$  è il gradiente della funzione di costo rispetto ai parametri al passo  $t$ ,  $\odot$  denota la moltiplicazione elemento per elemento,  $G_t$  è una matrice diagonale contenente la somma dei quadrati dei gradienti accumulati fino al passo  $t$  e  $\varepsilon$  è un termine che serve ad evitare divisioni per zero.

Il principale svantaggio di Adagrad, è l'accumulo dei quadrati dei gradienti al denominatore: poiché ogni termine aggiunto è positivo, la somma accumulata cresce continuamente durante l'allenamento, facendo diminuire il learning rate fino a valori infinitesimali, momento in cui l'algoritmo non riesce più ad apprendere.

## RMSprop

Root Mean Square propagation (RMSprop), è stato proposto per risolvere il problema di apprendimento di Adagrad. Questo metodo consente di mantenere una media esponenziale dei quadrati dei gradienti passati e la usa per adattare il learning rate, evitando che diminuisca drasticamente.

$$E[g^2]_t = 0.9 E[g^2]_{t-1} + 0.1 g_t^2 \quad \theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \varepsilon}} g_t$$

dove  $E[g^2]_t$  è la media esponenziale dei quadrati dei gradienti al passo  $t$ , la costante 0.9 determina il peso della media dei quadrati dei gradienti precedenti, mentre 0.1 è il peso del gradiente corrente.

## Adam

Adaptive Moment Estimation (Adam), è un altro metodo che calcola tassi di apprendimento adattivi per ciascun parametro. Oltre a memorizzare una media esponenzialmente decrescente dei gradienti al quadrato passati  $v_t$ , come RMSprop, conserva anche una media esponenzialmente decrescente dei gradienti passati  $m_t$ , simile al momentum.

$v_t$  e  $m_t$  sono rispettivamente stime del primo momento (la media) e del secondo momento (la varianza non centrata) dei gradienti, da cui deriva il nome del metodo.

Il problema di tali stime è che sono inizializzate a zero, rendendo l'aggiornamento iniziale dei parametri troppo piccolo e rallentando l'apprendimento. Per risolvere questo problema vengono calcolate le stime dei momenti di primo e secondo ordine corrette dal bias.

$$\theta_{t+1} = \theta_t - \frac{\eta \hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}$$

dove  $\hat{m}_t$  è il primo momento corretto,  $\hat{v}_t$  è il secondo momento corretto e  $\varepsilon = 10^{-8}$ .

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

dove  $\beta_1^t = 0.9$  e  $\beta_2^t = 0.999$  sono i tassi di decadimento.

Il vantaggio dell'utilizzo di Adam come ottimizzatore risiede nel calcolo del learning rate individuale per ogni parametro, migliorando così la stabilità della rete.

Inoltre, correggendo le stime iniziali si consente un aggiornamento dei parametri più realistico .

## 3.2 Parametri e iperparametri

Un'importante distinzione da fare, per comprendere come funziona l'allenamento delle reti neurali riguarda i parametri e gli iperparametri.

I parametri sono valori interni che vengono aggiornati durante l'allenamento e rappresentano ciò che la rete impara, stiamo parlando dei pesi e dei bias.

Gli iperparametri vengono definiti a priori e non è possibile modificarli durante l'apprendimento, rappresentano come la rete impara.

### Iperparametri

Alcuni esempi di iperparametri, utili al fine di questa tesi, sono:

- Batch size
- Learning rate
- Tipo di ottimizzatore

- Funzione di attivazione
- Tasso di dropout o regolarizzazione
- Numero di strati e di neuroni per strato
- Numero di epoche, ovvero quante volte il modello vede tutto il dataset di allenamento

### 3.3 Possibili problematiche dell’allenamento

Durante l’allenamento di una rete neurale è possibile riscontrare diversi problemi, legati al modello o ai parametri scelti per l’allenamento, di seguito sono riportate le problematiche più comuni.

#### 3.3.1 Overfitting e underfitting

Si parla di overfitting nell’apprendimento, quando il modello si adatta perfettamente al dataset di addestramento. Un buon modello dovrebbe essere in grado di generalizzare dal set di allenamento al set di test [22].

Questo fenomeno accade perchè la rete durante l’allenamento cerca di ridurre l’errore, assegnando pesi molto grandi alle feature che predicono perfettamente la classe, adattandosi anche al rumore.

Per evitare l’overfitting, si aggiunge un termine di regolarizzazione  $R(\theta)$  alla funzione di costo:

$$\hat{\theta} = \arg \max_{\theta} \sum_{i=1}^m \log P(y^{(i)} | x^{(i)}) - \alpha R(\theta)$$

Il termine di regolarizzazione serve a penalizzare i pesi troppo grandi.  $\alpha$  è il parametro di forza di regolarizzazione, più è grande  $\alpha$ , più bassi saranno i pesi del modello, diminuendo così la dipendenza del modello dal set di allenamento.

Soltanente la regolarizzazione è calcolata come la somma dei quadrati dei pesi:

$$R(\theta) = \sum_{j=1}^n \theta_j^2$$

Un’importante tecnica di regolarizzazione utilizzata per ridurre l’overfitting e migliorare la capacità di generalizzazione del modello, soprattutto nelle reti neurali convoluzionali è il **dropout**. Esso consiste nella disattivazione casuale di alcuni neuroni durante l’allenamento della rete, ciò impedisce al modello di dipendere da singoli neuroni o connessioni. Infatti in presenza di neuroni con pesi molto grandi la rete impara a fare affidamento su quei singoli, nel momento in cui vengono disattivati la rete deve ridistribuire le informazioni su altri neuroni.

Al contrario, l'underfitting, si verifica quando la rete non è in grado di adattarsi ai dati di allenamento. Le cause riguardano ad esempio l'utilizzo di modelli troppo semplici, una regolarizzazione eccessiva o un numero di epoche insufficiente.

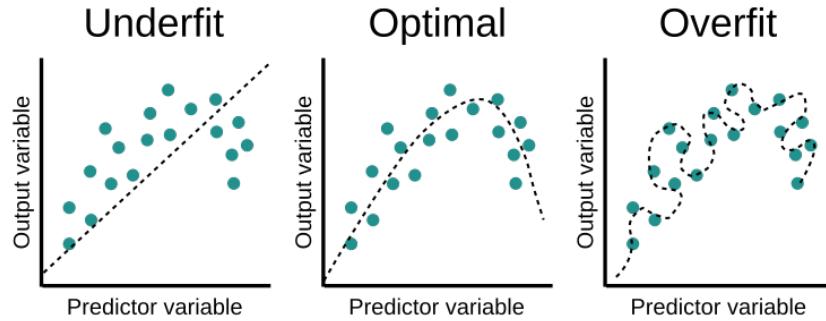


Figura 3.3: Underfitting e overfitting[38]

### 3.3.2 Internal covariate shift

L'internal covariate shift è un fenomeno che si presenta durante l'allenamento, perché i valori di input che uno strato riceve da quello precedente cambiano continuamente a causa dell'aggiornamento costante dei pesi, rallentando di conseguenza l'allenamento e rendendo instabili i gradienti [39].

Il metodo utilizzato per risolvere questo problema è il **batch normalization**. I valori del batch vengono normalizzati, concentrandosi sull'ottenere la media di tali valori intorno allo 0, e varianza 1 così da avere valori con ampiezza simile.

Il procedimento per calcolare la normalizzazione del batch è quello che segue:

Calcolo della media delle attivazioni del batch:

$$\mu_B = \frac{1}{m} \sum_{i=1}^m x_i$$

dove  $m$  è il numero di campioni nel batch e  $x_i$  è l'attivazione del neurone per l'i-esimo esempio del batch.

Calcolo della varianza del batch, cioè quanto le attivazioni si discostano dalla media:

$$\sigma_B^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2$$

Normalizzazione dell'attivazione di ogni neurone nel batch:

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

### 3.3.3 Vanishing gradient

Il problema del vanishing gradient si presenta durante il backpropagation, quando le derivate delle funzioni diventano sempre più piccole mano a mano che si attraversano gli strati della rete [40].

Questo fenomeno è particolarmente evidente nelle reti neurali profonde e, quando i valori di aggiornamento dei pesi diventano molto piccoli, il tempo impiegato per l'allenamento può aumentare notevolmente o perfino causare un'interruzione dell'apprendimento.

Il vanishing gradient è associato principalmente alle funzioni di attivazione sigmoide e tanh, in quanto quando i pesi assumono valori agli estremi delle funzioni, i gradienti diventano molto piccoli, ciò compromette notevolmente le prestazioni del modello.

Per risolvere il problema è possibile utilizzare le skip connection, funzioni di attivazioni differenti, e come nel caso dell'internal covariate shift, il batch normalization.

### 3.3.4 Scarsità di dati disponibili

Nel ipotesi in cui si dispone di pochi dati disponibili per l'allenamento è possibile utilizzare il **transfer learning** per aggirare il problema.

Il transfer learning consente di trasferire la conoscenza da un modello già allenato su un dataset più grande al modello che si vuole utilizzare, indebolendo l'ipotesi secondo cui i dati di addestramento e di test devono essere indipendenti e distribuiti in modo uguale , migliorando così la funzione predittiva del modello di destinazione [41].

È inoltre possibile riallenare in parte la rete pre-allenata su un nuovo dataset, in modo da adattare le conoscenze pregresse al nuovo scopo, questo metodo utilizzato nel transfer learning si chiama **fine tuning**. Ciò consente di migliorare le prestazioni della rete in presenza di dataset piccoli o quando l'allenamento da zero risulterebbe troppo costoso.

Un'altra tecnica utilizzata per aggirare il problema della scarsità di dati disponibili è la **data augmentation**. Per ogni esempio del dataset si generano nuove versioni modificate, nel caso delle reti convoluzionali le trasformazioni includono: rotazioni, flip orizzontali o verticali, cambiamenti di luminosità, ritagli e aggiunta di rumore.

Infine, si può anche utilizzare la **cross validation**, cioè addestrare la rete su dati di allenamento diversi. Per fare ciò si divide il dataset iniziale in k set e si allena il modello k volte, utilizzando ogni volta uno dei set come test set, infine si fa la media delle metriche ottenute dai vari test set. Questo consente di ottenere una stima più affidabile delle prestazioni del modello.

# Capitolo 4

## Applicazione ad un caso reale

Si vuole ora applicare una rete neurale convoluzionale ad un caso reale, nello specifico si utilizza la rete U-Net per la segmentazione automatica delle lesioni causate dalla sclerosi multipla, visibili tramite risonanza magnetica.

L'implementazione è stata sviluppata in linguaggio Python, utilizzando le librerie TensorFlow e Keras per la definizione e l'allenamento della rete neurale, e NumPy, NiBabel e Matplotlib per la gestione, la visualizzazione e il preprocessing dei dati.

### 4.1 Analisi e preparazione del dataset

Il dataset utilizzato è pubblico ed è disponibile su Kaggle [42], contiene le risonanze magnetiche, di tipo T1, T2 e FLAIR, di 60 pazienti, che sono state segmentate e validate da tre radiologi ed un neurologo. Le immagini di tipo T1 sono ottimali per mostrare l'anatomia del paziente, le T2 consentono di distinguere più facilmente il tessuto sano da quello patologico e di evidenziare i fluidi, mentre le immagini di tipo FLAIR sopprimono il segnale dei fluidi permettendo di evidenziare al meglio le lesioni.

#### 4.1.1 Struttura del dataset

Per ognuno dei 60 pazienti sono presenti 6 file, 3 dei quali sono immagini di risonanze, gli altri 3 sono le maschere che indicano dove sono presenti le lesioni per ogni tipologia di immagine.

Ogni immagine ha tre dimensioni: altezza, larghezza e profondità. La dimensione rappresenta la qualità delle immagini e dipende dall'intensità del campo magnetico che la macchina è in grado di generare, per ottenere immagini dettagliate del cervello generalmente si utilizzano macchinari ad alto campo magnetico, da 1.5 Tesla o superiori.

Nel caso di studio, le immagini delle risonanze magnetiche sono state ottenute utilizzando macchinari con Tesla diversi tra loro, perciò la minima dimensione presente è  $224 \times 176 \times 15$ , mentre la massima è  $1024 \times 1024 \times 33$ .

## Plot delle immagini con rispettive maschere

Di seguito sono mostrati i plot delle immagini del paziente n°6, per ogni tipologia, come campione rappresentativo della struttura delle immagini, delle lesioni e delle maschere. Le immagini sono plottate ad altezze differenti, perciò non sono riportate le stesse sezioni del cervello.

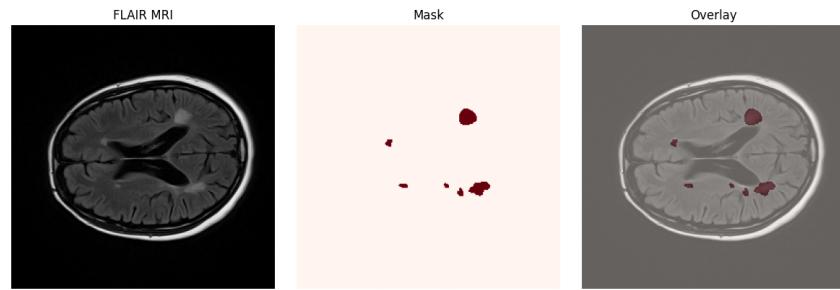


Figura 4.1: Immagine di risonanza magnetica di tipo FLAIR del paziente n°6 con la rispettiva maschera. A sinistra è visibile l'immagine originale senza maschera applicata, in cui le zone più chiare del cervello corrispondono alle lesioni. Al centro è visibile la maschera che indica la posizione delle lesioni, a destra la sovrapposizione della maschera sull'immagine.

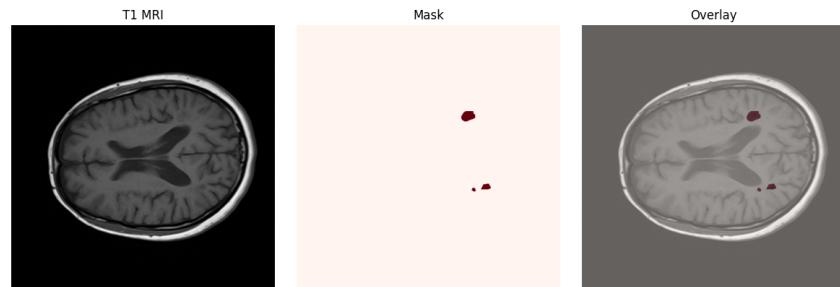


Figura 4.2: Immagine di risonanza magnetica di tipo T1 del paziente n°6 con la rispettiva maschera. A sinistra è mostrata l'immagine originale di tipo T1 senza maschera applicata, si nota il fatto che l'immagine sia più scura rispetto alla precedente e quindi le lesioni siano meno visibili, la struttura del plot al centro e a destra è identica alla precedente.

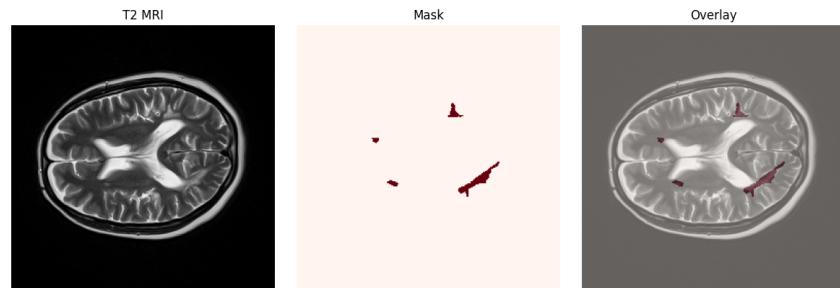


Figura 4.3: Risonanza magnetica di tipo T2 del paziente n°6 con la rispettiva maschera. Anche in questo caso la struttura è come nei casi precedenti, si nota la maggiore luminosità dell'immagine a sinistra, dovuta al segnale dei fluidi.

### 4.1.2 Preparazione del dataset

Per preparare il dataset, vengono convertite le immagini dal formato NIfTI (.nii) in file NumPy compressi (.npz) al fine di renderli più leggeri e gestibili dal modello.

Per ciascun paziente, si ridimensiona ogni slice alla dimensione  $128 \times 128$  e si normalizza l'intensità di ogni pixel volumetrico, dividendo i valori per l'intensità massima del volume, al fine di uniformare la scala dei dati tra i diversi pazienti. Inoltre, vengono incluse solo le slice contenenti almeno un pixel appartenente ad una lesione e vengono scartate il 25% delle prime e delle ultime slice così da ridurre il contenuto di dati non informativo.

In seguito si utilizzano tecniche di data augmentation, tra cui rotazione, flip orizzontali o verticali, crop casuali, modifiche di luminosità e aggiunta di rumore. Ciò consente di migliorare la generalizzazione del modello e di ridurre l'overfitting.

Infine il dataset viene diviso in:

- Training set, a cui viene assegnato l'80% del dataset, 48 pazienti
- Validation set, a cui spetta un 10% del dataset, 6 pazienti
- Test, il restante 10%, 6 pazienti

## 4.2 Implementazione del modello

Per la segmentazione delle lesioni da sclerosi multipla, si utilizza una rete neurale convoluzionale di tipo U-Net 2D.

La U-Net 2D tratta ogni slice del volume 3D come un'immagine separata, lavorando quindi su una slice alla volta. Ciò consente semplicità e velocità dell'allenamento, e un costo computazionale contenuto rispetto all'utilizzo della U-Net 3D, la quale garantirebbe un'accuratezza maggiore nella segmentazione delle lesioni.

Per lo scopo di questa tesi la scelta ottimale ricade sulla U-Net 2D, trovando un buon compromesso tra costo computazionale e precisione della segmentazione.

### 4.2.1 Struttura della U-Net 2D

La struttura della U-Net utilizzata è composta da: encoder, bottleneck, decoder e output, di seguito riportati.

#### Encoder

L'encoder è costituito da quattro blocchi, ciascuno composto da:

- Due convoluzioni  $3 \times 3$  seguite da:
  - Batch normalization, al fine di stabilizzare e accelerare l'allenamento

- ReLU, come funzione di attivazione
- Max pooling 2D, per dimezzare la dimensione spaziale
- Dropout, per ridurre l'overfitting

## Bottleneck

Il bottleneck è il blocco più profondo della rete ed è caratterizzato da una rappresentazione altamente astratta delle feature.

La struttura è simile a quella dell'encoder, con l'unica differenza che il bottleneck non presenta il max pooling;

- Due convoluzioni  $3 \times 3$  seguite da:
  - Batch normalization
  - ReLU
- Dropout

## Decoder

Come l'encoder, anche il decoder è composto da quattro blocchi, ognuno strutturato come segue:

- Una convoluzione trasposta bidimensionale, che aumenta le dimensioni spaziali
- Una skip connection, che concatena le feature map del decoder con le corrispondenti dell'encoder
- Due convoluzioni  $3 \times 3$  seguite da:
  - Batch normalization
  - ReLU

## Output

Infine, l'output della rete è composto da:

- Una convoluzione  $1 \times 1$
- Funzione di attivazione sigmoide, la quale restituisce 0 in caso di tessuto sano, 1 in caso di lesione.

La Tabella 4.1 riporta la struttura della rete.

Layer (type)	Output Shape	Parametri	Descrizione/Connessioni
<b>InputLayer</b>	(None, 128, 128, 1)	0	Immagine in scala di grigi
<i>Encoder</i>			
Conv2D + BN + ReLU ( $\times 2$ )	(128, 128, 32)	9,696	Estrazione caratteristiche di basso livello
MaxPooling2D + Dropout	(64, 64, 32)	0	Riduzione dimensionale
Conv2D + BN + ReLU ( $\times 2$ )	(64, 64, 64)	55,808	Feature map di livello intermedio
MaxPooling2D + Dropout	(32, 32, 64)	0	
Conv2D + BN + ReLU ( $\times 2$ )	(32, 32, 128)	222,464	Feature map più profonde
MaxPooling2D + Dropout	(16, 16, 128)	0	
Conv2D + BN + ReLU ( $\times 2$ )	(16, 16, 256)	885,376	Rappresentazioni ad alto livello
MaxPooling2D + Dropout	(8, 8, 256)	0	
<i>Bottleneck</i>			
Conv2D + BN + ReLU ( $\times 2$ )	(8, 8, 512)	3,542,016	Strato più profondo (massima astrazione)
Dropout	(8, 8, 512)	0	
<i>Decoder</i>			
Conv2DTranspose + Concatenate	(16, 16, 512)	524,544	Skip connection con blocco encoder 4
Conv2D + BN + ReLU ( $\times 2$ )	(16, 16, 256)	1,770,984	
Conv2DTranspose + Concatenate	(32, 32, 256)	131,200	Skip connection con blocco encoder 3
Conv2D + BN + ReLU ( $\times 2$ )	(32, 32, 128)	442,624	
Conv2DTranspose + Concatenate	(64, 64, 128)	32,832	Skip connection con blocco encoder 2
Conv2D + BN + ReLU ( $\times 2$ )	(64, 64, 64)	111,232	
Conv2DTranspose + Concatenate	(128, 128, 64)	8,224	Skip connection con blocco encoder 1
Conv2D + BN + ReLU ( $\times 2$ )	(128, 128, 32)	27,712	
<b>Output (Conv2D, sigmoid)</b>	(128, 128, 1)	33	Maschera di segmentazione binaria
<b>Totale parametri</b>	<b>7,771,299</b>		
<b>Parametri addestrabili</b>	7,765,409		
<b>Parametri non addestrabili</b>	5,888		

Tabella 4.1: Architettura del modello U-Net utilizzato

### 4.3 Allenamento della U-Net

L’allenamento della rete è impostato su 200 epoch, con un learning rate di  $10^{-4}$ .

Inoltre, si utilizzano i callback, delle funzioni che vengono eseguite durante l’addestramento. In particolare i callback servono a:

- Salvare il modello quando ottiene prestazioni migliori sulla funzione di costo
- Interrompere l’allenamento se la funzione di costo non migliora dopo un certo numero di epoch, in questo caso 20
- Dimezzare automaticamente il learning rate quando la funzione di costo smette di diminuire dopo 5 epoch

Nel momento in cui viene compilato il modello, oltre alle metriche descritte in seguito, come ottimizzatore viene scelto Adam.

### 4.3.1 Metriche utilizzate

#### Dice coefficient e Dice loss

Come anticipato nel capitolo dedicato all’allenamento delle reti neurali, il Dice coefficient misura la sovrapposizione tra la maschera predetta dalla rete e la maschera reale. La maschera è un’immagine binaria che indica la presenza della lesione.

$$Dice = \frac{2TP}{2 \times TP + FN + FP}$$

dove  $TP$  è il numero di pixel d’interesse classificati correttamente (True Positives),  $FN$  sono i piexel d’interesse non rilevati (False Negatives) e  $FP$  sono i False Negatives.

Dice Loss è la funzione di costo, che si vuole minimizzare.

$$DiceLoss = 1 - Dice$$

#### Intersection over Union - IoU

L’Intersection over Union è una metrica simile al Dice, in quanto entrambi confrontano la sovrapposizione tra la maschera predetta e la maschera reale.

Tuttavia, l’IoU penalizza di più gli errori poichè non raddoppia il termine al numeratore, che rappresenta l’intersezione, è perciò una stima più veritiera delle prestazioni del modello.

$$IoU = \frac{TP}{TP + FN + FP}$$

#### Sensitivity

La Sensitivity indica la capacità del modello di identificare i pixel appartenenti alla classe positiva; più è alta, meno pixel di interesse vengono persi.

$$Sensitivity = \frac{TP}{TP + FN}$$

#### Specificity

La Specificity misura la capacità del modello di riconoscere i pixel appartenenti alla classe negativa; più è alta, meno errori vengono commessi.

$$Specificity = \frac{TN}{TN + FP}$$

## 4.4 Validation e test

Al fine di avere una visione completa delle prestazioni del modello, vengono riportati i valori assunti dalle metriche utilizzate per una valutazione quantitativa e le immagini con le relative maschere, sia reali sia predette dalla rete.

### 4.4.1 Valutazione basata sulle metriche - Validation

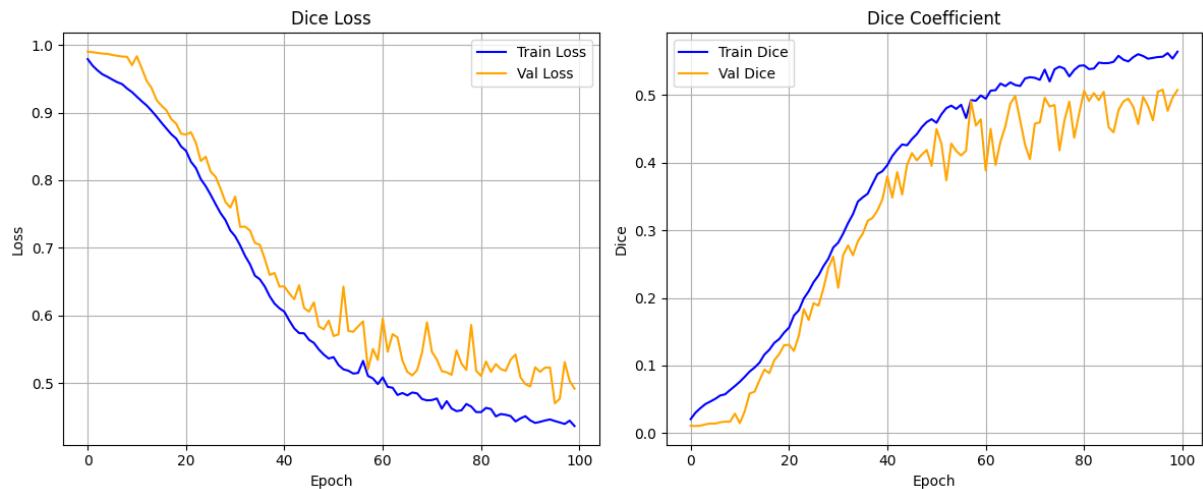


Figura 4.4: Andamenti della funzione di costo Dice loss e del Dice coefficient

Nella Figura 4.4, sono riportati gli andamenti della funzione di costo e del Dice coefficient durante le fasi di training e validation.

Dal grafico della Dice Loss si nota che, durante il training, la curva descresce fino a 0.4, indicando che il modello migliora nell'apprendimento. Durante il validation, invece, la loss raggiunge all'incirca 0.5, in linea con quanto ci si aspetta.

In modo analogo, nel grafico del Dice coefficient, si nota che durante il training si ottengono risultati migliori, infatti, la curva del Dice cresce fino a 0.6, mentre nel validation arriva ad un massimo di 0.5.

Questa differenza tra training e validation è attesa, in quanto il modello tende naturalmente a performare meglio sui dati di allenamento.

Dice	DiceLoss	IoU	Sensitivity	Specificity
0.516	0.486	0.350	0.507	0.997

Tabella 4.2: Valori assunti dalle metriche durante il validation

Questi valori implicano un errore moderato, compatibile con il modello U-Net utilizzato, la capacità di calcolo disponibile, e le limitazioni del dataset disponibile. La rete è in grado di riconoscere una parte significativa delle lesioni, tralasciando alcuni pixel positivi, come indicato dalla Sensitivity appena superiore a 0.5. D'altra parte, la Specificity elevata indica che la rete classifica correttamente la maggiorparte dei pixel negativi.

#### 4.4.2 Valutazione basata sui plot - Test

Come anticipato, la rete riconosce buona parte delle lesioni, riuscendo ad individuare, nella maggior parte dei casi, la regione del cervello in cui si trovano. Ciò indica che la segmentazione presenta una buona coerenza spaziale, identificando le zone interessate e delineando le lesioni principali. Tuttavia, in qualche caso, la rete tende a tralasciare alcuni pixel, specialmente in presenza di lesioni particolarmente ridotte, o che presentano un contrasto meno marcato rispetto al tessuto circostante.

Dalle Figure 4.5 e 4.6 si osserva che il modello riconosce facilmente le lesioni di dimensioni maggiori, riuscendo a individuare la maggior parte dell'area lesionata e restituire una segmentazione complessivamente accurata.

Inoltre, è rilevante notare che in questo caso la rete è in grado di rilevare anche lesioni di dimensioni inferiori.

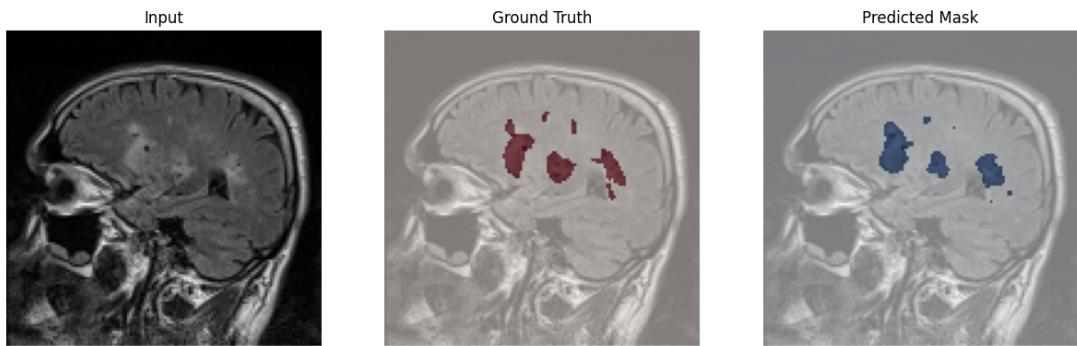


Figura 4.5: Risonanza magnetica orientata sul piano sagittale

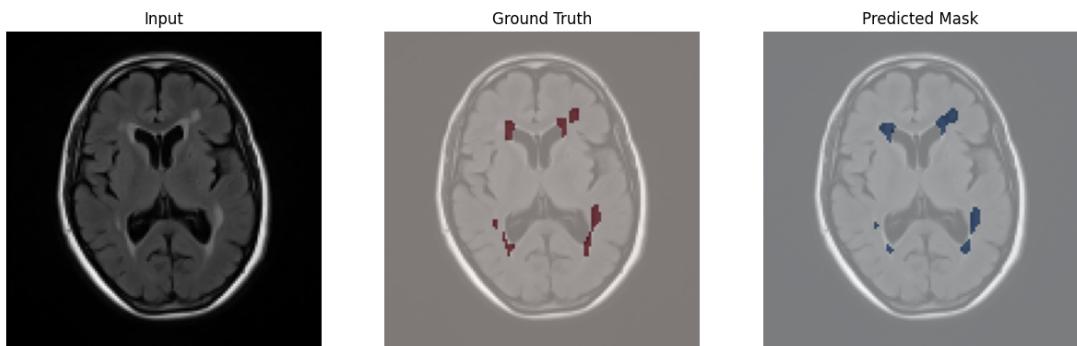


Figura 4.6: Risonanza magnetica orientata sul piano trasversale

Successivamente, sono riportati ulteriori plot del test che mostrano in maniera evidente il fenomeno descritto in precedenza.

In particolare, nella Figura 4.7 sono riportati i casi considerati positivi, in cui la rete è stata in grado di individuare con buona accuratezza le lesioni, incluse quelle di piccole dimensioni. Al contrario, nella Figura 4.8 sono riportati i plot in cui le prestazioni del modello risultano inferiori, poiché la rete non riesce ad individuare correttamente le lesioni presenti.

## Test con esito positivo

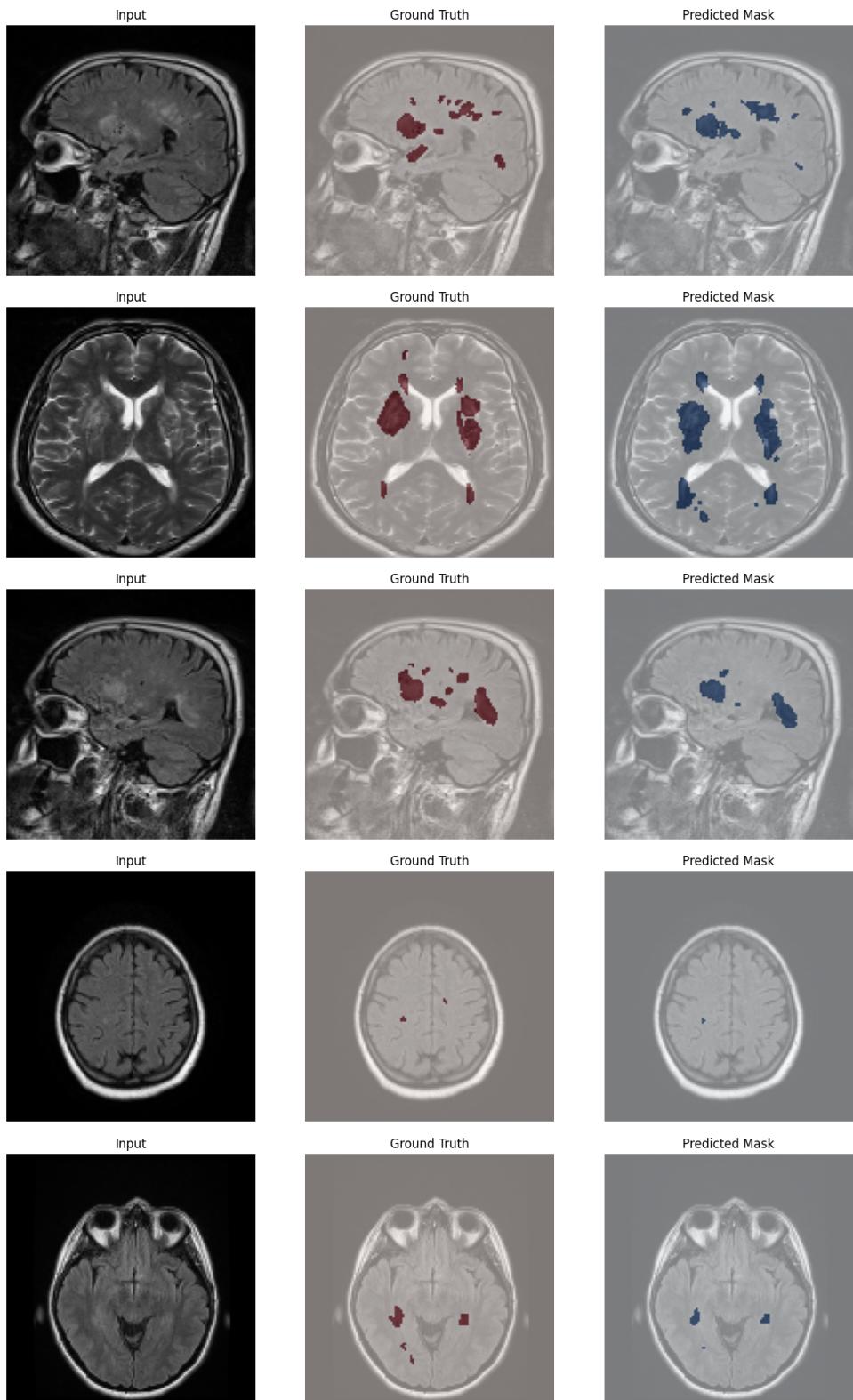


Figura 4.7

## Test con esito discreto

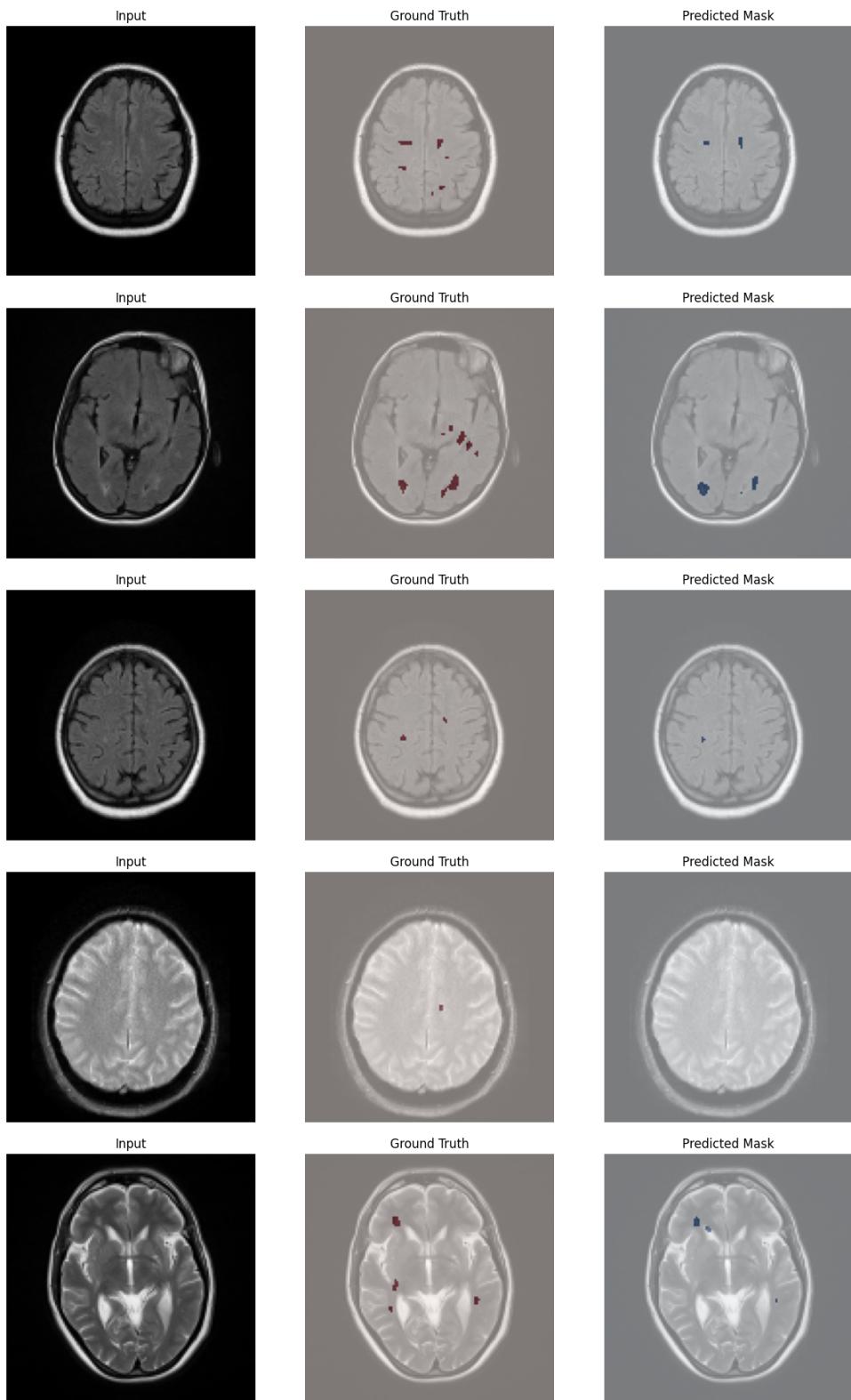


Figura 4.8

## 4.5 Studi rilevanti sulla segmentazione

Negli ultimi anni il tema della segmentazione delle lesioni da sclerosi multipla, è già stato affrontato varie volte.

In questa tesi si riportano tre studi rilevanti, al fine di confrontare il lavoro svolto con lo stato dell'arte.

### 4.5.1 GAU U-Net

Il primo studio considerato è stato condotto da Roba Gamal et al.[43].

In questo lavoro viene utilizzata la rete GAU U-NET, un'unione tra la classica U-Net e la Global Attention Upsample, mostrata in Figura 4.9, e viene utilizzato il dataset fornito da Open MS Data. La GAU fa parte delle tecniche di attenzione, ossia metodi che aiutano il modello a concentrarsi sulle caratteristiche più importanti e sulle relazioni tra le caratteristiche nei diversi strati.

Il modulo GAU si basa innanzitutto sul global average pooling per fornire un contesto globale e successivamente viene eseguita una convoluzione  $3 \times 3$  sulle feature di basso livello. Il contesto globale passa attraverso una convoluzione  $1 \times 1$  con batch normalization e attivazione ReLU, per poi essere moltiplicato per le caratteristiche di basso livello.

Infine, le feature di alto livello vengono sommate alle caratteristiche di basso livello pesate e viene eseguito l'upsample.

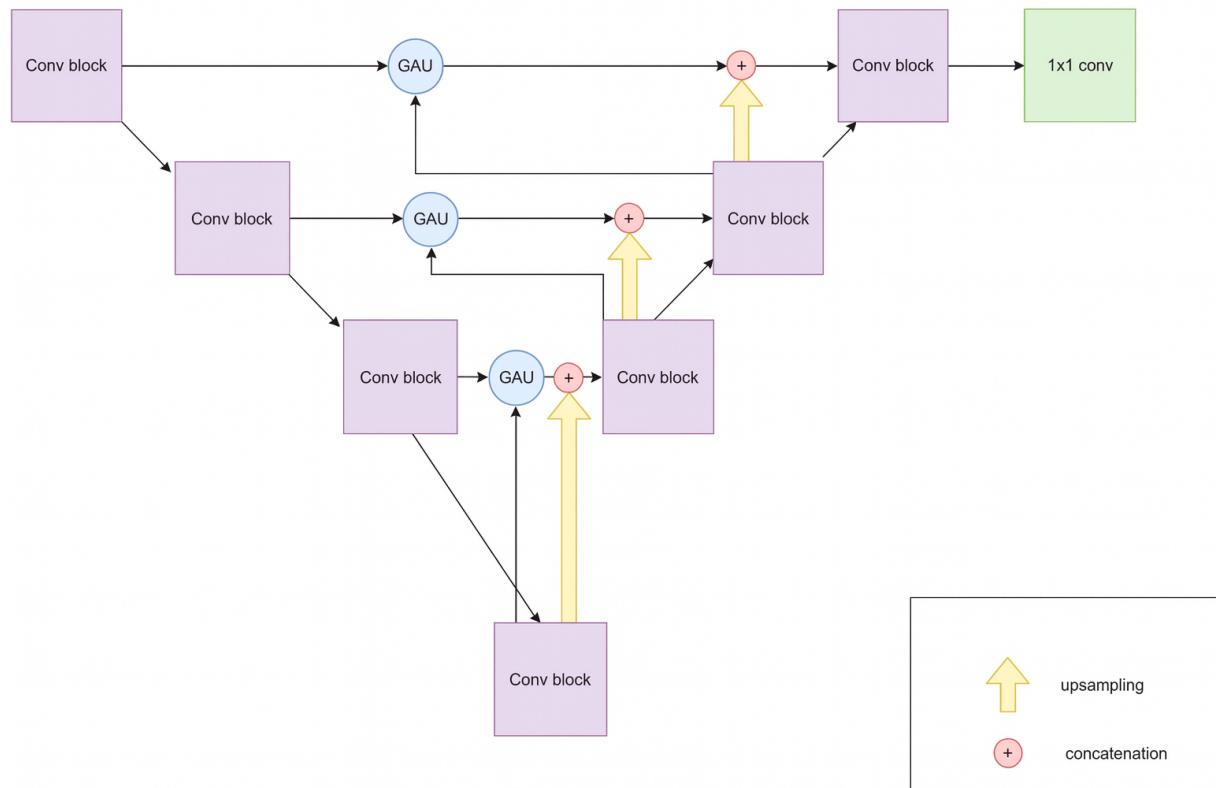


Figura 4.9: GAU U-Net [43]

L'unità GAU viene quindi implementata per migliorare le prestazioni del modello. Nello studio vengono utilizzate due tipologie di architetture: ad **alto livello** e a **basso livello**. Nell'architettura ad alto livello viene omessa la terza unità GAU, ossia quella collocata nella parte superiore della Figura 4.9. Nell'architettura a basso livello, invece, viene rimossa la prima unità GAU, posizionata alla fine del percorso di codifica in Figura 4.9. Le due architetture vengono quindi confrontate evidenziando come il GAU a basso livello consenta una notevole riduzione dei parametri utilizzati rispetto al GAU ad alto livello. Inoltre, vengono confrontati i valori medi del Dice, al fine di valutare l'efficacia dei due approcci.

Metodo	Dice medio	Numero di parametri
High level GAU	0.70	38 530 497
Low level GAU	0.72	28 883 649

Tabella 4.3: Confronto tra le prestazioni delle architetture utilizzate

#### 4.5.2 A dense residual U-Net

Un secondo studio rilevante è stato condotto da Beytullah Sarica et al.[44].

In questo lavoro viene modificata la U-Net al fine di migliorare le prestazioni del modello nella segmentazione delle lesioni. Nello specifico viene costruita una Dense Residual U-Net con AG, ECA e ASPP.

In questa variante di U-Net ogni blocco è connesso con quello successivo tramite una dense connection, oltre alla connessione della classica U-Net, consentendo che ogni strato riceva in ingresso tutte le feature map degli strati precedenti, migliorando il flusso di informazioni.

Gli AG (Attention Gates) vengono inseriti lungo le skip connections come in Figura 4.9. ECA (Efficient Channel Attention) è un meccanismo di attenzione che valorizza i canali più importanti delle feature map, insegnando alla rete a dare pesi maggiori a canali più informativi.

L'ASPP (Atrous Spatial Pyramid Pooling) è un modulo che permette l'integrazione spaziale multi-scala, fondamentale per problemi tridimensionali, come nel caso delle lesioni. Al fine di migliorare ulteriormente il modello viene introdotta la batch normalization, lo spatial dropout e la Exponential Linear Unit (ELU) al posto della funzione di attivazione ReLU.

I dataset adottati per l'analisi sono forniti dall'ISBI2015 challenge e dalla MSSEG2016 challenge. Entrambi i dataset vengono utilizzati per l'allenamento separatamente e in seguito viene eseguita una cross dataset validation, allenando il modello completo sul dataset dell'ISBI2015 e testato sul dataset della MSSEG2016, e viceversa.

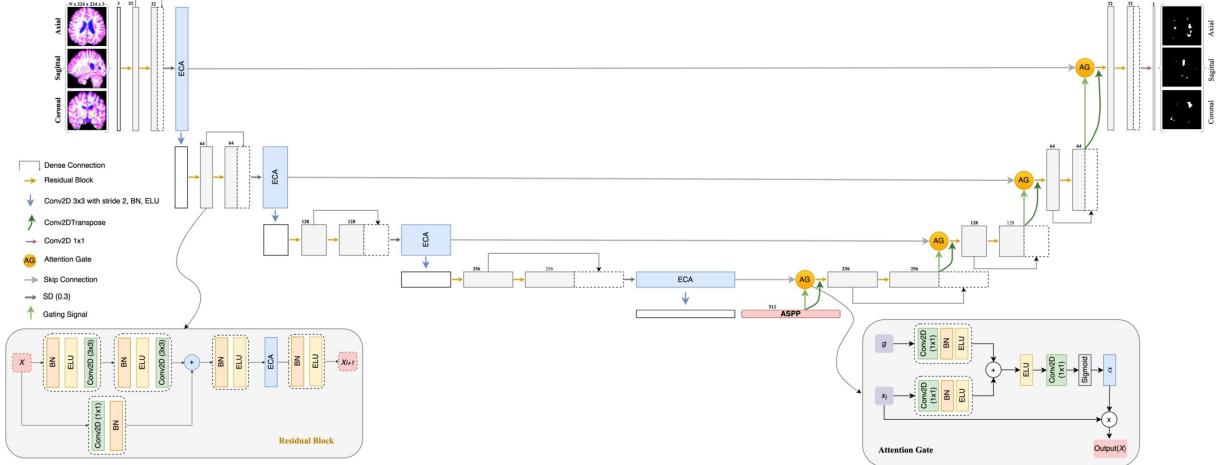


Figura 4.10: Dense residual U-Net [44]

In questo studio vengono allenate diverse varianti del modello proposto in Figura 4.10. Nella Tabella 4.4 vengono riportati i valori del Dice ottenuti.

Modello	Dice
Dense Residual U-Net with AG, ECA, and ASPP (ISBI2015)	0.6688
Dense Residual U-Net with AG and ECA	0.6653
Dense Residual U-Net with AG	0.6653
Dense Residual U-Net	0.6629
Dense U-Net	0.6652
U-Net	0.6613
Dense Residual U-Net with AG, ECA, and ASPP (MSSEG2016)	0.6727

Tabella 4.4: Valori del Dice per diversi modelli utilizzati

Nella Tabella 4.5 vengono riportati i valori del Dice in seguito al modello allenato su ISBI2015 e testato su MSSEG2016, e viceversa.

Allenamento	Test	Dice
ISBI2015	MSSEG2016	0.6031
MSSEG2016	ISBI2015	0.4819

Tabella 4.5: Valori del Dice utilizzando la cross data validation

### 4.5.3 A modified U-Net attention

Infine, un terzo studio rilevante è stato condotto da Maryam Hashemi et al.[45]. In modo analogo agli studi citati precedentemente, anche in questo lavoro viene utilizzata e modificata la rete U-Net al fine di ottenere una segmentazione migliore. Gli autori propongono due tipologie di U-Net, inizialmente viene modificata la funzione di attivazione, utilizzando la ELU al posto della ReLU, ad eccezione dell'ultimo strato, in seguito viene introdotta anche l'attention nelle skip connections, come in Figura 4.9.

Il dataset di riferimento è quello proposto dalla ISBI2015 challenge, ed è importante sottolineare che non vengono utilizzate le immagini di tipo T1, vengono eliminati i margini neri delle immagini al fine di ridurre il tempo di addestramento ed enfatizzare le aree delle lesioni.

Inoltre, è stato proposto un approccio basato sull'unione e sull'intersezione dei risultati delle due reti, per ciascuna rete ci sono perciò diversi casi a seconda delle immagini utilizzate:

- Solo immagini FLAIR utilizzate per l'allenamento e il test della rete
- Solo immagini T2
- Entrambe FLAIR e T2
- Intersezione
- Unione delle maschere stimate ottenute da FLAIR e T2

Nella Tabella 4.6 sono riportati i casi di studio.

<b>Tipo di immagine</b>	<b>Modified U-Net</b>	<b>Modified Attention U-Net</b>
FLAIR	Caso 1	Caso 6
T2	Caso 2	Caso 7
FLAIR + T2	Caso 3	Non Considerato
Intersezione	Caso 4	Caso 8
Unione	Caso 5	Caso 9

Tabella 4.6: Casi di studio

Nella Tabella 4.7 vengono mostrati i risultati dello studio in base alla tipologia di rete utilizzata e di immagini, a differenza degli studi citati in precedenza, oltre al Dice, vengono riportati anche la Sensitivity e la IoU.

<b>Network, Data</b>	<b>Dice</b>	<b>Sensitivity</b>	<b>IoU</b>
U-Net, FLAIR (Case 1)	0.8144	0.7984	0.6961
U-Net, T2 (Case 2)	0.8078	0.7814	0.6891
U-Net, FLAIR + T2 (Case 3)	0.8066	0.7705	0.6886
U-Net, Intersection (Case 4)	0.8044	0.7319	0.6838
U-Net, Union (Case 5)	0.8199	0.8527	0.7042
Attention U-Net, FLAIR (Case 6)	0.8031	0.7933	0.6830
Attention U-Net, T2 (Case 7)	0.8179	0.7915	0.7003
Attention U-Net, Intersection (Case 8)	0.7954	0.7198	0.6726
Attention U-Net, Union (Case 9)	0.8230	0.8650	0.7081

Tabella 4.7: Dice, Sensitivity e IoU per rete e tipo di dato.

#### 4.5.4 Confronto con gli studi riportati

Confrontando il lavoro svolto con lo stato dell’arte, emerge come una maggiore capacità computazionale, l’utilizzo di modelli più complessi e l’accesso a dataset forniti da enti riconosciuti possano condurre a prestazioni migliori.

In particolare nello studio condotto da Gamal et al.[43] vengono introdotti gli attention gate, che consentono alla rete di apprendere in modo più efficace le caratteristiche più rilevanti. Il dataset utilizzato dagli autori è l’Open MS del Laboratory of Imaging Technologies.

I risultati dello studio, riportati nella Tabella 4.3, sono promettenti: a seconda della variante di GAU U-Net considerata, viene ottenuto un valore di Dice pari 0.70 per l’architettura di alto livello, e un Dice di 0.72 per quella di basso livello. Nonostante non vengano fornite ulteriori metriche di valutazione, tali valori indicano una segmentazione accurata delle lesioni, con una capacità del modello di riconoscere la maggior parte dell’area interessata dalle lesioni.

Nel secondo studio riportato, condotto da Sarica et al.[44], il modello utilizzato è una Dense Residual U-Net arricchita con AG, ECA e ASPP, risultando in un’architettura decisamente più complessa rispetto a quella analizzata nel caso precedente. Gli autori impiegano due dataset differenti, al fine di confrontare le prestazioni del modello e vengono allenate diverse varianti del modello per confrontarne le prestazioni.

I risultati ottenuti, riportati nella Tabella 4.4, sono complessivamente molto buoni, anche se con valori di Dice leggermente inferiori allo studio citato in precedenza. Il valore migliore, pari a 0.6727, è stato raggiunto utilizzando il dataset di MSSEG2016, evidenziando l’importanza di poter disporre di dati di alta qualità.

La rete completa viene successivamente allenata sul dataset di ISBI2015 e testata sul MSSEG2016, e viceversa, ottenendo migliori risultati nel primo caso, con un Dice pari a 0.6031 rispetto a 0.4819, sottolineando l’importanza della coerenza tra i dati di training e di test per ottenere risultati migliori.

Infine, nel terzo studio riportato, condotto da Hashemi et al.[45], viene proposta una variante della rete U-Net, modificata attraverso l’introduzione degli attention gate e la sostituzione della funzione di attivazione ReLU con la ELU. Il dataset utilizzato dagli autori proviene dall’ISBI2015 challenge, ma con una differenza significativa rispetto allo studio precedente, in questo caso non vengono impiegate le immagini di risonanza magnetica di tipo T1, nelle quali le lesioni sono meno visibili a causa del ridotto contrasto con i tessuti circostanti.

Al fine di confrontare le prestazioni della rete, gli autori utilizzano diverse varianti del modello e diverse modalità di organizzazione dei dati. I risultati, riportati nella Tabella 4.7, sono significativamente superiori rispetto a quelli degli studi precedentemente citati. Il valore migliore, pari a 0.8230, è stato ottenuto utilizzando la versione della U-Net

dotata di attention gate e unendo le maschere stimate dalle immagini di tipo FLAIR e T2. Questo evidenzia come le prestazioni del modello dipendano significativamente dal dataset utilizzato e in particolare dalla tipologia di immagini: un maggiore contrasto tra lesioni e tessuti circostanti facilita il riconoscimento delle aree caratterizzate da lesioni da parte della rete, contribuendo ad ottenere una segmentazione più accurata.

# Capitolo 5

## Conclusioni

Lo scopo di questa tesi era applicare le reti neurali, e in particolare l’architettura U-Net, allo studio della sclerosi multipla, al fine di rendere più efficiente il processo diagnostico tramite una segmentazione automatica delle lesioni tipiche della patologia.

Si è scelto di utilizzare una rete neurale convoluzionale di tipo U-Net, poichè ampiamente riconosciuta come una delle architetture più efficaci per l’analisi e la segmentazione delle immagini biomediche.

Valutando la capacità computazionale disponibile per questa tesi, la configurazione più adeguata è risultata essere la U-Net 2D nella sua versione base, che ha permesso di ottenere un buon compromesso tra tempi di allenamento e accuratezza del modello.

Per agevolare il processo di addestramento, ogni immagine è stata ridimensionata alla risoluzione  $128 \times 128$ , accettando una parziale perdita di informazioni a favore di tempi di allenamento minori. Al fine di migliorare le prestazioni complessive del modello, sono state inoltre applicate tecniche di data augmentation per aumentare la variabilità degli esempi, insieme all’introduzione di batch normalization e dropout per migliorare la stabilità dell’addestramento e ridurre il rischio di overfitting.

Il dataset utilizzato è pubblico e liberamente accessibile su Kaggle [42].

I risultati ottenuti, riportati nella Tabella 4.2, con un valore di Dice pari a 0.5 e un IoU pari a 0.35, mostrano che il modello è in grado di riconoscere le lesioni, in particolare quelle di dimensioni maggiori, tralasciando però alcune porzioni delle lesioni che è riuscito ad individuare. Le metriche risultano comunque coerenti con il modello utilizzato e con le caratteristiche del dataset disponibile.

In conclusione, attraverso il lavoro svolto in questa tesi si dimostra l’importanza di disporre di una capacità computazionale adeguata all’allenamento delle reti neurali convoluzionali, le quali, operando su immagini, presentano un costo computazionale elevato.

Allo stesso tempo, si evidenzia l’importanza della qualità del dataset utilizzato, infatti, dataset completi e ben bilanciati forniscono alla rete informazioni affidabili, migliorando sensibilmente le prestazioni finali del modello.

Fondamentale è anche la tipologia di immagini impiegata: scegliere immagini con un

contrastò elevato tra lesioni e tessuti circostanti permette alla rete di apprendere più facilmente le caratteristiche delle aree interessate dalle lesioni, aumentando la precisione della segmentazione.

Pertanto, l'impiego delle reti neurali per la segmentazione automatica delle lesioni della sclerosi multipla, rappresenta un approccio promettente, un possibile strumento di supporto nel processo diagnostico. Grazie alla capacità di analizzare rapidamente grandi quantità di immagini e di individuare le aree patologiche, questi modelli potrebbero ridurre significativamente i tempi di valutazione clinica. Al contempo, vi è margine per ulteriori miglioramenti: l'ottimizzazione delle architetture di rete e l'incremento della qualità e della quantità dei dataset disponibili, possono contribuire a rendere i modelli più precisi, robusti e generalizzabili. Tali sviluppi potrebbero portare a sistemi diagnostici sempre più affidabili, capaci di affiancare efficacemente il lavoro svolto dal personale sanitario.

# Bibliografia

- [1] Subana Shanmuganathan e Sandhya Samarasinghe, cur. *Artificial Neural Network Modelling*. Vol. 628. Studies in Computational Intelligence. Cham: Springer, 2016, pp. vii, 472. ISBN: 978-3-319-28493-4.
- [2] Wikipedia contributors. *Rete neurale artificiale — Wikipedia, l'enciclopedia libera*. Accesso: 2025-10-08. 2025. URL: [https://it.wikipedia.org/wiki/Rete\\_neurale\\_artificiale](https://it.wikipedia.org/wiki/Rete_neurale_artificiale) (visitato il giorno 08/10/2025).
- [3] Wikipedia contributors. *Neurone — Wikipedia, l'enciclopedia libera*. Accesso: 2025-10-08. 2025. URL: <https://it.wikipedia.org/wiki/Neurone> (visitato il giorno 08/10/2025).
- [4] Humanitas. *Come funziona il nostro cervello?* Accesso: 2025-10-08. 2022. URL: <https://www.humanitas.it/news/come-funziona-il-nostro-cervello/> (visitato il giorno 08/10/2025).
- [5] Davide Manca. *Reti Neurali Artificiali*. Lezione disponibile sul sito PSELab, Politecnico di Milano, accesso: 2025-10-08. 2014. URL: <https://pselab.chem.polimi.it/wp-content/uploads/2014/03/DECDPC/Lez-05-Reti-Neurali-Artificiali.pdf>.
- [6] Freepik. *Diagramma dell'anatomia del neurone*. Accesso: 5 October 2025. 2018. URL: [https://it.freepik.com/vettori-premium/diagramma-dell-anatomia-del-neurone\\_2480497.htm](https://it.freepik.com/vettori-premium/diagramma-dell-anatomia-del-neurone_2480497.htm).
- [7] Electroyou. *Reti Neurali*. Accesso: 5 October 2025. 2009. URL: [https://www.electroyou.it/c1b8/wiki/rete\\_neurale](https://www.electroyou.it/c1b8/wiki/rete_neurale).
- [8] Vito Lavecchia. *Differenza tra deep learning e rete neurale*. Accessed: 2025-11-12. 2021. URL: <https://vitolavecchia.altervista.org/differenza-tra-deep-learning-e-rete-neurale/>.
- [9] Associazione Italiana Sclerosi Multipla. *Cosa è la sclerosi multipla*. Ultimo aggiornamento: 27 maggio 2025. 2025. URL: [https://www.aism.it/cosa\\_e\\_la\\_sclerosi\\_multipla](https://www.aism.it/cosa_e_la_sclerosi_multipla) (visitato il giorno 04/10/2025).

- [10] Yahya Naji et al. *Artificial Intelligence and Multiple Sclerosis: Up-to-Date Review*. PMC Article No. PMC10581506. 2023. URL: <https://pmc.ncbi.nlm.nih.gov/articles/PMC10581506/> (visitato il giorno 15/11/2025).
- [11] Moein Amin et al. *Artificial Intelligence and Multiple Sclerosis*. Curr Neurol Neurosci Rep. 2024 Aug;24(8):233-243.  
doi:10.1007/s11910-024-01354-x. PMCID: PMC11258192. 2024. URL: <https://pmc.ncbi.nlm.nih.gov/articles/PMC11258192/> (visitato il giorno 15/11/2025).
- [12] Neural Networks e Deep Learning. *Chapter 1: Using Neural Networks to Recognize Handwritten Digits — Sigmoid Neurons*. Accesso: 2025-10-11. 2015. URL: [http://neuralnetworksanddeeplearning.com/chap1.html%5C#sigmoid\\_neurons](http://neuralnetworksanddeeplearning.com/chap1.html%5C#sigmoid_neurons).
- [13] Sebai Dorsaf. *Comprehensive synthesis of the main activation functions pros and cons*. Accesso: 2025-10-15. 2020. URL: <https://medium.com/analytics-vidhya/comprehensive-synthesis-of-the-main-activation-functions-pros-and-cons-dab105fe4b3b>.
- [14] Tomasz Szandał. *Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks*. arXiv preprint arXiv:2010.09458, accesso: 2025-10-15. 2020. URL: <https://arxiv.org/pdf/2010.09458.pdf>.
- [15] GeeksforGeeks. *Stair Step Function*. Last updated: 23 Jul 2025; Accessed: 12 Nov 2025. 2025. URL: <https://www.geeksforgeeks.org/data-science/stair-step-function/>.
- [16] Wikipedia. *Sigmoid function*.  
Accesso: 2025-10-15. 2025. URL: [https://en.wikipedia.org/wiki/Sigmoid\\_function](https://en.wikipedia.org/wiki/Sigmoid_function).
- [17] MedCalc Software Ltd. *Hyperbolic tangent function*. Accesso: 2025-10-15. 2025. URL: <https://www.medcalc.org/en/manual/tanh-function.php>.
- [18] GeeksforGeeks. *ReLU Activation Function in Deep Learning*. Accesso: 2025-10-15. 2025. URL: <https://www.geeksforgeeks.org/deep-learning/relu-activation-function-in-deep-learning/>.
- [19] i2tutorials. *Explain Step / Threshold and Leaky ReLU Activation Functions*. Accesso: 2025-10-15. 2019. URL: <https://www.i2tutorials.com/explain-step-threshold-and-leaky-relu-activation-functions/>.
- [20] Dedeepya Lekkala. *Understanding Different Activation Functions*. Accessed: 2025-11-02. 2023. URL: <https://medium.com/@deepyachowdary/understanding-different-activation-functions-6aed5ed1c785>.
- [21] BotPenguin. *Softmax Function: Advantages and Applications*. Accessed: 2025-11-02. 2025. URL: <https://botpenguin.com/glossary/softmax-function>.

- [22] Daniel Jurafsky e James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. 3rd. Accesso: 2025-10-17. Pearson, 2025. URL: <https://web.stanford.edu/~jurafsky/slp3/>.
- [23] Michael A. Nielsen. *Chapter 4: A visual proof that neural nets can compute any function*. Accesso: 2025-10-11. 2015. URL: <http://neuralnetworksanddeeplearning.com/chap4.html>.
- [24] Lumen Learning. *Approximating Area*. Accesso: 17 ottobre 2025. 2025. URL: <https://courses.lumenlearning.com/calculus2/chapter/approximating-area/>.
- [25] Albert R. Ortiz Lasprilla, Juan M. Caicedo e Gustavo A. Ospina. «Modeling human–structure interaction using a closed loop control system». In: *Mechanics of Biological Systems and Materials – Proceedings of the 2013 Annual Conference on Experimental and Applied Mechanics*. Conference Proceedings of the Society for Experimental Mechanics Series, Vol. 4. Springer New York LLC, 2014, pp. 101–108. DOI: [10.1007/978-3-319-04546-7\\\_12](https://doi.org/10.1007/978-3-319-04546-7_12).
- [26] Keiron O’Shea e Ryan Nash. *An Introduction to Convolutional Neural Networks*. arXiv:1511.08458v2. 2015. URL: <https://arxiv.org/pdf/1511.08458.pdf> (visitato il giorno 28/10/2025).
- [27] Idea Verde Appia Antica. *Titolo della pagina o dell’articolo*. Accesso: 29 ottobre 2025. 2025. URL: <https://www.ideaverdeappiaantica.com/?detail/926873077>.
- [28] MathWorks. *Convolution - MATLAB & Simulink*. Accesso: 29 ottobre 2025. 2025. URL: <https://www.mathworks.com/discovery/convolution.html>.
- [29] Alex Krizhevsky, Ilya Sutskever e Geoffrey E. Hinton. «ImageNet Classification with Deep Convolutional Neural Networks». In: *Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS 2012)*. Curran Associates, Inc., 2012, pp. 1097–1105.
- [30] Wikipedia contributors. *AlexNet*. Accesso: 31 ottobre 2025. 2025. URL: <https://it.wikipedia.org/wiki/AlexNet>.
- [31] Olaf Ronneberger, Philipp Fischer e Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV]. URL: <https://arxiv.org/pdf/1505.04597.pdf>.
- [32] Alejandro Ito Aramendia. *The U-Net: A Complete Guide*. Accessed: 2025-10-31. 2024. URL: <https://medium.com/@alejandro.itoaramendia/decoding-the-u-net-a-complete-guide-810b1c6d56d8>.

- [33] Nikolas Adaloglou. *Intuitive Explanation of Skip Connections in Deep Learning*. Accessed: 2025-11-04. 2020. URL: <https://theaisummer.com/skip-connections/> (visitato il giorno 04/11/2025).
- [34] Sunny Panchal. *Optimizers*. Accesso: 2025-10-27. 2024. URL: <https://www.linkedin.com/pulse/optimizers-sunny-panchal-oen1c>.
- [35] Mohit Mishra. *The Learning Rate: A Hyperparameter That Matters*. Medium blog post, accesso: 2025-10-20. 2023. URL: <https://mohitmishra786687.medium.com/the-learning-rate-a-hyperparameter-that-matters-b2f3b68324ab>.
- [36] Ian Goodfellow, Yoshua Bengio e Aaron Courville. *Deep Learning*. <http://www.deeplearningbook.org>, accesso: 2025-10-20. MIT Press, 2016.
- [37] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. arXiv preprint arXiv:1609.04747, version v2 (15 Jun 2017), accessed on 2025-11-11. 2016. URL: <https://arxiv.org/abs/1609.04747>.
- [38] Jash Rathod. *Underfitting, Overfitting, and Regularization*. Accessed: 2025-10-23. Set. 2021. URL: <https://jashrathod.github.io/2021-09-30-underfitting-overfitting-and-regularization/>.
- [39] Sergey Ioffe e Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. arXiv preprint arXiv:1502.03167. 2015. arXiv: 1502.03167. URL: <https://arxiv.org/pdf/1502.03167> (visitato il giorno 06/11/2025).
- [40] GeeksforGeeks. *Vanishing and Exploding Gradients Problems in Deep Learning*. Last updated: 23 Jul 2025. 2025. URL: <https://www.geeksforgeeks.org/deep-learning/vanishing-and-exploding-gradients-problems-in-deep-learning/>.
- [41] Chuanqi Tan et al. *A Survey on Deep Transfer Learning*. arXiv preprint arXiv:1808.01974v1 [cs.LG], 6 Aug 2018. 2018. URL: <https://arxiv.org/pdf/1808.01974> (visitato il giorno 06/11/2025).
- [42] Orvile. *Multiple Sclerosis Brain MRI Lesion Segmentation Dataset*. Accessed: 7 November 2025. URL: <https://www.kaggle.com/datasets/orvile/multiple-sclerosis-brain-mri-lesion-segmentation/data>.
- [43] Roba Gamal e et al. *GAU U-Net for multiple sclerosis segmentation*. Consultato il 19 Novembre 2025. 2023. URL: <https://www.sciencedirect.com/science/article/pii/S1110016823003502>.

- [44] Beytullah Sarica e et al. *A dense residual U-net for multiple sclerosis lesions segmentation*. Consultato il 19 Novembre 2025. 2023. URL: <https://www.sciencedirect.com/science/article/pii/S1386505622002799>.
- [45] Maryam Hashemi, Mahsa Akhbari e Christian Jutten. *Delve into Multiple Sclerosis (MS) lesion exploration: A modified attention U-Net for MS lesion segmentation in Brain MRI*. Consultato il 19 Novembre 2025. 2022. DOI: 10.1016/j.compbiomed.2022.105402. URL: <https://www.sciencedirect.com/science/article/pii/S0010482522001949>.