

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	8
1.3 Simulazione di sistemi ottici	9
2 Regressione con reti neurali	13
2.1 Machine learning e tipi di rete	13
2.2 Struttura di una rete neurale	14
3 Previsione delle proprietà di un diagramma di radiazione	19
3.1 Interpolazione	19
3.1.1 Interp2d	19
3.1.2 Curve Fit	20
3.1.3 Risultati dell'interpolazione	21
3.2 Reti neurali	22
3.2.1 Architettura della rete	22
3.2.2 Pre Training	25
3.2.3 Training	27
3.3 Confronto dei risultati	28
4 Conclusioni	33

Capitolo 1

Sistemi ottici nelle microonde

Quasi tutta l'informazione che abbiamo a disposizione su oggetti astronomici molto distanti è contenuta nella radiazione elettromagnetica emessa. Per secoli l'unico tipo di analisi possibile è stata quella nello spettro del visibile. Oggigiorno esistono svariate branche dell'astrofisica che analizzano segnali provenienti dall'Universo a diverse frequenze elettromagnetiche; una di queste branche riguarda lo studio del cosmo attraverso le microonde.

1.1 Utilizzo delle microonde in astrofisica

Il range delle lunghezze d'onda delle microonde è $\lambda \sim 1\text{mm} \div 10\text{cm}$ ¹.

È estremamente importante fare osservazioni a grandi lunghezze d'onda, nel millimetrico e oltre, poichè esistono numerosissime sorgenti cosmologiche che emettono in questo range e possono essere analizzate tramite sistemi ottici nelle microonde. I segnali più comunemente studiati in radio astronomia e attraverso le microonde sono:

- radiazione emessa da gas ionizzato;
- radiazione di sincrotrone, dovuta al moto di particelle cariche libere di muoversi nell'Universo e deviate da campi magnetici;
- effetto Sunyaev-Zeldovich, che permette di rivelare ammassi di galassie altrimenti non visibili;
- emissioni del Sole;
- radiazioni da regioni H II; si tratta di regioni nello spazio in cui sono presenti stelle molto calde (di tipo O o di tipo B) che ionizzano il gas intorno ad esse il quale emette nelle microonde e nel radio;

¹Il confine tra onde radio e microonde non è netto, spesso si parla di radio estendendo il range di lunghezze d'onda anche a quello delle microonde.

- supernovae e resti di supernovae²;
- pulsar;
- radio galassie;
- CMB.

In questo lavoro di tesi lo studio è stato effettuato su sistemi ottici ottimizzati per l'analisi della CMB.

La **CMB**, Cosmic Microwave Background, è la radiazione a microonde di fondo cosmico che permea l'intero Universo. Secondo il *Modello Cosmologico Standard (SCM)* l'Universo ha avuto origine circa 14 miliardi di anni fa da una singolarità iniziale: il *Big Bang*. Nei primissimi istanti dopo il *Big Bang* vi fu una rapidissima fase di espansione (10^{-33} s), detta *inflazione*, seguita da un'espansione più lenta e regolare, che continua tutt'ora. Inizialmente materia e radiazione erano in equilibrio in un plasma estremamente caldo; la progressiva espansione ha causato un abbassamento della temperatura del plasma. L'equilibrio tra materia e radiazione venne a mancare quando la temperatura raggiunse $T \simeq 3000$ K. Questo causò il *disaccoppiamento* tra materia e radiazione: la maggior parte degli elettroni venne catturata dai nuclei atomici permettendo la formazione dei primi atomi neutri. Convenzionalmente si considera come tempo al quale si è verificato il *disaccoppiamento* tra materia e radiazione l'istante t_{dec} in cui il libero cammino medio dei fotoni divenne maggiore della scala dell'Universo osservabile. Questo accadde a circa 380 000 anni dal *Big Bang* e da allora i fotoni primordiali sono liberi di vagare nello spazio e sono i costituenti della radiazione a microonde di fondo cosmico. Nell'ipotesi semplificata di un Universo non in espansione, i fotoni primordiali che possiamo osservare oggi dalla Terra sono quelli che all'istante t_{dec} si trovavano ad una distanza $c(t_{now} - t_{dec})$; il che significa che per ogni istante un osservatore sulla Terra può osservare solo i fotoni emessi da un guscio sferico che prende il nome di *Last Scattering Surface* (LSS). Generalizzando il caso ad un Universo in espansione il concetto rimane lo stesso ma diventa matematicamente più complesso. I fotoni della CMB che vengono oggi misurati rappresentano una radiazione quasi-isotropa il cui spettro è quello di corpo nero a una temperatura $T \approx 2.73$ K. Per studiare le proprietà della CMB sulla sfera definita dalla LSS è utile utilizzare una decomposizione in *armoniche sferiche* $Y_l^m(\theta, \phi)$. La decomposizione in armoniche sferiche di una funzione $f(\theta, \phi)$ su una sfera è data da:

$$f(\theta, \phi) = \sum_{\ell=0}^{\infty} \sum_{m=-\ell}^{\ell} a_{\ell m} Y_{\ell}^m(\theta, \phi)$$

dove $a_{\ell m}$ sono coefficienti complessi che contengono informazioni sia sull'ampiezza che sulla fase delle armoniche sulla sfera. Tuttavia, data l'arbitrarietà del sistema di

²La Crab Nebula, per esempio, è estremamente visibile nelle microonde.

riferimento che si utilizza quando si compie un'analisi statistica della CMB, la fase può essere trascurata; diventa quindi più interessante utilizzare la quantità:

$$C_\ell = \langle a_{\ell m} a_{\ell m}^* \rangle$$

detta **spettro di potenza**. Nel caso di misura da parte di un solo strumento la media $\langle \cdot \rangle$ è effettuata su tutti i possibili valori di m .

La radiazione della CMB è polarizzata e lo spettro di potenza della polarizzazione può essere decomposto in due componenti, dette *modi E* e *modi B*. Una misura dei modi B permetterebbe una verifica del paradigma dell'inflazione in quanto questi sarebbero generati da onde gravitazionali primordiali, e solo l'inflazione sarebbe in grado di espanderne le scale angolari fino a risoluzioni osservabili oggi.³

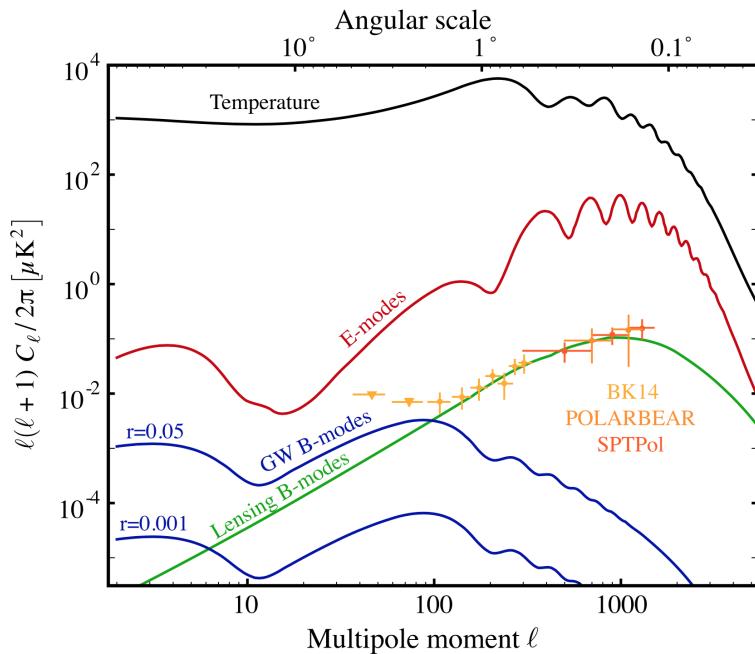


Figura 1.1: Predizioni teoriche dello spettro di potenza della temperatura (in nero), dei modi E (in rosso) e dei modi B (in blu). Lo spettro di potenza dei modi B è rappresentato per due valori diversi di r ($r=0.001$ e $r=0.05$) [1].

Se vogliamo effettuare una corretta misura della polarizzazione della CMB è necessario sottrarre al segnale misurato tutte le anisotropie dovute a oggetti presenti tra l'osservatore e la LSS. Si utilizza comunemente il termine *foreground* per riferirsi a tutte le emissioni nell'intervallo di frequenze compreso tra 10 e 1000 GHz presenti tra noi e il guscio sferico definito dalla LSS. Il problema della separazione delle componenti (*component separation*) implica la necessità di effettuare misure a multibanda lungo tutto il range di frequenze comprese tra 10 e 1000 GHz con una sensibilità sufficiente per poter rivelare il debole segnale dei modi B e allo stesso tempo caratterizzare il foreground ed ottenere una mappa pulita della CMB.

³In cosmologia r è un parametro che prende il nome di rapporto tensore-scalare ed è definito come il rapporto tra la potenza dei modi B e dei modi E primordiali.

1.2 Diagramma di radiazione

Un sistema ottico è un dispositivo il cui scopo è quello di focalizzare la luce proveniente dal cielo e farla propagare verso un rivelatore.

Per l'analisi della CMB è nostro interesse studiare la direzione di provenienza della radiazione. Consideriamo quindi, d'ora in poi, un fascio d'antenna modellato per misurare il segnale che proviene da una direzione specifica. La risposta di un sistema ottico ideale può essere allora rappresentata come una delta di Dirac: non nulla solo lungo la linea di vista. Tuttavia i fenomeni di interferenza e diffrazione rendono la situazione molto più complessa; in particolare nella radio astronomia e nell'astronomia a microonde il problema è particolarmente importante poichè le dimensioni degli elementi ottici degli strumenti sono comparabili con le lunghezze d'onda d'interesse.

La risposta angolare di un sistema ottico è quantificata da una funzione $\gamma(\theta, \phi)$ detta *beam function* che definisce il **diagramma di radiazione**. Idealmente $\gamma(\theta, \phi)$ dovrebbe corrispondere a una delta di Dirac⁴ ma quello che viene in realtà osservato è una risposta simile a quella riportata in Fig. 1.2. È possibile osservare la presenza di un *main beam* e di lobi secondari. Tipicamente la maggior parte della radiazione è contenuta nel main beam. A partire dal diagramma di radiazione si definiscono

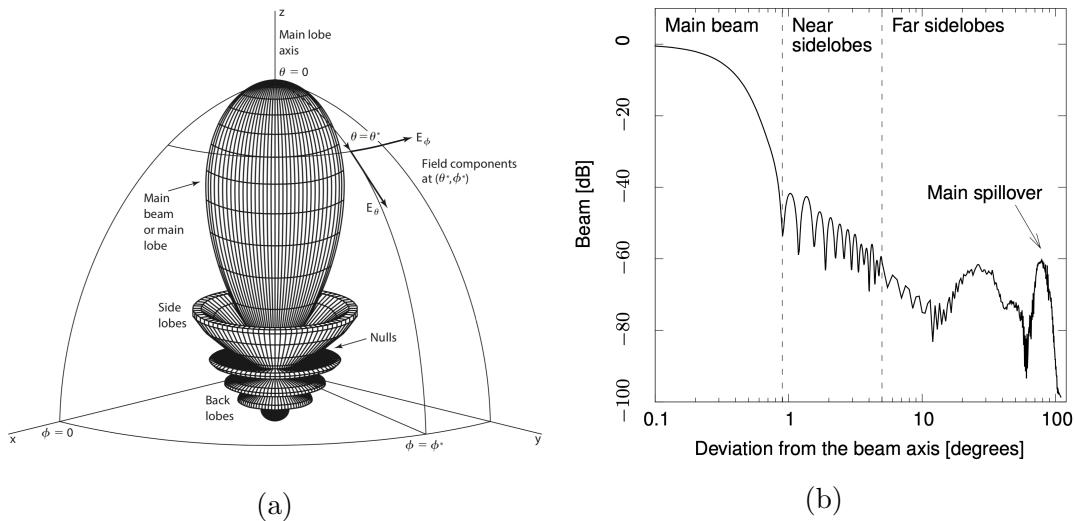


Figura 1.2: Tipico andamento della *beam function* $\gamma(\theta, \phi)$. La Fig. 1.2a rappresenta il diagramma di radiazione tridimensionale di un'antenna direzionale. [6]. La Fig. 1.2b rappresenta una sezione dello stesso grafico su un piano parallelo all'asse del main lobe. [3]

alcuni parametri che permettono una sua descrizione; tali parametri sono i protagonisti di questo lavoro di tesi. La **Full Width Half Maximum** (FWHM) del main beam è la larghezza angolare a metà della sua altezza ed è uno dei parametri

⁴Questa idealizzazione viene spesso chiamata *pencil beam idealization*

più importanti e più diffusi per la caratterizzazione della risoluzione di uno strumento⁵. Nel corso dei prossimi capitoli verranno considerati due diversi valori per la FWHM, rispetto all'asse x e rispetto all'asse y . Attraverso queste due grandezze è possibile definire un ulteriore parametro: l'**ellitticità**. Questa è definita come il rapporto tra le due FWHM ponendo al numeratore la più grande tra FWHM_x e FWHM_y . Hanno inoltre grande importanza i parametri che riguardano la polarizzazione. Supponiamo di studiare un'antenna in trasmissione⁶ e considerare un segnale polarizzato linearmente lungo una determinata direzione. È possibile definire una componente *co-polare* ed una componente *cross-polare* della radiazione. La componente co-polare è la radiazione irradiata lungo la direzione originale di polarizzazione mentre la componente cross-polare è la radiazione irradiata lungo la direzione perpendicolare a quella originale. In particolare i parametri utilizzati nei prossimi capitoli riguardano la componente **co-polare massima** e la componente **cross-polare massima**.

Per fare misurazioni di CMB è necessario richiedere alcune condizioni relative agli strumenti ottici. In riferimento alla Fig. 1.1 si nota che una certa scala angolare mi permette di selezionare i multipoli che è possibile osservare; è quindi necessario avere una FWHM che permetta di risolvere i dettagli della CMB entro certe scale angolari. La relazione approssimata che lega il valore di ℓ alla FWHM è: $\ell \sim 180^\circ / \theta_{\text{FWHM}}$. Inoltre il livello dello spettro di potenza dei modi B è molto più basso di quello dei foreground e dei modi E; poichè si ha grande interesse nel misurare i modi B è necessaria una grande sensibilità strumentale. Per raggiungere elevate sensibilità è fondamentale avere a disposizione vasti piani focali che mi permettano di utilizzare un elevato numero di rivelatori. Infine, per avere una misura pulita della CMB, è essenziale rimuovere tutti i foreground e avere quindi a disposizione misure a tante frequenze diverse, il che si traduce ancora una volta nella richiesta di un elevato numero di rivelatori.

1.3 Simulazione di sistemi ottici

Simulare un sistema ottico significa studiare quanta potenza viene ricevuta in funzione dell'angolo rispetto alla linea di vista. Esistono diversi software in grado di simulare un fascio d'antenna e quindi la sua risposta angolare.

Nel corso di questo lavoro di tesi ho utilizzato dei dati relativi ad un particolare strumento per l'analisi della CMB: *STRIP*. Lo strumento STRIP (STRatospheric Italian Polarimeter) fa parte dell'esperimento internazionale *LSPE* (Large-Scale Polarization Explorer) ideato per effettuare misure di CMB ad elevate scale angolari.

⁵Tale parametro viene anche indicato come θ_{FWHM}

⁶Strumenti ottici per lo studio della CMB lavorano in ricezione ma è possibile studiare alternativamente un'antenna in trasmissione per il *principio di reciprocità*.

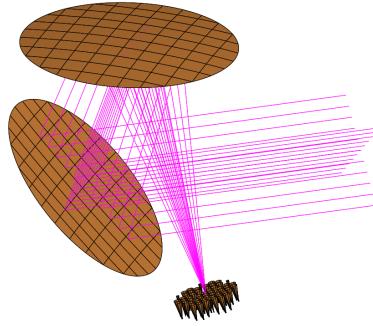


Figura 1.3: Modello in GRASP del telescopio di STRIP.

In particolare STRIP è stato progettato per la misura di radiazione a basse frequenze; esso presenta 49 antenne a 43 GHz e 6 antenne a 90 GHz. In Fig. 1.3 è rappresentato il modello di STRIP attraverso il software *GRASP* (General Reflector Antenna Software Package). Attraverso GRASP è stato possibile ottenere la tabella in Fig. 1.4 che riporta le caratteristiche del beam data una posizione (x, y, z) sulla superficie focale.

GRASP è uno strumento molto potente che permette di simulare sistemi ottici complessi. Tuttavia i tempi di calcolo di GRASP sono elevati⁷; per ogni simulazione, e quindi per ogni antenna, vengono costruiti dei grafici come quelli riportati in Fig. 1.5 e poi da quelli vengono ricavati i valori riportati nel dataset 1.4. Un grafico come quello in Fig. 1.5 produce quindi una singola riga della tabella 1.4.

	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 1.4: Dataset relativo allo strumento STRIP ottenuto tramite la simulazione in GRASP. Tale dataset è stato utilizzato per l'analisi descritta nei capitoli successivi.

Nel caso di STRIP è ancora possibile simulare l'intera ottica tramite GRASP poichè il numero di antenne è piuttosto limitato. Tuttavia per sistemi ottici più

⁷Per produrre i dati in Fig. 1.4 sono stati impiegati due giorni.

complessi in cui il numero di antenne diventa di circa 2 ordini di grandezza superiore, risulta del tutto impossibile effettuare una simulazione completa dell'intera ottica. È quindi forte la necessità di trovare una via alternativa che permetta di stimare i parametri che descrivono il beam in una qualsiasi posizione.

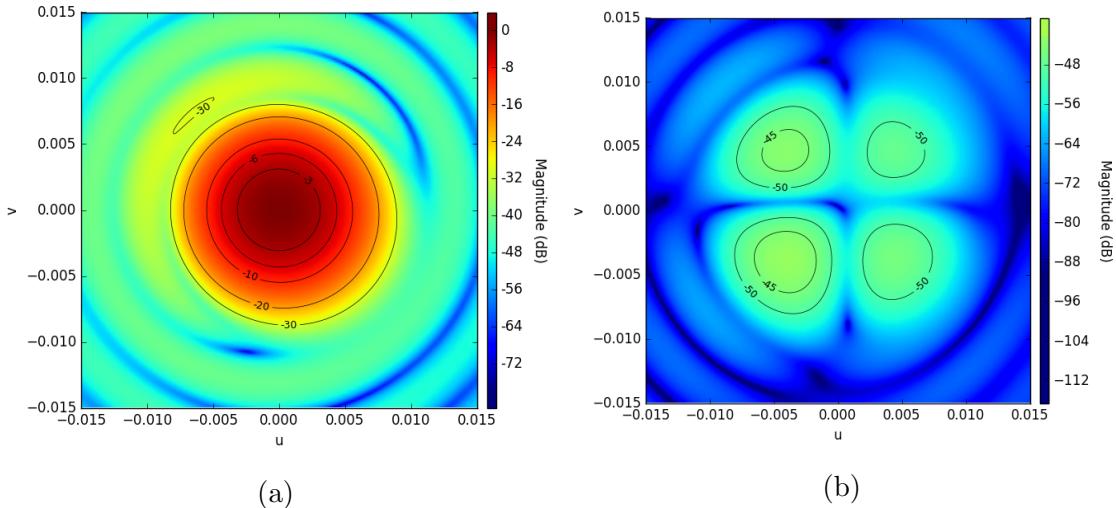


Figura 1.5: Diagramma della componente co-polare 1.5a e della componente cross-polare 1.5b di un'antenna off-axis. È possibile notare come il profilo del beam sia leggermente ellittico; per il sistema ottico di STRIP infatti l'ellitticità del fascio aumenta spostandosi dal centro del piano focale.

Capitolo 2

Regressione con reti neurali

2.1 Machine learning e tipi di rete

Il *machine learning (ML)* è un campo di ricerca a metà tra statistica, intelligenza artificiale ed informatica ed è una banca della disciplina chiamata *intelligenza artificiale (AI)*. La sua applicazione riguarda ormai vastissimi campi della scienza e non solo. Sono usate tecniche di machine learning per esempio per il riconoscimento facciale o per i suggerimenti di prodotti da comprare sulle piattaforme di shopping online, ma anche in ambito medico per la diagnosi di particolari malattie.

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. Una branca del ML vede protagoniste le *reti neurali artificiali (ANN)*¹. Queste tecniche si basano sull'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**. Una rete neurale approssima una funzione $y = f(x)$ tramite una serie di neuroni connessi tra loro. Ogni neurone è caratterizzato da 2 parametri, come spiegato in Sez. 2.2, e l'approssimazione della relazione tra x e y è effettuata tramite l'ottimizzazione di tali parametri a partire dalle coppie (x, y) che descrivono il comportamento della funzione f in casi realistici. L'ottimizzazione dei parametri è da intendersi in termini di minimizzazione di una particolare funzione che prende il nome di **loss function**, che quantifica la discrepanza tra il comportamento di f e quello della rete neurale. Riassumendo, quello che accade all'interno di una rete neurale quando si vuole approssimare una corrispondenza tra input e output è una modifica dei parametri di rete in modo da ottenere un valore di *loss* il più piccolo possibile.

¹Spesso vengono chiamate semplicemente reti neurali (NN).

Il *deep learning* è un’ulteriore branca delle NN che entra in gioco quando si utilizzano reti neurali con un elevato numero di layers e di neuroni: le *deep neural networks*. In questo lavoro di tesi ho creato reti con un ridotto numero di layers, queste non rientrano nel campo del deep learning.

Esistono due classi di problemi affrontati con la tecnica delle reti neurali: la *classificazione* e la *regressione*. Si ha una rete classificazionale se nell’approssimare $y = f(x)$, y può assumere un valore preso dagli elementi di un insieme discreto, mentre si ha una rete regressionale quando y può assumere un continuo di valori. 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. Un esempio di classificazione supervisionata è il riconoscimento di un numero a partire da un’immagine manoscritta. Un esempio invece di classificazione non supervisionata si ha quando, a partire da immagini di cifre manoscritte, si vuole determinare quali tra esse rappresentino lo stesso numero.

Le reti neurali per la regressione entrano in gioco quando, date delle coppie input-output, si vuole determinare la funzione che approssimi al meglio la relazione. Per effettuare la stima delle proprietà di un diagramma di radiazione (Sez. 3.2) ho utilizzato quest’ultimo tipo di reti neurali. In particolare la ricerca alla base di questo lavoro di tesi è stata quella di una rete neurale che permetesse, a partire da una posizione sulla superficie focale, di stimare il valore di un determinato parametro del diagramma di radiazione. Come sarà possibile notare nei paragrafi successivi, la posizione sulla superficie focale è data da una coppia (x, y) , senza considerare la coordinata z ; ho deciso di procedere così poichè i punti si trovano su una superficie ben definita, il che significa che dati x e y la z è univocamente determinata.

Il programma utilizzato in questa tesi per sviluppare i codici che permettono di effettuare questo task è Python. Questo fornisce librerie apposite per il caricamento, l’analisi e la visualizzazione dei dati; le librerie utilizzate nei codici implementati sono: *pandas*, *numpy*, *matplotlib*, *seaborn*, *pytorch*.

2.2 Struttura di una rete neurale

Un rete neurale è basata sull’interconnessioni di unità fondamentali: i **neuroni**. Essa può essere rappresentata 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. Un neurone è rappresentato da una funzione non lineare, detta **funzione di attivazione**, applicata ad una trasformazione lineare. Ogni neurone è caratterizzato da due parametri liberi: *w* (*weight*) e *b* (*bias*). L’espressione matematica che descrive

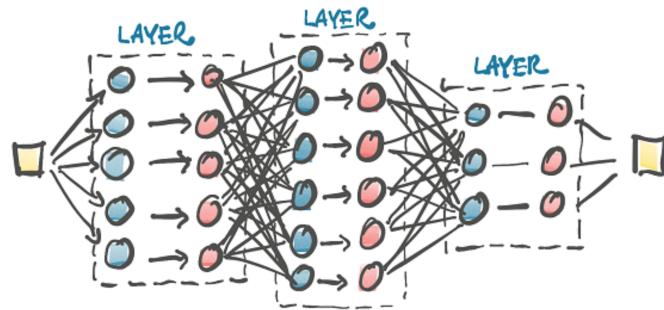


Figura 2.1: Schema di una rete neurale con tre hidden layers. Il quadrato giallo sulla sinistra rappresenta l'input layer mentre quello sulla destra rappresenta l'output layer. Immagine presa da *Deep Learning with PyTorch*, E. Stevens, L. Antiga [5].

un singolo neurone è quindi:

$$o = f(wx + b)$$

dove x è il dato di input, o rappresenta l'output e f una determinata funzione di attivazione non lineare.

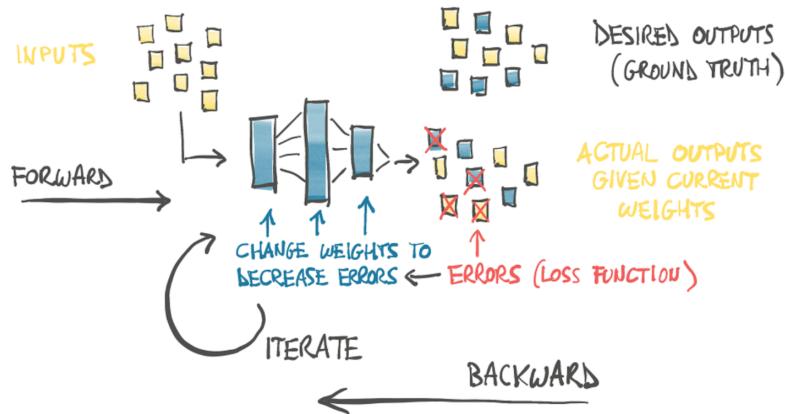


Figura 2.2: Schema del processo di apprendimento di una rete neurale. Immagine presa da *Deep Learning with PyTorch*, E. Stevens, L. Antiga [5].

Il cuore del processo di apprendimento, mostrato schematicamente in Fig. 2.2, sta nella stima dei giusti parametri (weights e biases) della rete. Questo avviene seguendo determinati step:

- vengono forniti alla rete inputs e outputs desiderati;
- il modello calcola gli outputs a partire dagli inputs forniti; questo step prende il nome di *forward pass*;
- viene misurato l'errore comparando l'output calcolato e quello fornito in partenza. La funzione che quantifica tale discrepanza è detta *loss function*;

- vengono modificati i parametri di rete in modo da minimizzare la loss function; questo step prende il nome di *backward pass*²;
- viene ripetuta l'intera procedura fino alla convergenza della rete, ovvero fino a che la loss function non raggiunge un andamento stabile.

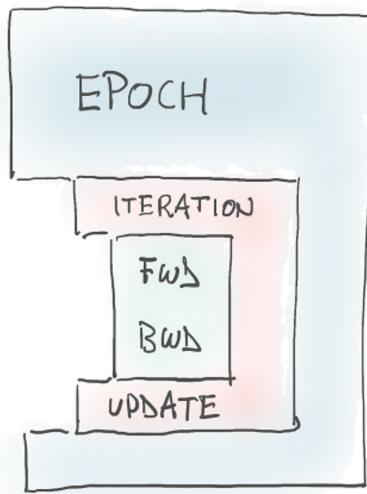


Figura 2.3: Rappresentazione schematica del ciclo di training. Immagine presa da *Deep Learning with PyTorch*, E. Stevens, L. Antiga [5].

Com'è possibile notare dalla Fig. 2.3, un ciclo completo su tutti i dati messi a disposizione per l'apprendimento viene chiamato *epoca*; la fase di apprendimento avviene attraverso numerose epoche.

Le reti che ho costruito, descritte in dettaglio nella Sez. 3.2.1, sono chiamate *reti neurali feed-forward*; il termine sta ad indicare che le connessioni tra i neuroni non formano dei cicli³. In particolare si tratta di reti *fully connected*, ovvero dove ciascun neurone è connesso a tutti i neuroni del layer precedente. Per la costruzione, il training e la validazione della rete ho utilizzato la libreria **PyTorch** di Python. PyTorch lavora con elementi chiamati *Tensor*; questi sono array multi dimensionali molto simili agli array di *numpy*. PyTorch mette a disposizione strumenti che permettono di automatizzare, per esempio, il processo di apprendimento. Lo step più complesso in fase di training è quello di *backward* durante il quale viene valutato il gradiente dell'errore rispetto ai parametri di rete. Per dare un'idea della potenza di PyTorch mostro qui una parte del codice implementato per la fase di training:

```
train_loss.backward()
opt.step()
```

²Tale valutazione viene fatta da un *optimizer* attraverso la valutazione del gradiente dell'errore rispetto ai parametri di rete.

³Le reti che presentano cicli al loro interno vengono chiamate *ricorsive*.

dove `train_loss` è la funzione di perdita che calcola l'errore tra output calcolato e output desiderato, mentre `opt` è un oggetto ben definito chiamato *optimizer*. Attraverso queste due righe di codice viene valutato il gradiente e vengono aggiornati i parametri di rete.

Nel prossimo capitolo illustrerò le scelte fatte per la definizione dell'architettura di rete tutti gli step che sono stati necessari per ottenere le stime dei parametri del diagramma di radiazione.

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. L'analisi è stata effettuata su ogni parametro di interesse; tuttavia i risultati ottenuti sono spesso molto simili al variare del parametro e quelli riportati nei prossimi capitoli sono grafici che hanno carattere generale, nonostante si riferiscono quasi esclusivamente all'ellitticità. Ricordo che 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 paraboloida. Per poter effettuare l'interpolazione dei parametri e in seguito verificare la sua bontà, il dataset in fig. 1.4 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) appartenti al subset `data_int`. Una volta specificato il parametro di interesse, l'algoritmo effettua un'interpolazione

¹La griglia non deve essere necessariamente regolare.

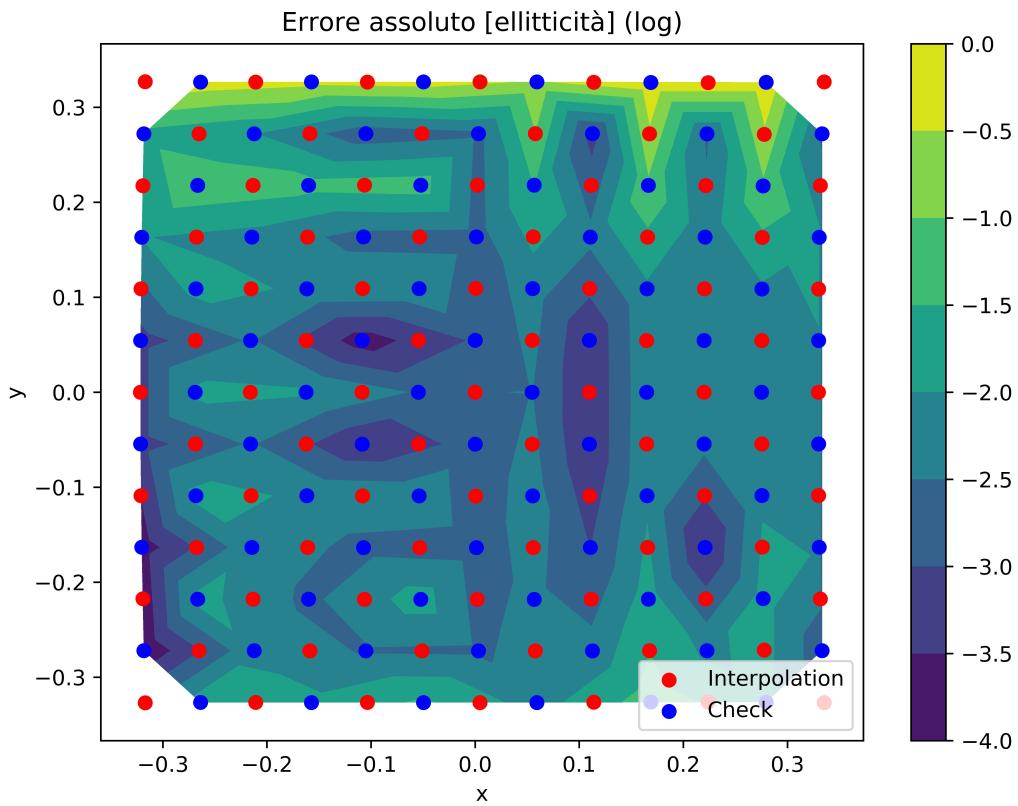


Figura 3.1: Contour plot in scala logaritmica dell’errore assoluto tra ellitticità interpolata tramite il metodo `interp2d` e ellitticità corretta. In rosso sono evidenziati i punti attraverso i quali è stata effettuata l’interpolazione mentre in blu sono rappresentati i punti nei quali è stato valutato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

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 (x, y) 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 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 il

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

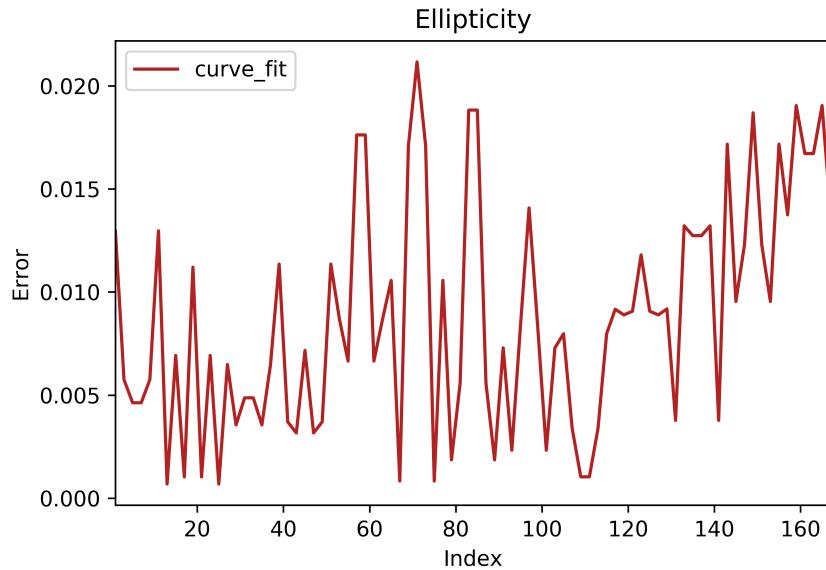


Figura 3.2: Plot che mostra l'errore assoluto tra ellitticità interpolata tramite il metodo `curve_fit` e ellitticità corretta.

fit 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

Ho poi comparato i due metodi per stabilire quale poter utilizzare come confronto con i risultati delle reti neurali. In fig. 3.3 sono confrontate le curve che rappresen-

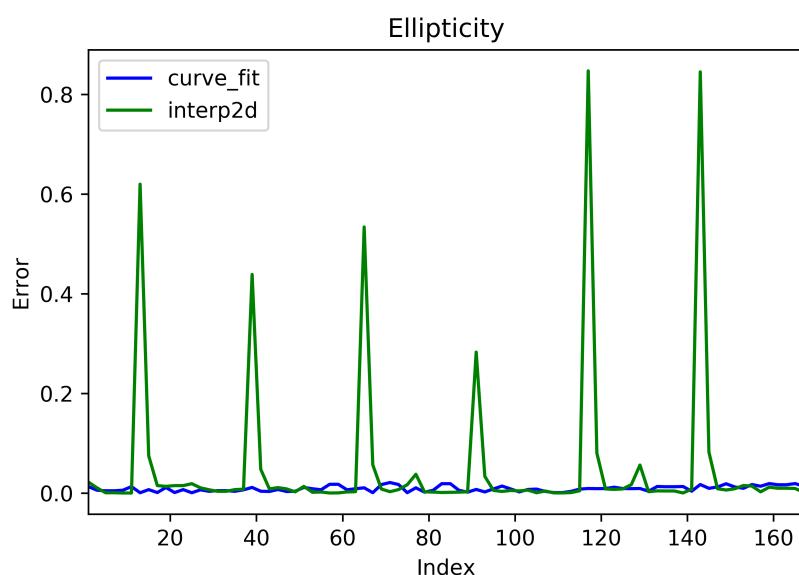


Figura 3.3: Confronto dell'errore prodotto dai due metodi di interpolazione relativo all'ellitticità.

tano l'errore tra dato interpolato e dato corretto per i due metodi utilizzati. Si nota immediatamente la presenza di alcuni punti problematici riguardanti `interp2d` che non rendono possibile un confronto tra le due curve.

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 passaggio successivo è stato piazzare 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 relativi all'interpolazione e quelli relativi alle reti neurali, come verrà mostrato nella sezione 3.3.

3.2 Reti neurali

In questa sezione mostro come sono state create le architetture delle reti neurali utilizzate per effettuare la regressione, com'è stata verificata la loro validità, 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 delle reti (3.3).

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*.

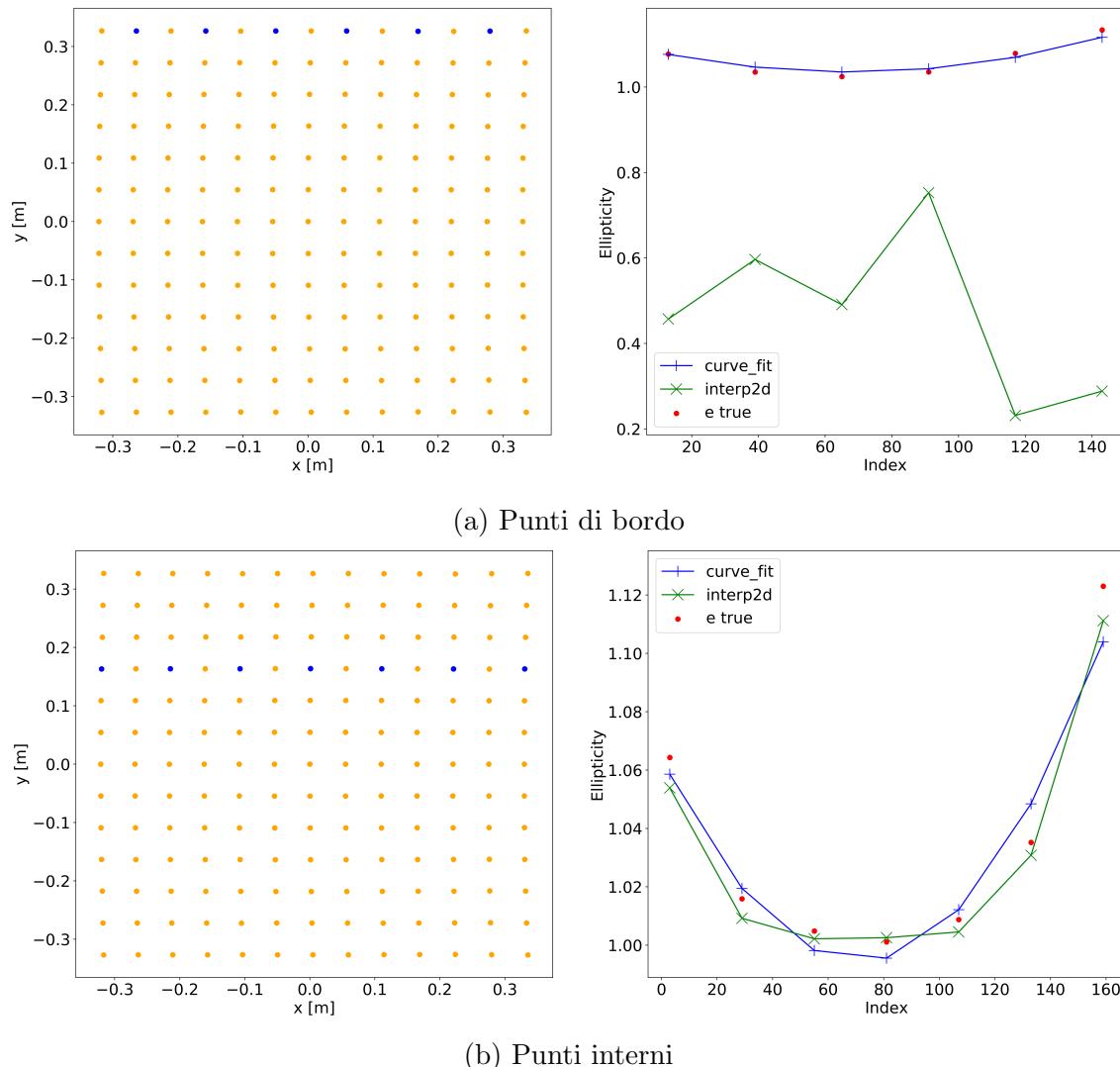


Figura 3.4: Plot del valore dell’ellitticità per due diverse righe di punti. Sono indicati il valore vero, in rosso, il dato previsto con `interp2d`, in verde e con `curve_fit`, in blu. Nel grafico 3.4a è mostrato l’andamento dell’ellitticità per una riga di punti di bordo mentre nel grafico 3.4b viene analizzata l’ellitticità di una riga interna della griglia di punti. Nel grafico sulla sinistra sono evidenziati in blu i punti, sulla riga analizzata, su cui è effettuata la verifica dell’interpolazione. Al variare del parametro variano i punti problematici per `interp2d` ma l’andamento generale rimane lo stesso, il presente plot è dunque da considerarsi rappresentativo di tutti i casi analizzati.

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 architettura³, mentre altri sono stati modificati.

Per poter descrivere al meglio la scelta delle architetture è necessario fare alcu-

³L’input, per esempio, è sempre una coppia (x, y) che definisce una posizione mentre l’output è un singolo valore di un particolare parametro in quel punto.

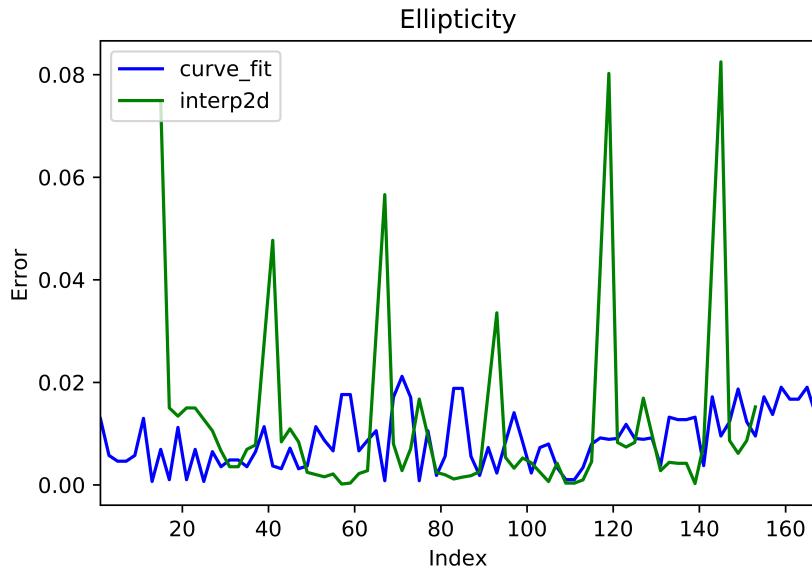


Figura 3.5: Confronto dell’errore relativo ai due metodi di interpolazione dopo aver rimosso i punti di bordo.

ne 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 predittivo necessario per ottenere risultati validi. La dimensione del *trainig set* andrà dunque a fissare un limite superiore del numero di neuroni totali della rete. Un’ulteriore osservazione di fondamentale importanza riguarda la normalizzazione dei dati. I dati relativi al parametro d’interesse, ovvero i dati di output, sono stati infatti normalizzati per agevolare la convergenza della rete. Tale normalizzazione è descritta in dettaglio nella sezione 3.2.2. Non è stato invece necessario normalizzare l’input poichè appartiene, già in partenza, a un range ottimale⁴. Con tali premesse è ora possibile giustificare le scelte fatte nella definizione dell’architettura delle reti. I parametri rimasti sempre invariati sono:

- Dimensione dell’**input layer**: 2
- Dimensione dell’**output layer**: 1
- **Optimizer**: `optim.Adam`⁵

⁴Sia i valori di x che di y oscillano tra -0.34 e +0.34

⁵È uno dei più comuni *optimizer* forniti da *PyTorch*.

Funzione d'attivazione	# hidden layers	# neuroni/hidden layer
Tanh	1	7
Tanh	2	4
Tanh	3	3
Sigmoid	1	7
Sigmoid	2	4
Sigmoid	3	3

Tabella 3.1: Rappresentazione schematica delle architetture costruite.

- **Loss function:** `MSELoss`⁶

Le 6 architetture costruite sono state ottenute tramite una diversa combinazione di funzioni di attivazione e numero di hidden layers. Le funzioni di attivazione che ho utilizzato sono la **Tanh** e **Sigmoid**, rappresentate in fig. 3.6. Per ogni funzione di attivazione ho considerato 3 reti con un numero crescente di hidden layers. Al variare del numero di hidden layers ho modificato il numero di neuroni relativo ad ogni layer. Le architetture finali sono riportate in tabella 3.1.

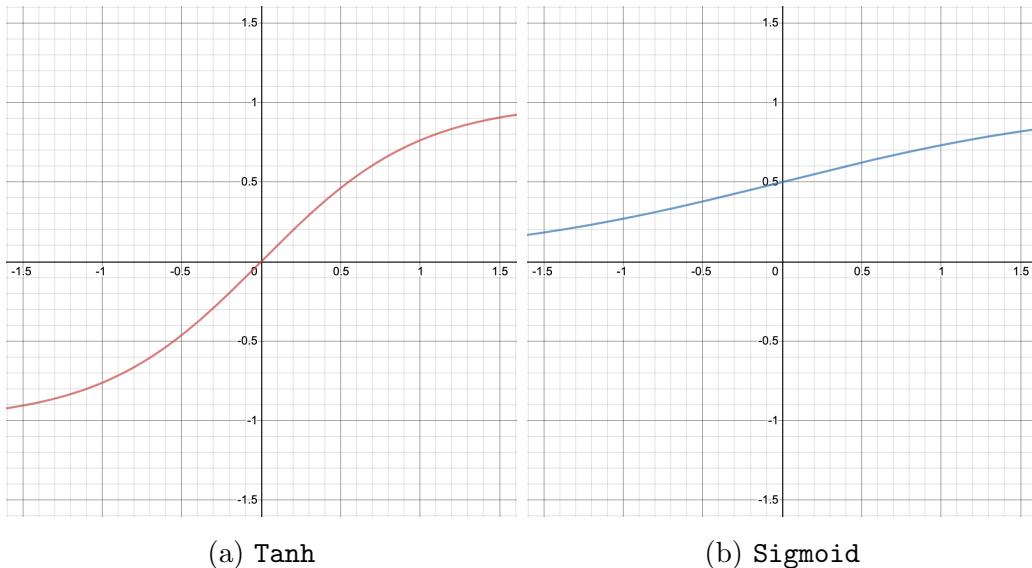


Figura 3.6: Funzioni di attivazione utilizzate. La funzione sigmoidea rappresentata in figura 3.6b è definita come: $P(t) = \frac{1}{1+e^{-t}}$.

3.2.2 Pre Training

In fase di pre-training ho effettuato due operazioni principali: la normalizzazione dei dati e lo split del dataset in un `training_set` ed un `validation_set`. Ho inoltre effettuato un test per la verifica della convergenza della rete.

⁶Errore quadratico medio.

Gli output sono stati normalizzati in modo da appartenere all'intervallo $[0, 1]$, nel caso di *Sigmoid* come funzione di attivazione, oppure a $[-1, 1]$ nel caso di *Tanh*. Le formule utilizzate per la normalizzazione nei due casi sono:

$$y' = \frac{y + y_m}{y_m - y_M} \quad (3.1)$$

$$y' = \frac{2y - y_m - y_M}{y_M - y_m} \quad (3.2)$$

dove y_m corrisponde al minimo valore del parametro mentre y_M corrisponde al massimo. L'equazione 3.1 permette una normalizzazione dell'output in $[0, 1]$ mentre la 3.2 in $[-1, 1]$ ⁷.

Come nel caso dell'interpolazione, a partire dal dataset in Fig. 1.4 sono stati creati due subsets per poter effettuare separatamente il training della rete e la verifica della bontà delle sue previsioni. Circa il 75% (126) degli elementi totali (169) costituisce il **training_set** mentre il restante 25% (43) costituisce il **validation_set**. Questa operazione di caricamento è stata fatta tramite un metodo messo a disposizione da *PyTorch*, **DataLoader**, attraverso il seguente codice:

```
loader = torch.utils.data.DataLoader(dataset,
                                      batch_size = 126,
                                      shuffle = True)

train, validation = loader
```

Listing 3.1: Loader

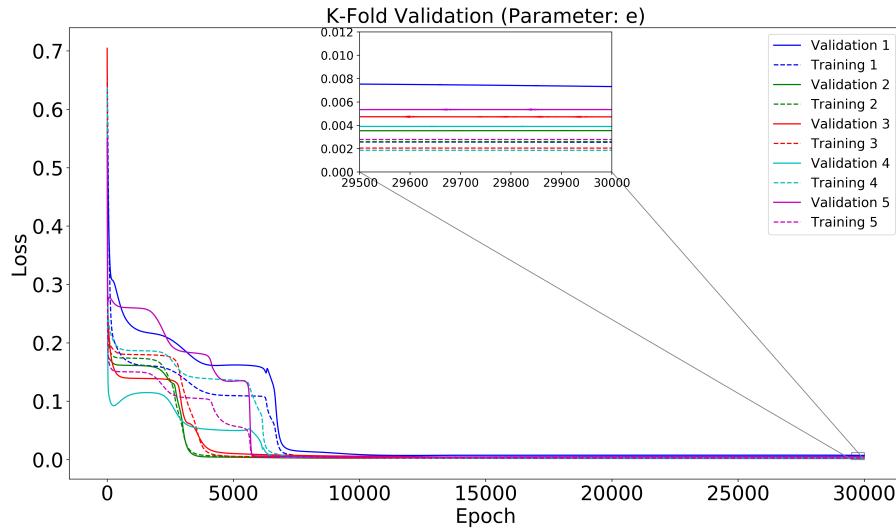
La scelta degli elementi che vanno a finire in un determinato subset è fatta casualmente grazie alla specifica **shuffle = True**.

Una volta definita l'architettura di rete ed effettuate le operazioni di pre-training è necessario verificare la convergenza della rete, per essere certi che non vada in overfitting. Questo è stato fatto per ogni parametro e per ogni architettura. Per effettuare il test della rete ho utilizzato un metodo di validazione chiamato **K-Fold**. Il metodo consiste in:

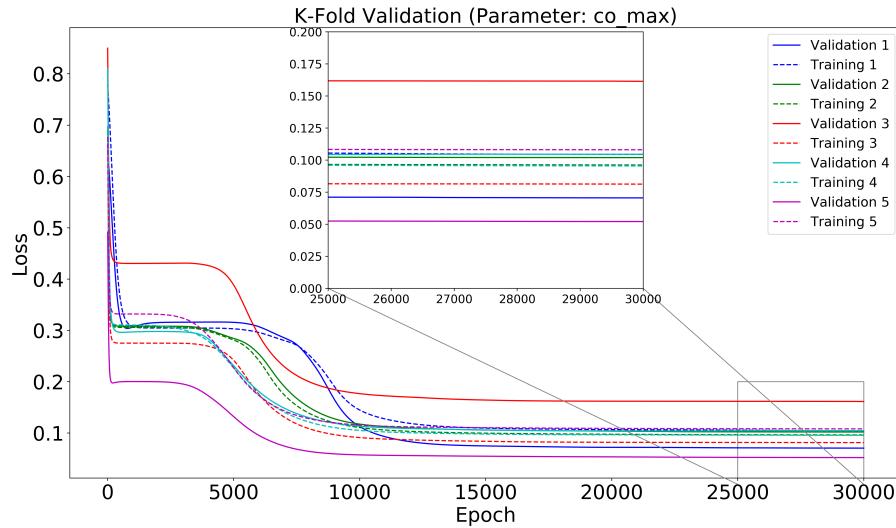
- mescolare il dataset iniziale;
- sezionare il dataset attraverso K splits di eguale numerosità. Nel mio caso $K = 5$;
- utilizzare $K-1$ parti per il **training_set** e la parte rimanente per il **validation_set**;
- valutare la *loss function* relativa al training e al validation;
- ripetere l'operazione per K volte in modo da testare, a turno, tutti gli elementi del dataset.

In Fig. 3.7 sono rappresentati alcuni dei risultati ottenuti.

⁷Dopo aver effettuato il training e aver ottenuto i risultati, i dati sono stati rinormalizzati in modo da tornare al range iniziale.



(a) Test di rete per il parametro e . L’architettura di rete è data da 1 hidden layer e dalla *Tanh* come funzione d’attivazione.



(b) Test di rete per il parametro co_max . L’architettura di rete è data da 1 hidden layer e dalla *Sigmoid* come funzione d’attivazione.

Figura 3.7: Test di rete per due diversi parametri e due diverse architetture. I grafici rappresenta il valore di *loss* per il training e il validation; Nel caso 3.7a le curve tendono a 0 ma, come si nota dallo zoom-in, non valgono esattamente 0. Nel caso 3.7b invece non viene raggiunto un buon valore di *loss*, quella particolare architettura non sarà quindi ottimale per il parametro co_max . Grafici di questo tipo ci mostrano che le reti non vanno in overfitting perchè, se così fosse, le curve relative al validation dovrebbero crescere all’aumentare delle epoche.

3.2.3 Training

Per effettuare il training della rete ho innanzitutto creato una funzione che mi permettesse di inizializzare i pesi della rete. Questo è stato fatto poichè essi vengono inizializzati da Pytorch casualmente ed è quindi necessario avere a disposizione una

funzione apposita che sia in grado di determinare la distribuzione dei loro valori iniziali. Con la giusta inizializzazione è più probabile che la rete converga.

```
def init_weights(m):
    if type(m) == nn.Linear:
        m.weight.data.uniform_(0.0, 1.0)
        m.bias.data.fill_(0.0)
```

Listing 3.2: Inizializzazione di weights e biases

Com’è possibile osservare dal codice 3.2.3, i biases di rete sono inizializzati a 0 mentre i weights sono distribuiti uniformemente tra 0 e 1.

Dopo aver inizializzato i pesi ho caricato i dataset per il training e per il validation attraverso il codice 3.2.2. Il training è stato effettuato su 30000 epoche. Il numero è elevato ma, avendo provato tramite la fase di validazione, spiegata in dettaglio nella Sez. 3.2.2, che la rete non va in overfitting ho deciso di mantenerlo. Inoltre ogni 5000 epoche è stato rieffettuato uno *shuffle* del `training_set` e del `validation_set`, sempre attraverso il codice 3.2.2. Questa tecnica, chiamata di *shuffling*, è utilizzata per minimizzare i problemi di overfitting e permette di rendere più robusto il processo di convergenza.

Nella prossima sezione verranno presentati i risultati ottenuti e confrontati con quelli relativi all’interpolazione.

3.3 Confronto dei risultati

Prima di presentare i risultati ottenuti riassumo le analisi che ho effettuato. I parametri presi in considerazione sono 5: ellitticità, FWHM (rispetto a x), FWHM (rispetto a y), componente co-polare massima e componente cross-polare massima. Le reti costruite sono 6 e gli elementi che ne caratterizzano l’architettura sono riportati in Tab. 3.1. In totale ho quindi effettuato 30 analisi. Per ogni analisi ho prodotto due grafici che confrontano i risultati ottenuti tramite le reti neurali e tramite l’interpolazione. Infine ho creato dei *violin plots* che mi permettessero, in un unico grafico, di comparare i due metodi al variare delle architetture.

Il grafico riportato in Fig. 3.8 è rappresentativo del caso in cui le reti neurali predicono il valore del parametro in maniera più efficace rispetto all’interpolazione, mentre quello riportato in Fig. 3.9 mostra che nessuno dei due approci è apprezzabilmente più efficace dell’altro.

VIOLINS!

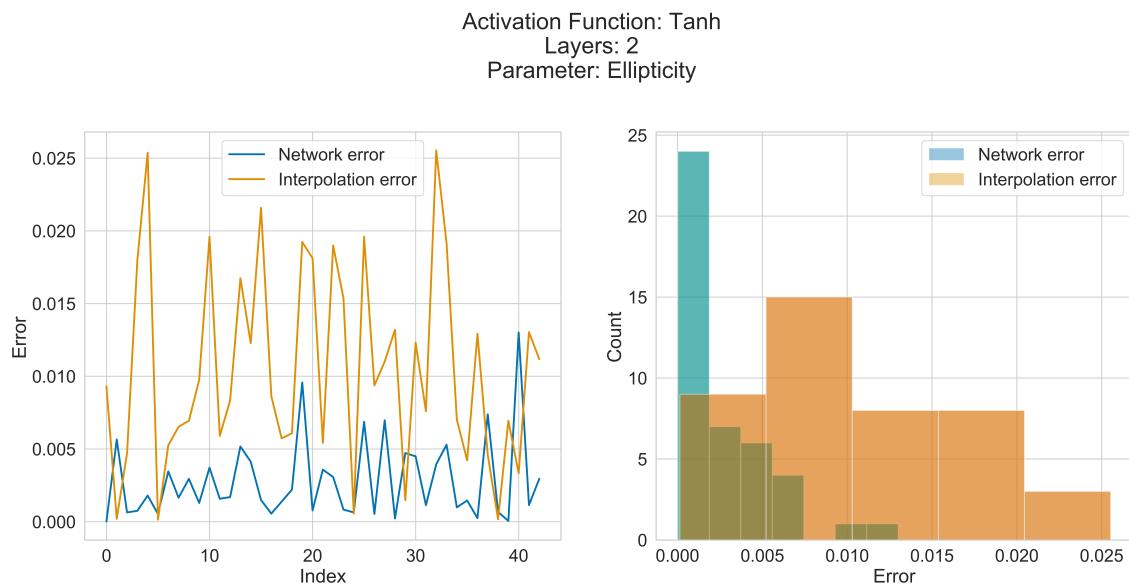


Figura 3.8: Sulla sinistra è rappresentato il confronto dell'errore tra valore vero e valore stimato tramite il metodo di interpolazione, in arancione, e tramite la rete neurale, in blu. Sulla destra è rappresentato lo stesso grafico sotto forma di istogramma.

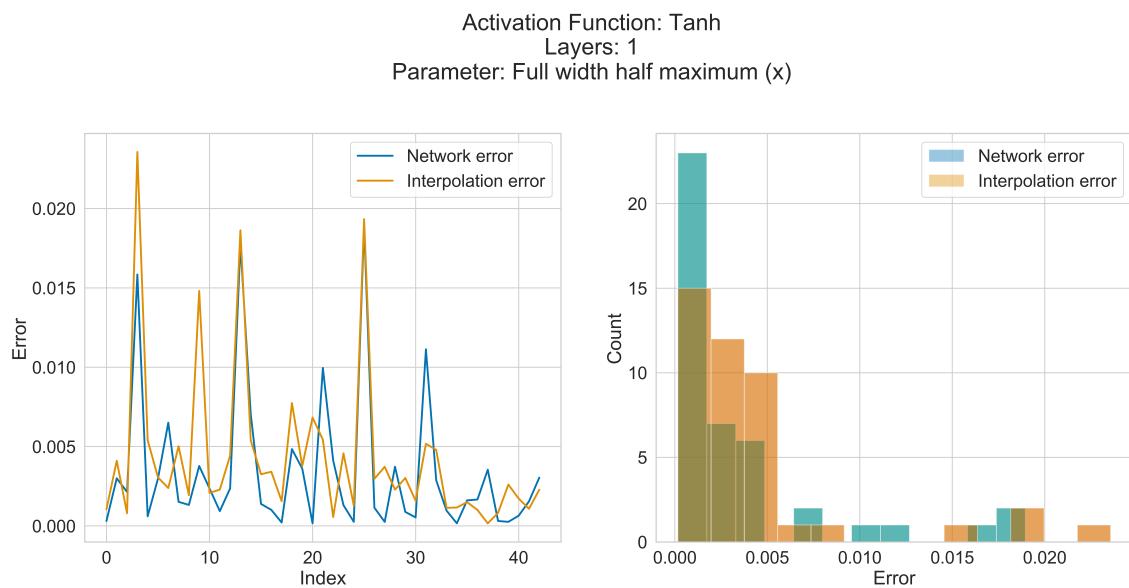


Figura 3.9: Sulla sinistra è rappresentato il confronto dell'errore tra valore vero e valore stimato tramite il metodo di interpolazione, in arancione, e tramite la rete neurale, in blu. Sulla destra è rappresentato lo stesso grafico sotto forma di istogramma.

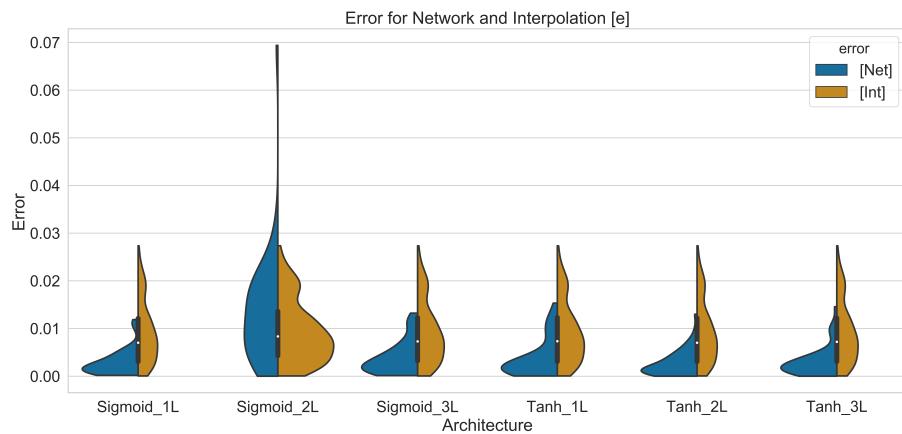
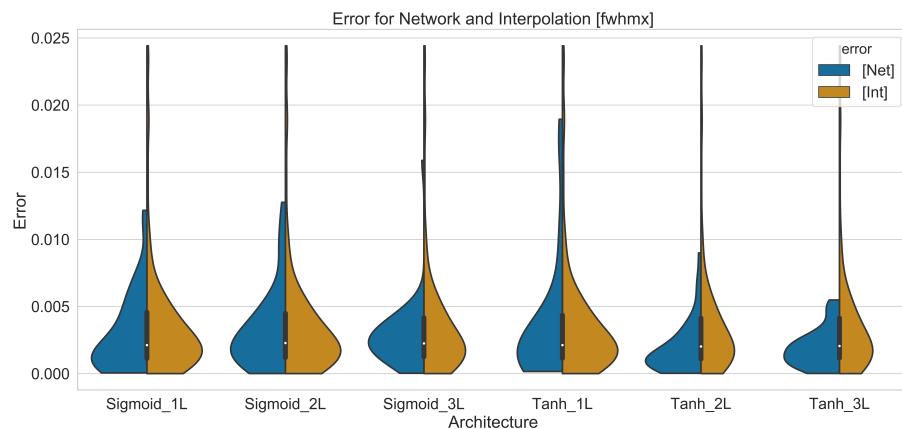
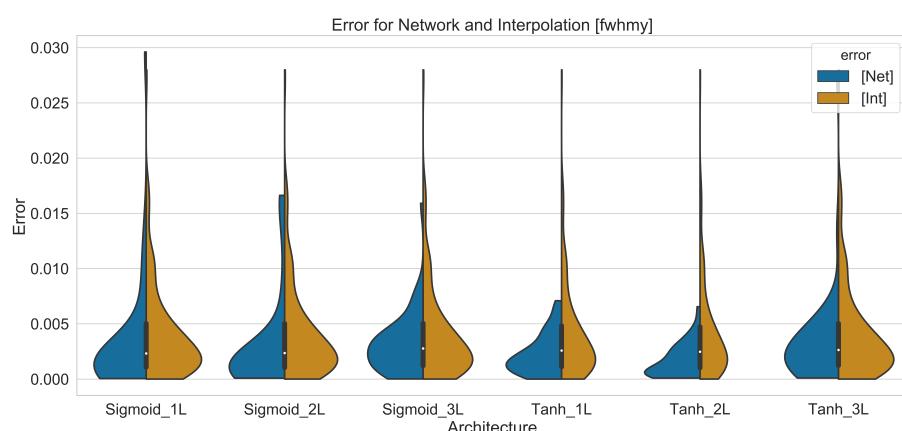


Figura 3.10: Violin plot per l'ellitticità.

Figura 3.11: Violin plot per la $fwhm(x)$.Figura 3.12: Violin plot per la $fwhm(y)$.

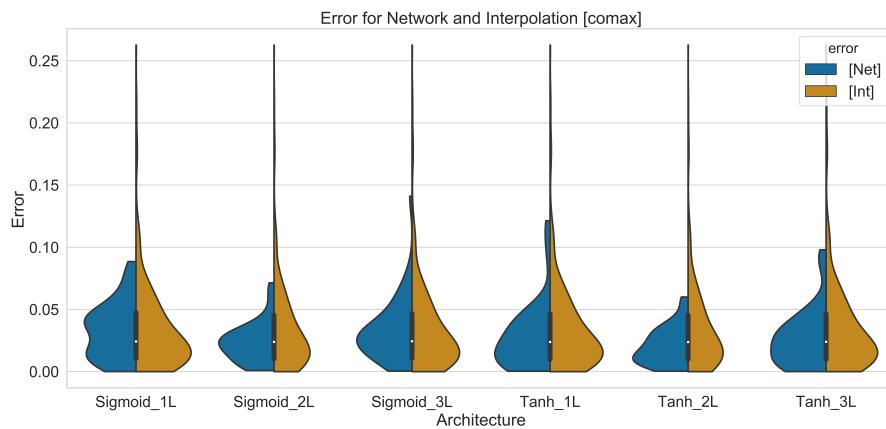


Figura 3.13: Violin plot per la componente co-polare massima.

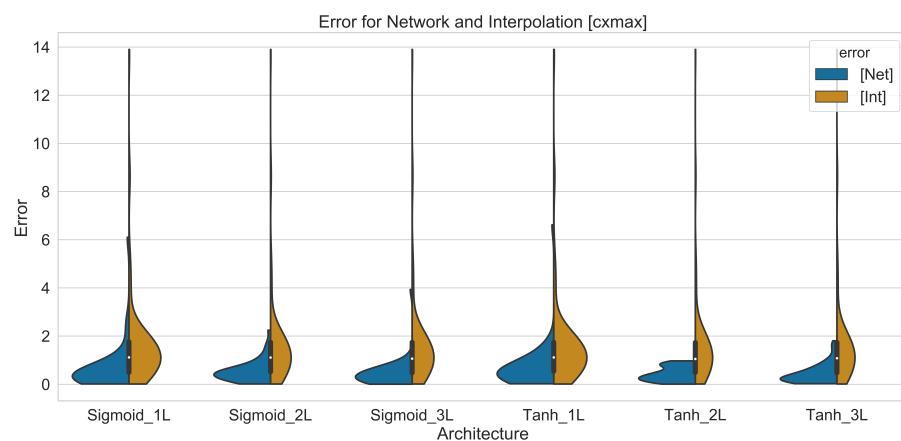


Figura 3.14: Violin plot per la componente cross-polare massima.

Capitolo 4

Conclusioni

La mia speranza è che questa tesi rappresenti un punto di partenza per ulteriori studi futuri. Ho mostrato che le reti neurali hanno un grande potenziale in questo ambito ma la ricerca sull'architettura migliore non la considero terminata. Uno sviluppo futuro riguarda senza dubbio l'applicazione del metodo di stima dei parametri del diagramma di radiazione attraverso le reti neurali a quegli strumenti che presentano sulla loro superficie focale migliaia di detector e per i quali risulta infattibile una simulazione completa con i metodi classici (1.3).

Bibliografia

- [1] K. N. Abazajian, P. Adshead, Z. Ahmed, S. W. Allen, D. Alonso, K. S. Arnold, C. Baccigalupi, J. G. Bartlett, N. Battaglia, B. A. Benson, C. A. Bischoff, J. Borrill, V. Buza, E. Calabrese, R. Caldwell, J. E. Carlstrom, C. L. Chang, T. M. Crawford, F.-Y. Cyr-Racine, F. De Bernardis, T. de Haan, S. di Serego Alighieri, J. Dunkley, C. Dvorkin, J. Errard, G. Fabbian, S. Feeney, S. Ferraro, J. P. Filippini, R. Flauger, G. M. Fuller, V. Gluscevic, D. Green, D. Grin, E. Grohs, J. W. Henning, J. C. Hill, R. Hlozek, G. Holder, W. Holzapfel, W. Hu, K. M. Huffenberger, R. Keskitalo, L. Knox, A. Kosowsky, J. Kovac, E. D. Kovetz, C.-L. Kuo, A. Kusaka, M. Le Jeune, A. T. Lee, M. Lilley, M. Loverde, M. S. Madhavacheril, A. Mantz, D. J. E. Marsh, J. McMahon, P. D. Meerburg, J. Meyers, A. D. Miller, J. B. Munoz, H. N. Nguyen, M. D. Niemack, M. Peloso, J. Peloton, L. Pogosian, C. Pryke, M. Raveri, C. L. Reichardt, G. Rocha, A. Rotti, E. Schaan, M. M. Schmittfull, D. Scott, N. Sehgal, S. Shandera, B. D. Sherwin, T. L. Smith, L. Sorbo, G. D. Starkman, K. T. Story, A. van Engelen, J. D. Vieira, S. Watson, N. Whitehorn, and W. L. Kimmy Wu. CMB-S4 Science Book, First Edition. page arXiv:1610.02743, Oct 2016.
- [2] F. Chollet. *Deep Learning with Python*. Manning, 2018.
- [3] Planck Collaboration, Aghanim, N., Armitage-Caplan, C., Arnaud, M., Ashdown, M., Atrio-Barandela, F., Aumont, J., Baccigalupi, C., Banday, A. J., Barreiro, R. B., Battaner, E., Benabed, K., Benoît, A., Benoit-Lévy, A., Bernard, J.-P., Bersanelli, M., Bielewicz, P., Bobin, J., Bock, J. J., Bonaldi, A., Bonavera, L., Bond, J. R., Borrill, J., Bouchet, F. R., Bridges, M., Bucher, M., Burigana, C., Butler, R. C., Cappellini, B., Cardoso, J.-F., Catalano, A., Chamberlain, A., Chen, X., Chiang, L.-Y., Christensen, P. R., Church, S., Colombi, S., Colombo, L. P. L., Crill, B. P., Curto, A., Cuttaia, F., Danese, L., Davies, R. D., Davis, R. J., de Bernardis, P., de Rosa, A., de Zotti, G., Delabrouille, J., Dickinson, C., Diego, J. M., Dole, H., Donzelli, S., Doré, O., Douspis, M., Dupac, X., Efstathiou, G., Enßlin, T. A., Eriksen, H. K., Finelli, F., Forni, O., Frailis, M., Franceschi, E., Gaier, T. C., Galeotta, S., Ganga, K., Giard, M., Giardino, G., Giraud-Héraud, Y., Gjerløw, E., González-Nuevo, J., Górski, K. M., Gratton, S., Gregorio, A., Gruppuso, A., Hansen, F. K., Hanson, D.,

- Harrison, D., Henrot-Versillé, S., Hernández-Monteagudo, C., Herranz, D., Hildebrandt, S. R., Hivon, E., Hobson, M., Holmes, W. A., Hornstrup, A., Hovest, W., Huffenberger, K. M., Jaffe, A. H., Jaffe, T. R., Jewell, J., Jones, W. C., Juvela, M., Kangaslahti, P., Keihänen, E., Keskitalo, R., Kisner, T. S., Knoche, J., Knox, L., Kunz, M., Kurki-Suonio, H., Lagache, G., Lähteenmäki, A., Lamarre, J.-M., Lasenby, A., Laureijs, R. J., Lawrence, C. R., Leach, S., Leahy, J. P., Leonard, R., Lesgourgues, J., Liguori, M., Lilje, P. B., Linden-Vørnle, M., López-Caniego, M., Lubin, P. M., Macías-Pérez, J. F., Maino, D., Mandolesi, N., Maris, M., Marshall, D. J., Martin, P. G., Martínez-González, E., Masi, S., Massardi, M., Matarrese, S., Matthai, F., Mazzotta, P., Meinhold, P. R., Melchiorri, A., Mendes, L., Mennella, A., Migliaccio, M., Mitra, S., Moneti, A., Montier, L., Morgante, G., Mortlock, D., Moss, A., Munshi, D., Naselsky, P., Natoli, P., Netterfield, C. B., Nørgaard-Nielsen, H. U., Novikov, D., Novikov, I., O'Dwyer, I. J., Osborne, S., Paci, F., Pagano, L., Paladini, R., Paoletti, D., Partridge, B., Pasian, F., Patanchon, G., Pearson, D., Peel, M., Perdereau, O., Perotto, L., Perrotta, F., Pierpaoli, E., Pietrobon, D., Plaszczynski, S., Pointecouteau, E., Polenta, G., Ponthieu, N., Popa, L., Poutanen, T., Pratt, G. W., Prézeau, G., Prunet, S., Puget, J.-L., Rachen, J. P., Rebolo, R., Reinecke, M., Remazeilles, M., Ricciardi, S., Riller, T., Rocha, G., Rosset, C., Rossetti, M., Roudier, G., Rubiño-Martín, J. A., Rusholme, B., Sandri, M., Santos, D., Scott, D., Seiffert, M. D., Shellard, E. P. S., Spencer, L. D., Starck, J.-L., Stolyarov, V., Stompor, R., Sureau, F., Sutton, D., Suur-Uski, A.-S., Sygnet, J.-F., Tauber, J. A., Tavagnacco, D., Terenzi, L., Toffolatti, L., Tomasi, M., Tristram, M., Tucci, M., Tuovinen, J., Türler, M., Umana, G., Valenziano, L., Valiviita, J., Van Tent, B., Varis, J., Vielva, P., Villa, F., Vittorio, N., Wade, L. A., Wandelt, B. D., Watson, R., Wilkinson, A., Yvon, D., Zacchei, A., and Zonca, A. Planck 2013 results. v. Ifi calibration. *A&A*, 571:A5, 2014.
- [4] S. Silver and I. of Electrical Engineers. *Microwave Antenna Theory and Design*. IEE electromagnetic waves series. P. Peregrinus, 1984.
- [5] E. Stevens and L. Antiga. *Deep Learning with PyTorch*. Manning, 2019.
- [6] M. Tomasi, C. Franceschet, and S. Realini. The quest for cmb b-modes.
- [7] T. Wilson and S. Hüttemeister. *Tools of Radio Astronomy: Problems and Solutions*. Astronomy and Astrophysics Library. Springer Berlin Heidelberg, 2012.