

Corso di Laurea triennale in Fisica (L-30)



UNIVERSITÀ DEGLI STUDI DI MILANO

**STIMA DELLE PROPRIETÀ DI SISTEMI
OTTICI NELLE MICROONDE TRAMITE
L'UTILIZZO DI RETI NEURALI**

Relatore: Maurizio TOMASI

Correlatore: Cristian FRANCESCHET

Correlatrice: Sabrina REALINI

Tesi di Laurea di:
Eleonora GATTI
Matricola: 885664

Anno Accademico 2019/2020

Indice

1 Sistemi ottici nelle microonde	5
1.1 Utilizzo delle microonde in astrofisica	5
1.2 Diagramma di radiazione	5
1.3 Simulazione di sistemi ottici	5
2 Regressione con reti neurali	7
2.1 Machine learning e tipi di rete	7
2.2 Struttura di una rete neurale	7
3 Previsione delle proprietà di un diagramma di radiazione	9
3.1 Interpolazione	9
3.1.1 Interp2d	9
3.1.2 Curve Fit	10
3.1.3 Risultati dell'interpolazione	10
3.2 Reti neurali	13
3.2.1 Architettura della rete	13
3.2.2 Pre Training	14
3.2.3 Training	14
3.2.4 Confronto di risultati	14
4 Conclusioni	15

Capitolo 1

Sistemi ottici nelle microonde

1.1 Utilizzo delle microonde in astrofisica

1.2 Diagramma di radiazione

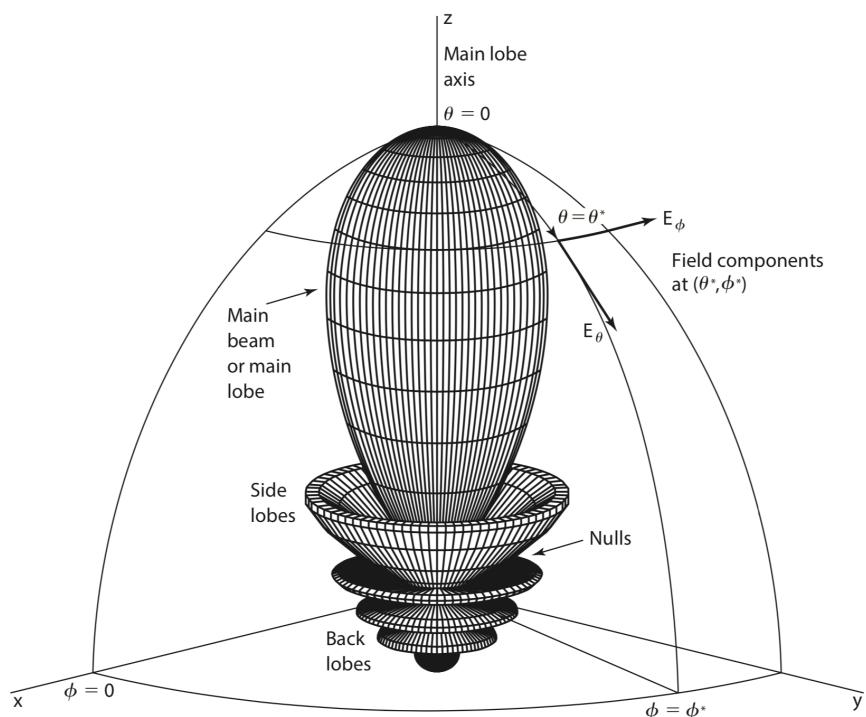


Figura 1.1: Diagramma di radiazione tridimensionale di un'antenna direzionale. [2]

1.3 Simulazione di sistemi ottici

	id	x	y	z	fwhm_x	fwhm_y	e	co_max	cx_max
0	21	-0.316965	0.326756	50	0.398546	0.365183	1.091361	53.84	13,1
1	42	-0.318192	0.272044	50	0.364137	0.398634	1.094738	53.86	12,79
2	63	-0.319138	0.217441	50	0.364608	0.396464	1.087371	53.88	12,11
3	83	-0.320197	0.163140	45	0.361428	0.384678	1.064326	54.07	12,21
4	103	-0.321053	0.108822	40	0.358714	0.376443	1.049424	54.20	11,6
...
164	3464	0.329983	-0.108774	45	0.351460	0.385294	1.096266	54.17	12,91
165	3486	0.330462	-0.163029	50	0.351985	0.395278	1.122996	54.04	13,01
166	3507	0.331717	-0.217468	50	0.351115	0.398046	1.133661	54.02	12,87
167	3528	0.333326	-0.271998	50	0.350770	0.401112	1.143520	53.99	13,38
168	3549	0.335293	-0.326679	50	0.350950	0.404582	1.152821	53.95	13,95

Figura 1.2: Dataset relativo allo strumento STRIP ottenuto tramite la simulazione in Grasp

Capitolo 2

Regressione con reti neurali

2.1 Machine learning e tipi di rete

Negli ultimi decenni il campo del *machine learning* si è fortemente evoluto. È stata costruita un'ampia classe di algoritmi in grado di approssimare molto efficacemente processi non lineari. Tali architetture fanno parte di un particolare campo di ricerca, il *Deep Learning*, che vede protagoniste le reti neurali artificiali.

Queste tecniche si basano su l'apprendimento tramite esempi, i quali sono rappresentati da coppie input-output come un'immagine e la sua descrizione (input: foto di un gatto, output: "gatto") o una posizione a cui è associato un particolare valore di campo elettrico (input: (x, y, z) , output: $E(x, y, z)$). È quindi di fondamentale importanza avere a disposizione un ampio database attraverso il quale effettuare un *training*. Le reti neurali consentono quindi di approssimare una corrispondenza, vera o presunta, tra un input e un output.

Esistono due classi di problemi affrontati con la tecnica delle reti neurali: la *classificazione* e la *regressione*. La *classificazione* individua l'appartenenza ad una classe e può essere supervisionata o non supervisionata. Parliamo di classificazione supervisionata quando sono note a priori le diverse classi di appartenenza; se invece si vogliono determinare delle classi di similitudine senza conoscere a priori i pattern rappresentativi, si ha un problema di classificazione non supervisionata.

Le reti neurali per la *regressione* entrano in gioco quando, a partire da coppie input-output, si vuole determinare la funzione che approssimi al meglio la relazione.

2.2 Struttura di una rete neurale

L'idea utopica su cui si fondano le reti neurali artificiali è quella di simulare il comportamento del cervello umano. Questo è un sistema estremamente complesso basato sull'interconnessioni di unità fondamentali: i **neuroni**.

Una rete neurale può essere schematizzata come in figura 2.1. Un insieme di neuroni che agiscono allo stesso livello è detto **layer** e l'interconnessione di diversi layers forma una rete.

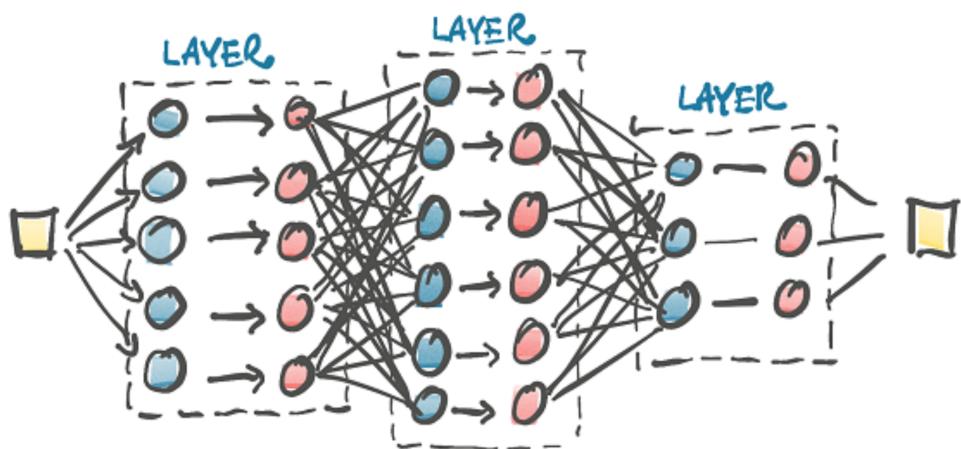


Figura 2.1: Schema di una rete neurale con tre hidden layers [1]

Capitolo 3

Previsione delle proprietà di un diagramma di radiazione

In questo capitolo varrà analizzato come sono stati previsti i parametri del diagramma di radiazione tramite i metodi di interpolazione e tramite l'utilizzo di reti neurali. Nota: nei seguenti paragrafi verranno mostrati alcuni grafici rappresentativi dei risultati ottenuti. L'analisi è stata effettuata su ogni parametro di interesse ma i risultati qui riportati riguardano esclusivamente l'ellitticità. I risultati sono infatti molto simili al variare del parametro e quelli riportati sono grafici che hanno carattere generale. I parametri analizzati sono: ellitticità, FWHM (rispetto a x), FWHM (rispetto a y), componente co-polare massima e componente cross-polare massima.

3.1 Interpolazione

Per stimare le proprietà di un diagramma di radiazione attraverso strumenti classici di interpolazione ho utilizzato due metodi: `interp2d` del modulo `scipy.interpolate`, basato su un'interpolazione lineare, e `curve fit` del modulo `scipy.optimize`, che ha permesso di fittare i dati attraverso un paraboloide. Per poter effettuare l'interpolazione dei parametri e in seguito verificare la sua bontà, il dataset in fig. 1.2 è stato suddiviso in due subsets, considerando righe alterne, che ho chiamato `data_int` e `data_check`. Così facendo è stato possibile utilizzare metà dei dati per l'interpolazione e l'altra metà per la valutazione dell'errore tra il parametro stimato e quello esatto.

3.1.1 Interp2d

Il metodo `interp2d` effettua l'interpolazione a partire da una griglia bidimensionale¹, nel mio caso questa è rappresentata dalle coppie (x, y) appartenenti al subset `data_int`. Una volta specificato il parametro di interesse, l'algoritmo effettua un'interpolazione lineare e ritorna una funzione in grado di prevedere il valore del parametro in nuovi punti.

La successiva analisi ha riguardato la valutazione dell'errore tra il parametro stimato nei punti appartenenti a `data_check` e il suo valore vero. In fig. 3.1 è mostrato l'errore assoluto relativo all'ellitticità. Si nota che i punti della prima riga a partire dall'alto sono quelli affetti da errore massimo e, come verrà specificato nella

¹La griglia non deve essere necessariamente regolare.

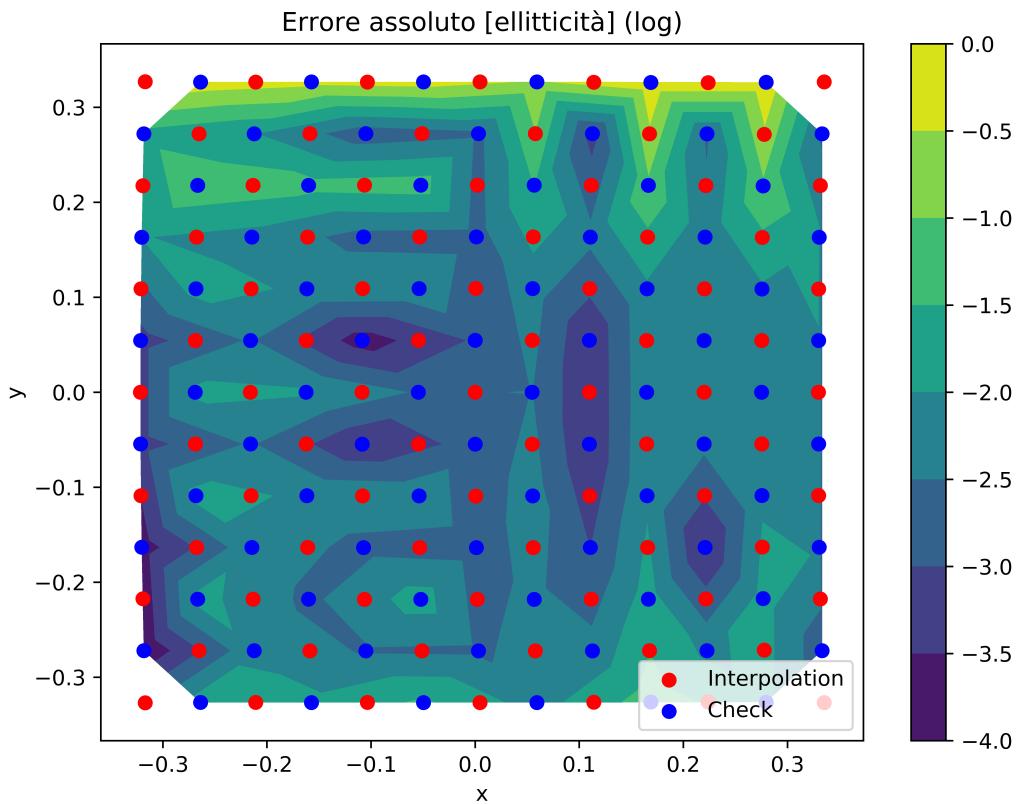


Figura 3.1: Contour plot in scala logaritmica dell’errore assoluto tra ellitticità interpolata tramite il metodo `interp2d` e ellitticità corretta. In rosso sono evidenziati i punti attraverso i quali è stata effettuata l’interpolazione mentre in blu sono rappresentati i punti nei quali è stato valutato l’errore.

sez. 3.1.3, tali anomalie di bordo hanno determinato la scelta del metodo `curve fit` come termine di paragone con le reti neurali.

3.1.2 Curve Fit

Il metodo `curve_fit` permette di fissare i dati con una funzione non lineare tramite il metodo dei minimi quadrati. Il metodo restituisce i valori ottimali dei parametri `popt` che minimizzano la discrepanza $f(xdata, *popt) - ydata$ ². Anche in questo caso ho utilizzato i due subset, `data_int` e `data_check`, per effettuare rispettivamente la regressione e la valutazione dell’errore. Nel mio caso il fit è stato eseguito su un paraboloida in quanto questo richiama la forma della superficie focale.

3.1.3 Risultati dell’interpolazione

In fig. 3.3 sono confrontate le curve che rappresentano l’errore tra dato interpolato e dato corretto per i due metodi utilizzati. Si nota immediatamente la presenza

²In particolare `xdata` è una coppia (x, y) e `ydata` è il valore del parametro di interesse.

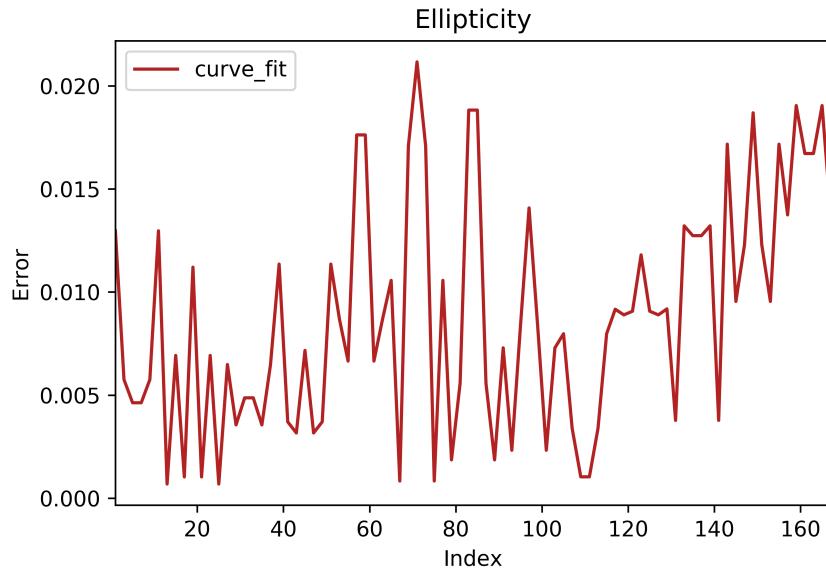


Figura 3.2: Plot dell'errore assoluto tra ellitticità interpolata tramite il metodo `curve_fit` e ellitticità corretta.

di alcuni punti problematici riguardanti `interp2d` che non rendono possibile un confronto tra le due curve.

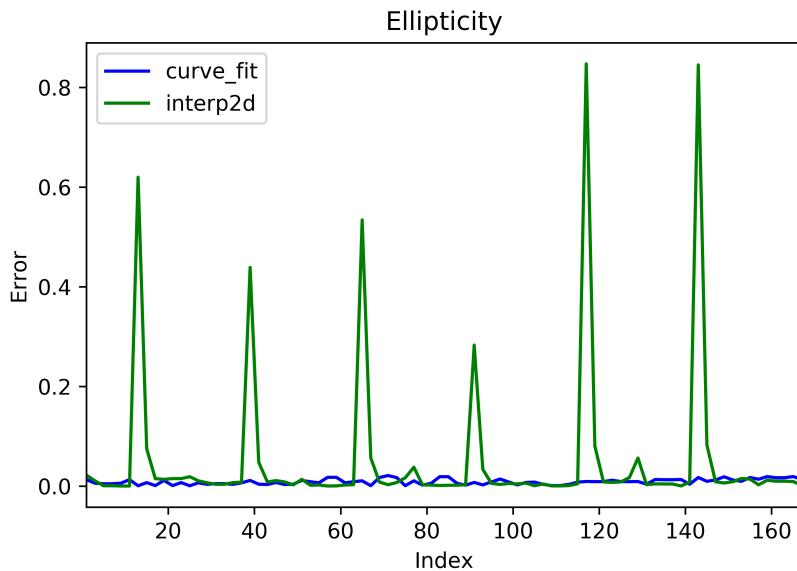
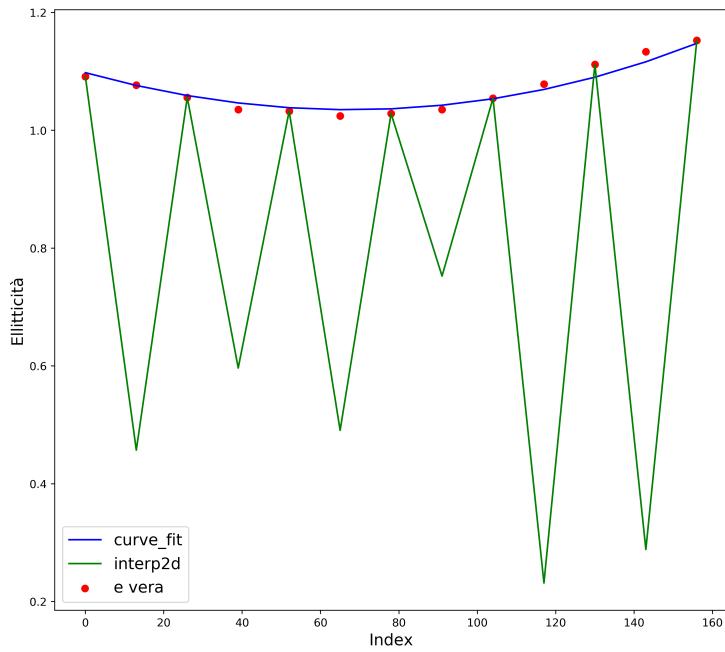


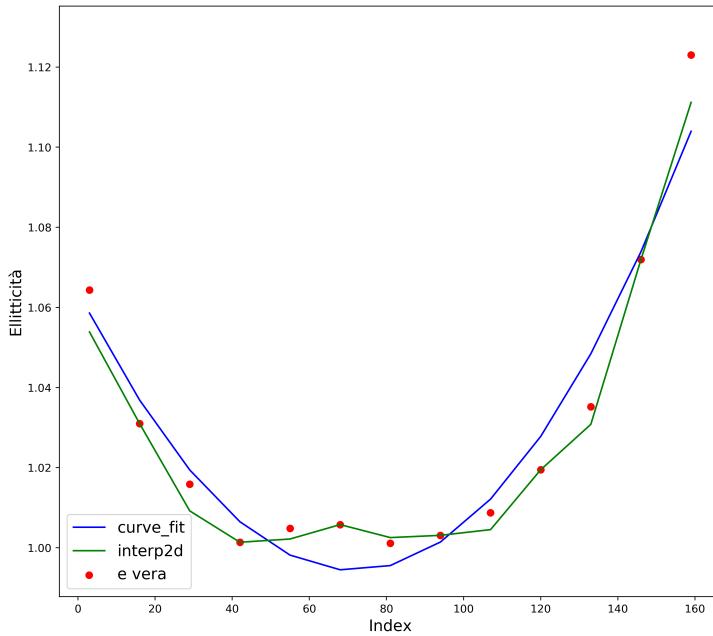
Figura 3.3: Confronto dell'errore relativo ai due metodi di interpolazione.

Analizzando più in dettaglio il comportamento del metodo `interp2d` è possibile mostrare che i punti affetti da errore maggiore sono punti di bordo. L'analisi è stata fatta piazzando il valore interpolato e valore esatto del parametro per ogni riga di punti della griglia 13x13. In fig. 3.4 è mostrato il diverso comportamento di `interp2d` per una riga di punti di bordo e una riga di punti interni alla griglia.

Per poter andare a confrontare gli errori dei due metodi di interpolazione ho creato un subset del dataset iniziale rimuovendo i punti più esterni: `data_mask`. Il



(a) Punti di bordo



(b) Punti interni

Figura 3.4: Plot del valore dell'ellitticità per due diverse righe di punti. Nel grafico 3.4a è mostrato l'andamento dell'ellitticità per una riga di punti di bordo mentre nel grafico 3.4b viene analizzata l'ellitticità di una riga interna della griglia di punti.

passaggio successivo è stato plottare nuovamente l'errore come in fig. 3.3 e il risultato ottenuto è mostrato in fig. 3.5.

Il plot in fig. 3.5 mostra ancora dei punti nei quali l'errore di `interp2d` è molto maggiore rispetto a quello di `curve_fit`, per tale motivo ho deciso di considerare esclusivamente il metodo `curve_fit` come termine di paragone tra i risultati rela-

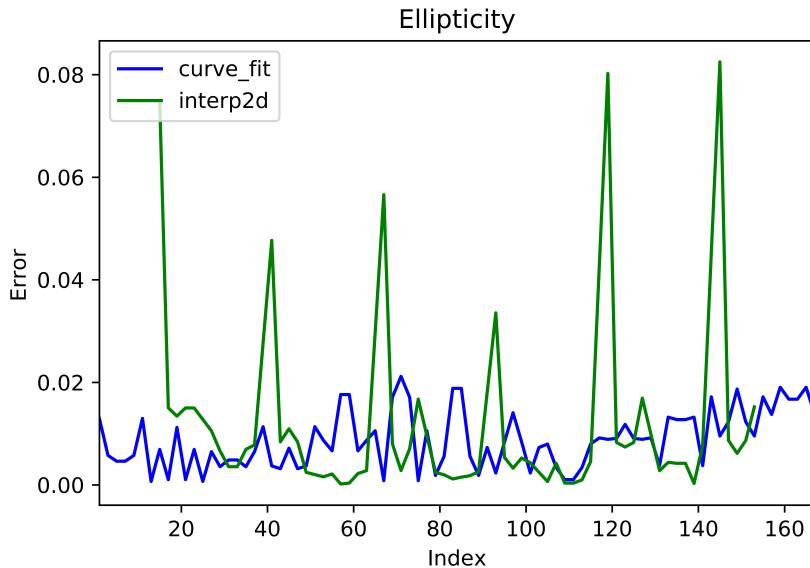


Figura 3.5: Confronto dell’errore relativo ai due metodi di interpolazione dopo aver rimosso i punti di bordo.

tivi all’interpolazione e quelli relativi alle reti neurali, come verrà mostrato nella sezione 3.2.4.

3.2 Reti neurali

In questa sezione mostro come sono state create le architetture delle reti neurali utilizzati per effettuare la regressione, come sono stati gestiti i dati a disposizione per definire **training set** e **validazion set** e com’è avvenuto il processo di training. Infine verrà presentato il confronto dei risultati ottenuti tramite l’utilizzo dell’interpolazione e della rete neurale (3.2.4).

3.2.1 Architettura della rete

Quando si costruisce una rete neurale è necessario definire alcuni elementi che andranno a caratterizzare una particolare architettura di rete. Tali elementi sono:

- La dimensione dell’*input layer*.
- La dimensione dell’*output layer*.
- Il numero di *hidden layers*.
- Il numero di *neuroni* per ogni hidden layer.
- La *funzione di attivazione*.
- L’*optimizer*.
- La *loss function*.

La scelta dei loro valori non è mai univoca, non esiste una regola fissa per avere la ricetta perfetta che porta ai risultati migliori. Nel mio lavoro di tesi ho utilizzato 6 diverse architetture; alcuni degli elementi appena citati sono stati mantenuti invariati per ogni architettura, mentre altri sono stati variati.

Per poter descrivere al meglio la scelta delle architetture è necessario fare alcune osservazioni. Uno dei problemi più diffusi nell'ambito delle reti neurali è quello dell'**overfitting**. Si va incontro ad overfitting quando, per esempio, si utilizza un numero estremamente elevato di neuroni se confrontato con gli elementi che si hanno a disposizione per il training. Ogni neurone è infatti caratterizzato da due parametri liberi (*weight* e *bias*), la presenza di un elevato numero di parametri liberi fa sì che la rete impari perfettamente la corrispondenza tra input e output degli elementi nel training set, ma in tal modo si perde il carattere necessario per ottenere risultati validi.

3.2.2 Pre Training

Bla bla bla

3.2.3 Training

Bla bla bla

3.2.4 Confronto di risultati

Bla bla bla

Capitolo 4

Conclusioni

Bibliografia

- [1] E. Stevens and L. Antiga. *Deep Learning with PyTorch*. Manning, 2019.
- [2] M. Tomasi, C. Franceschet, and S. Realini. The quest for cmb b-modes.