Alma Mater Studiorum · Università di Bologna

Dipartimento di Fisica e Astronomia "Augusto Righi"
Corso di Laurea in Fisica

Metodi di deep learning per la classificazione di immagini dal rivelatore RIPTIDE

Relatore:
Prof. Francesco Giacomini

Correlatore:
Dott. Samuele Lanzi

Presentata da:
Alessandro Pinto

Anno Accademico 2023/2024

Abstract

Questa tesi esplora l'applicazione delle reti neurali convoluzionali (CNN) in un innovativo sistema di rivelazione di neutroni basato sulle tecniche di Recoil Proton Track Imaging (RPTI). Queste tecniche sfruttano lo scattering elastico neutrone-protone in uno scintillatore plastico per ricostruire la cinematica dei neutroni a partire da proiezioni ortogonali sul sensore della luce di scintillazione generata dalle tracce dei protoni. Tuttavia, nel volume attivo di uno scintillatore plastico possono avvenire interazioni neutrone-carbonio, che causano la frammentazione nucleare e il rilascio di altre particelle cariche, introducendo un fondo indesiderato nei dati raccolti. Per discriminare tra eventi di segnale e di fondo, è stata sviluppata una rete neurale convoluzionale in grado di classificare con successo le immagini generate dal rivelatore, raggiungendo un'accuratezza superiore al 99%.

Indice

Introduzione

A differenza delle particelle cariche, i neutroni interagiscono con la materia esclusivamente tramite forze nucleari, rendendo la loro rilevazione particolarmente complessa e richiedendo soluzioni innovative. La rilevazione e l'analisi dei neutroni rappresentano quindi una sfida fondamentale in molti campi della fisica moderna, dalla ricerca di base alle applicazioni tecnologiche avanzate.

Questa tesi si propone di esplorare l'uso di tecniche di deep learning per la classificazione delle immagini generate da eventi di scattering di neutroni veloci, nel contesto del progetto RIPTIDE (RecoIl Proton Track Imaging DEtector), un innovativo concetto di tracciatore per il rilevamento di neutroni veloci attraverso il metodo RPTI (Recoil Proton Track Imaging), che sfrutta le tracce luminose generate dai protoni di rinculo nel processo di scattering. L'obiettivo è lo sviluppo di una rete neurale convoluzionale (CNN) capace di distinguere tra le tre tipologie di segnale rilevate da RIPTIDE: singolo scattering, doppio scattering e background, in modo da riservare l'operazione di ricostruzione della traccia ai soli dati rilevanti. Poiché RIPTIDE non è ancora entrato nella fase di presa dati, le immagini utilizzate per l'addestramento e la validazione della rete sono state generate tramite simulazioni Monte Carlo, che hanno permesso di riprodurre realisticamente gli scenari attesi.

Il contenuto di questa tesi è organizzato in tre capitoli:

- Il primo capitolo introduce le applicazioni scientifiche e i principi fisici della rilevazione dei neutroni, insieme a una descrizione del progetto RIPTIDE e della simulazione Monte Carlo utilizzata.

- Nel secondo capitolo è riportata un'introduzione alle reti neurali, illustrandone i concetti fondamentali e ponendo particolare attenzione al modello delle reti neurali convoluzionali (CNN), particolarmente efficaci in compiti quali la classificazione di immagini.

- Nel terzo capitolo è descritta in dettaglio la CNN realizzata per la classificazione dei dati di RIPTIDE e sono riportati i risultati ottenuti. Questo capitolo motiva l'architettura della rete e gli iperparametri scelti per l'addestramento, analizzandone infine la performance.

Capitolo 1

L'importanza dei tracciatori di neutroni

I neutroni, essendo particelle prive di carica, non possono interagire con la materia tramite l'interazione elettromagnetica, che domina nei processi che coinvolgono particelle cariche. Per questo motivo, i neutroni interagiscono con i nuclei del materiale attraversato tramite la forza nucleare forte, e a causa del breve raggio d'azione di quest'ultima, possono viaggiare a lungo nella materia senza subire alcun tipo di interazione.

Il rilevamento di neutroni, poiché non ionizzano direttamente la materia, si basa sulle particelle secondarie prodotte dalle interazioni con i nuclei del materiale rivelatore, che generano particelle cariche o radiazione elettromagnetica, entrambe rilevabili. In base all'energia dei neutroni, possono avvenire diversi tipi di processi nucleari e per questa ragione i neutroni sono classificati proprio in base alla loro energia (Fig. 1.1).

Questo capitolo esplora i motivi per cui il tracciamento dei neutroni veloci è un problema essenziale per la ricerca fondamentale ed applicata e descrive il funzionamento del rivelatore RIPTIDE [1].

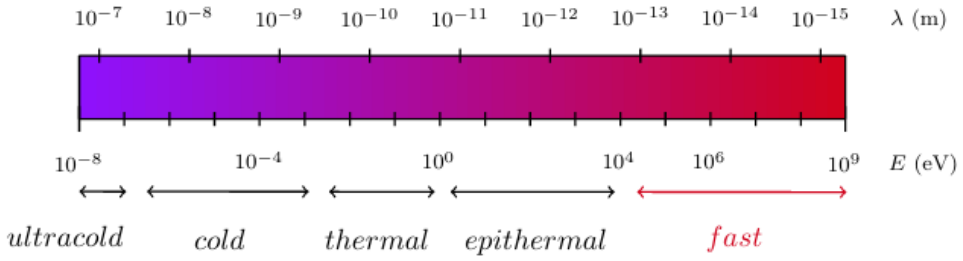


Figura 1.1: *Classificazione dei neutroni in base all'energia cinetica. Sono riportate anche le corrispondenti lunghezze d'onda di de Broglie.*

1.1 Motivazioni scientifiche

Questa sezione evidenzia la rilevanza delle applicazioni di un tracciatore di neutroni in diversi ambiti scientifici, ad esempio in astrofisica, nella radioprotezione spaziale, in medicina nucleare, per esperimenti di fisica nucleare e anche per l'ambiente.

1.1.1 Neutroni solari

Come molti corpi celesti, il Sole emette neutroni che, data la sua vicinanza, possono essere rilevati vicino alla Terra. Durante i periodi di alta attività solare, come le eruzioni solari, le particelle vengono accelerate a energie elevate e, interagendo con l'atmosfera solare, generano neutroni ad alta energia. Alcuni di questi neutroni sfuggono all'attrazione gravitazionale del Sole e, non essendo influenzati dai suoi campi magnetici, viaggiano in linea retta fino alla Terra [2]. Studiandone il flusso, è possibile analizzare la dinamica dei brillamenti solari.

Per monitorare il flusso di neutroni, occorre considerare che essi hanno una vita media di 14.63 minuti e che, durante il tragitto tra il Sole e la Terra, una parte significativa è soggetta a decadimento beta, lasciando arrivare nei pressi della Terra solo il 30% dei neutroni originari. Con strutture a terra è possibile monitorare continuamente la radiazione spaziale, inclusi i neutroni in essa contenuti, ma il passaggio attraverso l'atmosfera terrestre riduce notevolmente l'intensità del flusso di neutroni e per questo motivo, le misurazioni vengono spesso effettuate in osservatori situati ad alta quota o mediante l'uso di satelliti.

Un tracciatore di neutroni sarebbe molto utile per distinguere i neutroni di provenienza solare, che arrivano in linea retta, da quelli generati dall'interazione dei raggi cosmici con l'ambiente.

1.1.2 Esposizione a radiazione in viaggi spaziali

L'esposizione alla radiazione durante le esplorazioni spaziali rappresenta una grave minaccia per la salute degli astronauti a bordo, aumentando il rischio di cancro, danni ai tessuti e avvelenamento da radiazioni. La radiazione spaziale è composta non solo dalle particelle provenienti dal Sole, ma anche dai raggi cosmici. Questi ultimi sono costituiti da atomi completamente ionizzati, originati da lontane supernove, che spaziano dall'idrogeno all'uranio, anche se per la maggior parte sono costituiti da nuclei di ferro o elementi più leggeri. Il numero di raggi cosmici in un dato punto è ridotto, ma, data la massa di tali nuclei e le velocità relativistiche a cui viaggiano, essi generano intensi fenomeni di ionizzazione quando attraversano la materia, rendendo la radiazione spaziale molto più pericolosa rispetto a quella a cui si è esposti sulla Terra.

Sebbene esistano linee guida riguardo all'equipaggio e agli accorgimenti da adottare per le navicelle in orbita terrestre bassa, mancano informazioni adeguate per missioni di lunga durata, come un eventuale viaggio su Marte. Diversamente dalla Terra, Marte non presenta un campo magnetico significativo, e la sua superficie è continuamente esposta a un intenso bombardamento di raggi cosmici e particelle solari ad alta energia. I dati raccolti dal rover Curiosity suggeriscono che, sulla superficie marziana, gli astronauti riceverebbero una dose media giornaliera di radiazione pari a 0.67 mSv, mentre durante il viaggio di 360 giorni (andata e ritorno) riceverebbero una dose giornaliera di circa 1.8 mSv. Complessivamente, durante una missione di questo tipo, un astronauta assorbirebbe una dose di radiazione stimata intorno a 1 Sv, sufficiente a incrementare il rischio di cancro del 5% [3].

Nel tentativo di schermare i raggi cosmici utilizzando scudi composti da materiali densi, sono generati ioni leggeri e neutroni, i quali hanno un elevato impatto biologico. A causa della loro bassa probabilità di interazione, rappresentano una sfida significativa per

i sistemi di schermatura. Nei modelli di fisica nucleare impiegati per prevedere questa produzione, esistono discrepanze dovute alla mancanza di dati sperimentali sullo scattering a grandi angoli e sulla diseccitazione dei raggi cosmici [4]. La disponibilità di un tracciatore di neutroni potrebbe colmare queste lacune, consentendo di raccogliere i dati sperimentali mancanti.

1.1.3 Adroterapia

Nel trattamento di tumori tramite radioterapia, radiazione elettromagnetica è usata per danneggiare il DNA delle cellule tumorali e inibirne così la riproduzione. Nella radioterapia convenzionale con i raggi γ gran parte del corpo del paziente è esposta, aumentando il rischio di tumori indotti da radiazione.

La ricerca attuale si concentra sul colpire le cellule tumorali minimizzando il rischio di esposizione per i tessuti sani. L'adroterapia [5] in particolare è un trattamento che usa atomi interamente ionizzati, quali ioni idrogeno e ioni carbonio, che rilasciano gran parte della loro energia poco prima di arrestarsi, localizzando l'irraggiamento sulla regione dove è il tumore. Il principio di funzionamento dell'adroterapia sfrutta il comportamento peculiare di particelle fortemente cariche interagenti con la materia, noto come curva di Bragg (in Fig. 1.2), per concentrare il rilascio di energia soltanto nella zona tumorale.

Il punto chiave che si può trarre è che durante una sessione di adroterapia i tessuti prima e dopo la posizione del picco vengono investiti da una dose minima di radiazione, mentre se si fossero utilizzati fotoni (come i raggi γ) la dose maggiore di radiazione sarebbe arrivata ai tessuti superficiali, con rapida diminuzione al crescere della profondità. In adroterapia svolgono un ruolo di primo piano le interazioni nucleari, che possono essere di tipo elastico o anelastico. Mentre le interazioni elastiche alterano la direzione e le energie cinetiche delle particelle utilizzate, le interazioni anelastiche causano la loro frammentazione in protoni, neutroni e altre componenti ma, viste le energie utilizzate, non portano alla frammentazione dei nucleoni.

Per garantire un posizionamento accurato del picco di Bragg sulla regione tumorale durante la seduta di adroterapia, anche minimi movimenti del paziente devono essere tenuti in conto per migliorare la precisione. Le macchine per l'irradiazione necessitano quindi di sistemi online di controllo che monitorino in tempo reale la posizione del raggio relativa al paziente. Un modo proposto per ottenere la posizione del raggio, dato che la frammentazione nucleare degli ioni utilizzati in adroterapia produce oltre a protoni e neutroni anche isotopi radioattivi, è l'uso della Positron Emission Tomography (PET), che rivelerebbe i fotoni prodotti per annichilamento tra i positroni prodotti dai nuclei radioattivi e gli elettroni nei tessuti.

Vista la produzione di neutroni durante la frammentazione, un dispositivo in grado di individuarli e ricostruirne in tempo reale la direzione di provenienza potrebbe fungere da alternativa o potenziamento all'uso della PET per monitorare il raggio.

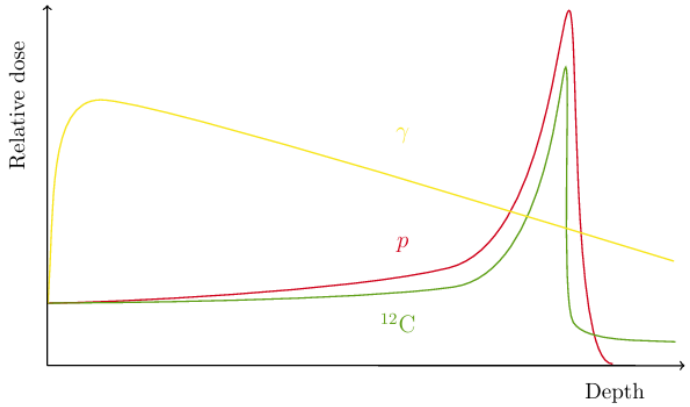


Figura 1.2: *Perdita di energia delle radiazioni ionizzanti in funzione della distanza: per un fascio di fotoni (linea gialla), protoni (linea rossa) e ioni di carbonio (linea verde). Nel caso dei protoni o degli ioni carbonio è evidente il picco a una certa profondità dove è rilasciata la maggior parte della radiazione.*

1.1.4 Esperimenti di fisica nucleare

La facility n_TOF (neutron Time Of Flight) del CERN [6] è stata progettata per lo studio delle interazioni tra neutroni e nuclei, con l'obiettivo di misurare le sezioni d'urto di tali reazioni attraverso il metodo del tempo di volo e la tecnica di attivazione. Questa tecnica consente di analizzare neutroni con energie che spaziano da pochi MeV fino a diversi GeV.

La produzione di neutroni a n_TOF avviene tramite la reazione di spallazione, in cui un fascio di protoni altamente energetici provenienti dal proto-sincrotrone colpisce un massivo bersaglio di piombo. I neutroni generati in questo processo sono inizialmente neutroni veloci e successivamente vengono moderati. Questo passaggio produce un fascio di neutroni con un ampio spettro energetico, adatto per una varietà di studi.

Il fascio di neutroni così generato è poi collimato verso due aree sperimentali, EAR1 ed EAR2, che ospitano esperimenti focalizzati su diversi canali di reazione neutrone-bersaglio.

Di recente è stata fatta una proposta [7] per la misura della lunghezza dello scattering neutrone-neutrone. L'idea è di sfruttare i neutroni finali prodotti durante la rottura di un atomo di deuterio causata da un neutrone:

$$n + {}^{2}_{1}\mathrm{H} \rightarrow p + n + n \tag{1.1.1}$$

per determinare con un solo esperimento la lunghezza di scattering neutrone-neutrone in un ampio range energetico (tra 10 e 100 MeV). L'esperimento è basato sulla rilevazione delle tre particelle uscenti, che porterebbe alla ricostruzione cinematica di un problema a tre corpi. La fattibilità di un simile esperimento richiede l'investigazione della possibilità di usare un bersaglio attivo costituito da uno scintillatore liquido altamente arricchito di deuterio e alla disponibilità di un rilevatore di neutroni come RIPTIDE.

1.1.5 Applicazioni per l'ambiente

Il contenuto di umidità nel suolo gioca un ruolo importante a livello agricolo e ambientale, in quanto sostiene la crescita delle piante, influenza il clima e determina la capacità

del suolo di assorbire e trattenere l'acqua proveniente dalle precipitazioni. Forse sorprendentemente, la rilevazione di neutroni offre un valido strumento per monitorare tale caratteristica [8].

I raggi cosmici, che costituiscono una fonte di radiazione per la Terra, sono particelle altamente energetiche che collidono con le molecole di gas presenti in atmosfera, producendo una cascata di particelle sub-atomiche. Nel tempo necessario a colmare la distanza tra i primi strati atmosferici e il suolo, l'energia contenuta inizialmente nei raggi cosmici è stata convertita attraverso le varie collisioni in un flusso di neutroni veloci vicino alla superficie del terreno. Tali neutroni si disperdono in maniera isotropica e vengono rallentati principalmente attraverso collisioni con gli atomi di idrogeno presenti nel terreno, per poi essere assorbiti dai vari elementi che compongo il suolo. Poiché sono gli atomi di idrogeno quelli con la maggiore capacità di rallentare tali neutroni veloci fino a permetterne l'assorbimento, e poiché la forma più comune in cui si possono trovare atomi di idrogeno nei sistemi terrestri è sotto forma di molecole d'acqua, è possibile trovare una relazione che leghi il conteggio di neutroni con la quantità di acqua in una certa zona del suolo.

L'utilizzo di una simile tecnica beneficerebbe chiaramente della disponibilità di un rilevatore di neutroni sensibile alla direzione di provenienza degli stessi.

1.2 RecoIl Proton Track Imaging DEtector

Allo stato dell'arte, esistono diversi tracciatori di neutroni che sfruttano lo scattering neutrone-protone per ottenere informazioni sulla direzione del neutrone incidente partendo dalla luce di scintillazione del protone. Tra i dispositivi più rappresentativi abbiamo SONTRAC [9], sviluppato per il monitoraggio dei neutroni solari, MONDO [10], progettato per l'adroterapia, e il Weiming-1 CubeSat [11], un sistema innovativo destinato alla rilevazione di neutroni nello spazio.

In questo contesto, si inserisce la proposta di RIPTIDE (RecoIl Proton Track Imaging DEtector), un nuovo concetto di tracciatore descritto in dettaglio in [1]. RIPTIDE si distingue per la capacità di tracciare interazioni neutroniche sia in singolo che in doppio scattering con protoni, aprendo nuove possibilità nella ricostruzione delle proprietà cinematiche dei neutroni.

Questa sezione esplora i fondamenti teorici e sperimentali di RIPTIDE, includendo i principi fisici alla base del rilevamento e della ricostruzione delle tracce, i dettagli del setup sperimentale e il principio di funzionamento complessivo del sistema.

1.2.1 Interazione radiazione materia

Il passaggio di una particella carica pesante (ad esempio protoni o nuclei) attraverso la materia è caratterizzato da una progressiva perdita di energia cinetica, che continua fino al completo arresto della particella. Questo processo avviene principalmente a causa di interazioni elettromagnetiche con gli elettroni atomici del materiale attraversato e, meno frequentemente, con i nuclei. Tali interazioni determinano collisioni che sottraggono energia alla particella incidente.

Consideriamo dunque una radiazione costituita da particelle di numero atomico z che attraversano un bersaglio che ha una densità elettronica per unità di volume N_e, per

una distanza dx. Le perdite di energia delle particelle nell'attraversare il bersaglio sono dunque:

$$-\frac{\mathrm{d}E}{\mathrm{d}x} = 4\pi N_e r_e^2 m_e c^2 \frac{z^2}{\beta^2}\left(\ln\frac{2m_e c^2\beta^2\gamma^2}{I} - \beta^2 - \frac{\delta(\gamma)}{2} - \frac{C}{Z}\right) \tag{1.2.1}$$

dove m_e è la massa dell'elettrone, Z e A sono il numero atomico e di massa del materiale irradiato, β è il rapporto tra la velocità della particella e la velocità della luce, γ è il fattore di Lorentz, ρ è la densità del materiale irradiato, I è il potenziale di eccitazione medio degli atomi del materiale, δ e C sono fattori di correzione [12]. Si può notare che, per energie non relativistiche, il termine dominante è $1/\beta^2$ e quindi l'andamento di dE/dx cresce seguendo una dipendenza logaritmica (vedi Fig. 1.3).

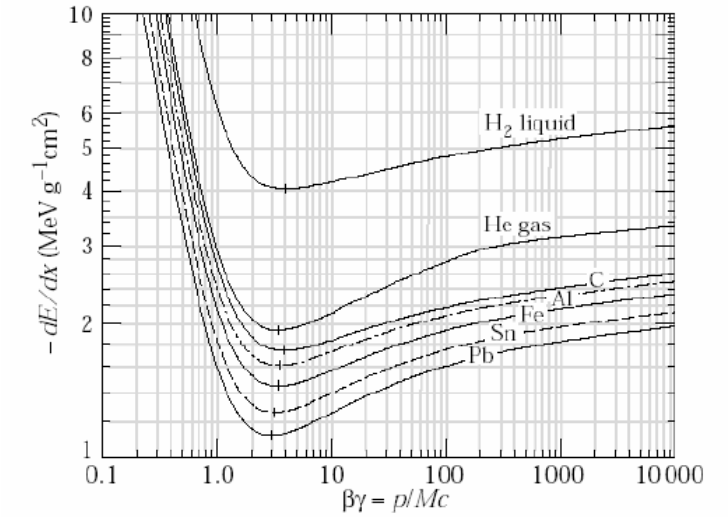


Figura 1.3: *Perdita di energia per unità di percorso in funzione della quantità di moto della particella carica in regime non relativistico [13].*

Una particella carica come il protone, attraversando un mezzo come lo scintillatore, rilascia un valore di energia all'incirca costante, fino alla prossimità del suo arresto, momento in cui la quantità di energia rilasciata cresce esponenzialmente. Se dE/dx fosse espresso in funzione della profondità invece che del momento, si otterrebbe proprio la curva di Bragg (Fig. 1.2).

1.2.2 Tecnica del *Recoil Proton Track Imaging*

Negli ultimi anni, un nuovo metodo per la rilevazione dei neutroni, noto come Recoil Proton Track Imaging (RPTI) [14], ha attirato un notevole interesse. Questo approccio mira a ottenere informazioni sull'energia e sulla direzione dei neutroni analizzando la luce di scintillazione prodotta dalle tracce dei protoni in uno scintillatore. La reazione nucleare preferita per questa tecnica di ricostruzione è lo scattering elastico neutrone-protone. Se la posizione della sorgente di neutroni è nota la tecnica permette di determinare l'energia del neutrone osservando un singolo scattering.

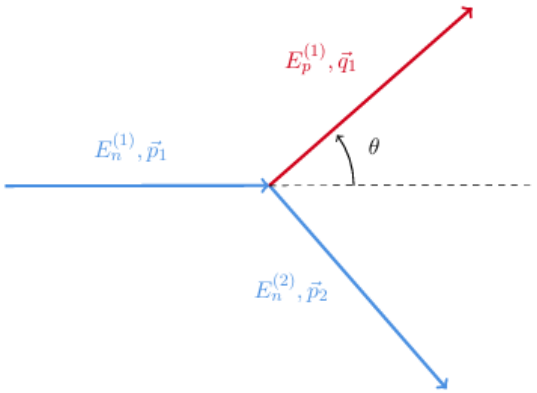


Figura 1.4: *Single scattering n-p.*

Infatti, come possiamo vedere in Fig. 1.4, applicando la conservazione del quadrimomento e approssimando la massa del protone uguale quella del neutrone, si ottiene:

$$|\vec{p}_1| = \frac{|\vec{q}_1|}{\cos\theta} \iff E_n^{(1)} = \frac{E_p^{(1)}}{\cos^2\theta} \tag{1.2.2}$$

dove E_n e E_p sono rispettivamente l'energia del neutrone prima dello scattering e l'energia del protone dopo lo scattering, mentre θ è l'angolo di scattering. L'energia del protone E_p è determinata misurando la lunghezza della traccia del protone, nota come *range*. Il range (R) di una particella carica pesante in un materiale è legato alla sua energia (E) dalla relazione:

$$R(E) = \alpha E^p \tag{1.2.3}$$

Dove α dipende dal materiale e p dal tipo di particella.

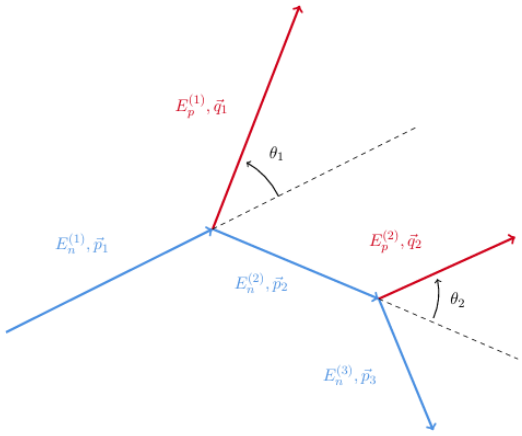


Figura 1.5: *Schema del processo di doppio scattering n-p.*

Se la posizione della sorgente di neutroni non è nota, è possibile determinare sia l'energia sia la direzione del neutrone incidente tramite un evento di doppio scattering neutrone-protone (Fig. 1.5). Infatti, collegando i due vertici di interazione è possibile

stimare l'angolo θ_2 e utilizzando due volte l'equazione 1.2.2 è possibile ricostruire sia θ_1 che $E_n^{(1)}$.

1.2.3 L'idea del detector

RIPTIDE (RecoIl Proton Track Imaging DEtector) è costituito da un cubo di scintillatore plastico realizzato in polivinilitoluene (BC-408), con un volume pari a $6\times6\times6$ cm^3. Questo materiale è composto principalmente da carbonio e idrogeno, rendendolo ideale per le applicazioni di imaging basate sulla diffusione elastica neutrone-protone. All'interno del volume dello scintillatore i neutroni possono interagire elasticamente sia con gli atomi di idrogeno sia con i nuclei di carbonio. Nel range di energie considerate, il range del carbonio di rinculo è troppo piccolo rispetto alla risoluzione spaziale del sistema, rendendo queste interazioni non rilevabili. Questa limitazione ha un impatto diretto sull'efficienza del sistema. Ad esempio, se un neutrone interagisce prima con un nucleo di carbonio e successivamente con un protone, il doppio scattering è erroneamente interpretato come singolo, portando a errori nella stima dell'energia iniziale del neutrone. Per stimare l'efficienza del sistema, è essenziale valutare le probabilità di interazione dei neutroni con il carbonio nello scintillatore.

Un sistema ottico, composto da una lente e un sensore per la registrazione dei fotoni, fornisce l'immagine della luce nello scintillatore. Per la ricostruzione 3D delle tracce sono necessari almeno due dispositivi ottici, sincronizzati tramite un trigger come un fotomoltiplicatore. La configurazione è illustrata in Fig. 1.6.

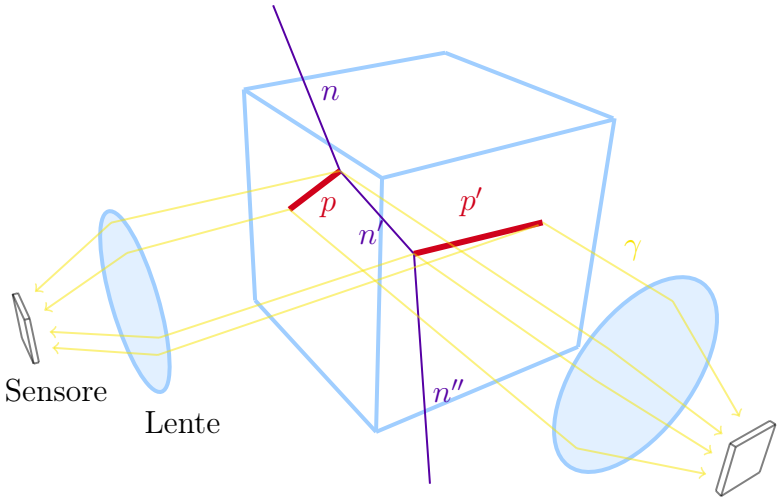


Figura 1.6: *Lo schema del concetto del rivelatore RIPTIDE.*

Grazie alla sua configurazione basata su un unico cubo di scintillatore plastico, RIPTIDE offre numerosi vantaggi rispetto a rivelatori come SONTRAC e MONDO, che utilizzano matrici di fibre scintillanti. In particolare, RIPTIDE supera le limitazioni di risoluzione spaziale imposte dalla dimensione delle fibre. Nei rivelatori a fibre, la ricostruzione della traccia del protone si basa sull'analisi della luce prodotta in ciascuna fibra, mentre in RIPTIDE i fotoni possono essere generati in tutto il volume dello scintillatore. Un protone che attraversa lo scintillatore plastico produce circa 10^3 fotoni per MeV. Ad esempio, un protone con energia 3.5 MeV genera circa 10^4 fotoni lungo una traccia di 200 μm, che

corrisponde alla risoluzione spaziale target del sistema. Un'altra limitazione delle fibre scintillanti è che il protone deve essere sufficientemente energetico per attraversarne almeno tre. Per SONTRAC, ciò corrisponde a una distanza di circa 600 μm, traducendosi in un'energia minima rilevabile del protone pari a circa 6 MeV.

1.2.4 Setup e componenti

Come abbiamo introdotto nella sezione precedente, RIPTIDE è costituito da tre elementi principali: scintillatore, sistema di acquisizione e trigger. Entriamo più nel dettaglio nel descrivere i componenti del detector:

Scintillatore La componente principale del rivelatore è lo scintillatore plastico BC-408, selezionato per la sua eccellente risoluzione temporale con tempi di salita e discesa nell'ordine del nanosecondo. Il materiale presenta un buon rapporto carbonio-idrogeno, pari a 0.9. La luce emessa ha un picco a $\lambda = 430$ nm, ideale per il sistema ottico. Le dimensioni del cubo scintillatore, pari a 6 cm per lato, sono state scelte per ottimizzare lo scattering dei neutroni e minimizzare la fuoriuscita dei protoni, considerando un'energia massima di 50 MeV.

Sistema di acquisizione La scelta del sistema ottico è cruciale per garantire alta risoluzione spaziale e buona efficienza nella rivelazione dei fotoni. Due opzioni sono attualmente in valutazione:

- Sensori CMOS retroilluminati: offrono elevata sensibilità e alta efficienza quantica, con una risoluzione spaziale adeguata e basso consumo energetico.
- Microchannel Plate (MCP): garantiscono una risoluzione spaziale tra 10 μm e 100 μm e una risoluzione temporale di 100-200 ps. Gli MCP amplificano i segnali elettronici generati da fotocatodi e possono essere accoppiati ai sensori CMOS.

Trigger RIPTIDE utilizza un fotomoltiplicatore come trigger per sincronizzare il sistema ottico con il passaggio delle particelle.

1.2.5 Simulazione Monte Carlo

Il progetto RIPTIDE è appena nato e rappresenta un grande sfida dal punto di vista sperimentale. Per studiare e ottimizzare la costrizione del detector sono state fatte delle simulazioni basate sul toolkit Geant4 [15], versione 11.4.1. Il cubo scintillatore, di dimensioni $6\times6\times6$ cm^3, è stato modellato come poliviniltoluene (`G4_PLASTIC_SC_VINYLTOLUENE`), con densità di 1.032 g/cm^3, energia di ionizzazione pari a 64.7 eV e rapporto C/H di 9:10. Le simulazioni includono la produzione e il trasporto di fotoni ottici, con proprietà ottiche basate sui dati del BC-408: 10^4 fotoni per MeV, indice di rifrazione $n = 1.59$ e una lunghezza di assorbimento molto superiore alle dimensioni del cubo.

Grazie alla simulazione siamo in grado di produrre un fascio di neutroni a diverse energie e registrare i prodotti delle reazioni che possono avvenire nel volume attivo del rivelatore. Per le particelle cariche sono inoltre registrati tutti i punti di emissione e le rispettive direzioni iniziali dei fotoni ottici generati. Questi dati rappresentano l'input

per un altro software personalizzato che simula la propagazione dei fotoni ottici dal cubo al sensore, ottenendo proiezioni bidimensionali di una faccia del cubo scintillatore. Il sistema ottico comprende una lente che introduce un ingrandimento di -0.5 e un sensore di dimensione 20 mm composto da 100×100 pixels, scelto per coprire un volume attivo interno di 40 mm per lato. Un esempio di immagine della luce di scintillazione di un doppio scattering neutrone-protone generata e propagata sul sensore dal codice Monte Carlo è riportato in Fig. 1.7.

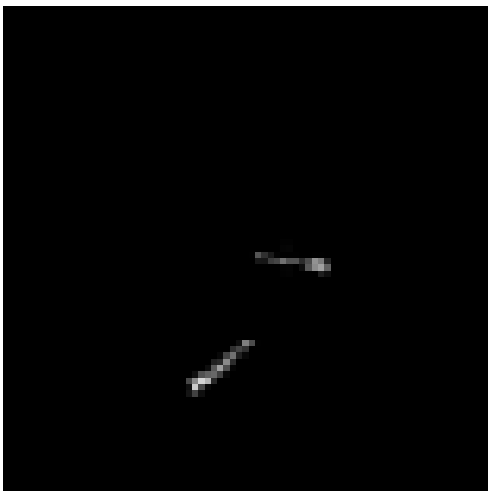


Figura 1.7: *Esempio di immagine della luce di scintillazione di un doppio scattering neutrone-protone generata e propagata sul sensore dal codice Monte Carlo.*

Capitolo 2

Reti neurali

Secondo la definizione di Wechsler, «L'intelligenza è la capacità globale e complessa dell'individuo di agire per uno scopo determinato, pensare in maniera razionale e affrontare efficacemente il proprio ambiente» [16]. L'intelligenza artificiale rappresenta l'ambito che mira a fornire ai computer proprio questa capacità. All'interno di questo campo, il *machine learning* e il *deep learning* costituiscono due tecniche fondamentali per raggiungere tali obiettivi.

Il machine learning è focalizzato sull'addestramento di algoritmi affinché apprendano autonomamente da dati specifici, adattandosi man mano a nuove informazioni. Tra i vari approcci di machine learning, il deep learning si distingue per l'uso di *deep neural networks*, in grado di apprendere rappresentazioni complesse dei dati. Le reti neurali appena descritte risultano particolarmente efficaci in applicazioni quali il riconoscimento delle immagini, l'elaborazione del linguaggio naturale e il riconoscimento vocale.

Lo sviluppo delle reti neurali ha radici storiche che risalgono al secolo scorso, quando Frank Rosenblatt propose il primo modello ispirato al neurone umano, il *perceptron* [17], gettando le basi della ricerca nel settore. Negli anni '80, l'introduzione delle reti neurali multistrato e dell'algoritmo di *backpropagation*, sviluppato da Rumelhart, Hinton e Williams [18], ha segnato un significativo progresso, consentendo per la prima volta alle reti di apprendere in modo efficace. Nonostante questi successi, le reti neurali hanno mostrato limitazioni significative, incentivando la ricerca verso approcci più avanzati come le reti neurali convoluzionali (CNN) [19]. Le CNN, ottimizzate per compiti complessi come l'analisi delle immagini, rappresentano oggi uno dei capisaldi del deep learning.

L'attuale esplosione di interesse e risultati nel campo è stata possibile solo grazie a due fattori principali: il rapido aumento della potenza computazionale, in particolare tramite l'uso delle GPU, e la disponibilità di enormi dataset per l'addestramento delle reti. Questo contesto ha reso le reti neurali strumenti fondamentali per affrontare problemi complessi e ampliare i confini della ricerca scientifica e tecnologica.

2.1 Introduzione alle reti neurali

Il mondo delle reti neurali è vasto e in costante evoluzione, con una varietà di architetture e tecniche progettate per affrontare problemi specifici. Per questa ragione, in questa sezione ci concentreremo sui concetti fondamentali delle reti neurali, con particolare attenzione alle reti neurali multistrato.

Le reti multistrato, sebbene siano più semplici rispetto a modelli più avanzati come le reti convoluzionali, costituiscono la base teorica e pratica su cui si fondano molte tecniche moderne e rappresentano un esempio ideale per introdurre i principi di funzionamento delle reti neurali artificiali come l'architettura a strati e il processo di apprendimento.

2.1.1 Perceptron

Ogni rete neurale è costituita da una unità fondamentale: il perceptron, che svolge nella rete un ruolo analogo a quello di un neurone biologico. Un perceptron (vedi Fig. 2.1) presenta diverse connessioni, ognuna associata a un peso w_i, con le quali riceve gli input $(x_1, x_2, \ldots, x_m)$.

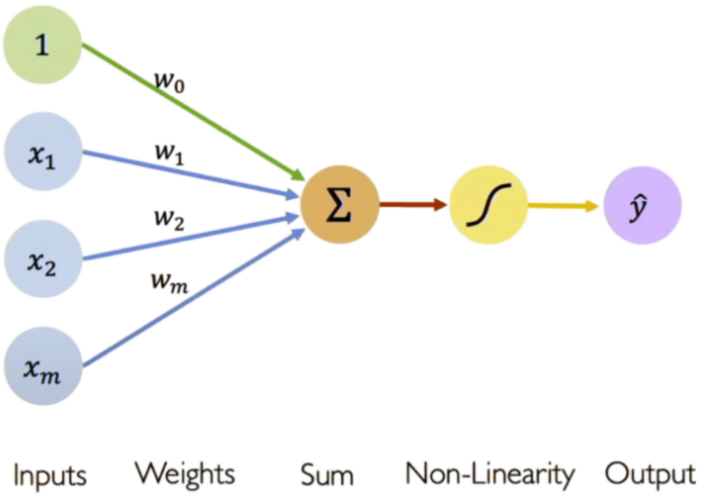


Figura 2.1: *Rappresentazione schematica di un perceptron dove sono indicati gli input (in blu) e il bias (in verde), l'operazione di somma pesata (in arancione) e l'applicazione di una funzione di attivazione (in giallo) [20].*

Queste informazioni sono rielaborate internamente dal perceptron, che calcola la media pesata degli input ricevuti, a cui aggiunge il bias w_0. Al risultato viene applicata la funzione di attivazione g, che produce il valore di output $\hat{y}$:

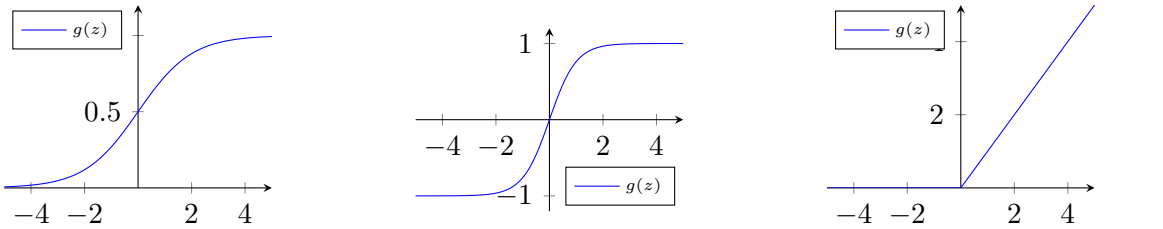
$$\hat{y} = g\left(w_0 + \sum_{i=1}^{m} x_i w_i\right) \tag{2.1.1}$$

Oltre ai pesi associati agli input, un perceptron è quindi caratterizzato anche da un bias, che gli permette di traslare orizzontalmente la funzione di attivazione.

La funzione di attivazione, anche detta funzione di non linearità, è introdotta in una rete neurale per aumentarne l'espressività e la compattezza, ma soprattutto per permetterle di adattarsi a comportamenti non lineari, che rappresentano la maggioranza dei casi pratici. Una rete neurale con funzione di attivazione lineare, per quanto ricca possa essere, resterebbe incapace di risolvere problemi complessi, dato che la combinazione lineare di funzioni lineari è ancora una funzione lineare.

Vi sono diversi tipi di funzioni di attivazione. Tra le più usate spiccano: la sigmoide, la tangente iperbolica e la ReLU (Rectified Linear Unit).

Sigmoid Function **Hyperbolic Tangent** **Rectified Linear Unit**



$$g(z) = \frac{1}{1+e^{-z}} \qquad g(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}} \qquad g(z) = \max(0, z)$$

Ognuna di queste opzioni è differente ma tutte fungono in pratica da funzioni di soglia, sopprimendo l'output del neurone se non è stato eccitato a sufficienza. La funzione sigmoide e la tangente iperbolica restituiscono un valore basso se ricevono in argomento valori negativi e un valore alto se ricevono in argomento valori positivi, con la caratteristica che indipendentemente dal valore in ingresso, l'output è sempre compreso tra 0 ed 1 per la sigmoide e -1 e 1 per la tangente iperbolica. La Rectified Linear Unit (ReLU) invece ferma i valori negativi ponendo l'output del neurone a zero, mentre lascia passare inalterati i valori positivi. È diventata molto popolare negli ultimi anni grazie alla facilità con cui introduce la non-linearità nel sistema e uno dei suoi principali vantaggi rispetto alla sigmoide o alla tangente iperbolica è la sua semplicità computazionale sia per la propagazione delle informazioni in avanti che durante l'algoritmo di backpropagation.

Costruiamo ora una rete neurale costituita da uno strato di neuroni di input, un solo strato di neuroni nascosti e uno strato di neuroni di output, organizzando i perceptron come in Fig. 2.2.

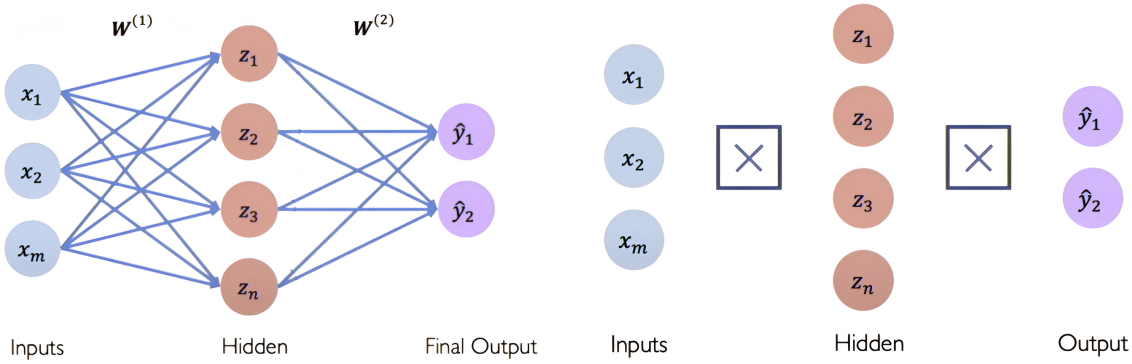


Figura 2.2: *A sinistra notiamo la struttura di una rete neurale a singolo strato dove ogni neurone di input è collegato tramite connessioni pesate a ogni neurone dello strato nascosto, e ogni neurone dello strato nascosto è connesso ai neuroni dello strato di output tramite la funzione di attivazione. A destra le connessioni della stessa rete sono state semplificate con una croce [20].*

In una simile architettura gli strati della rete sono detti densi o *fully connected* (FC), poiché ogni neurone è connesso a tutti i neuroni dello strato precedente e a tutti i neuroni dello strato successivo. Occorre precisare che i neuroni dello strato di input non svolgono operazioni ma servono solo a contenere e passare correttamente i dati al primo strato

nascosto. Lo strato dei neuroni di ingresso deve avere infatti le stesse dimensioni degli input che la rete deve ricevere. L'elaborazione è svolta solo dallo strato nascosto e dallo strato di output, i cui neuroni effettuano l'ultima elaborazione e fungono da contenitore per i risultati. Indicando con $\hat{y}_i$ il valore in uscita di un neurone di output e con z_i il valore in uscita di un neurone nascosto, si ottengono le seguenti formule:

$$\hat{y}_i = g\left(w_{0,i}^{(2)} + \sum_{j=1}^{n} \mathbf{z_j}\, w_{j,i}^{(2)}\right) \quad \text{dove} \quad z_i = g\left(w_{0,i}^{(1)} + \sum_{j=1}^{m} x_j\, w_{j,i}^{(1)}\right) \tag{2.1.2}$$

Possiamo ora introdurre le reti neurali multistrato, ottenibili aumentando il numero di strati di neuroni nascosti come in Fig. 2.3.

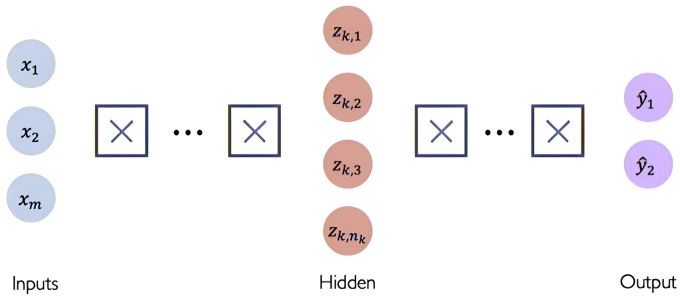


Figura 2.3: *Architettura di una rete neurale multistrato dove le connessioni tra i neuroni di strati successivi sono state indicate con una croce [20].*

Mantenendo la notazione introdotta per la rete neurale a singolo strato, il valore di uscita da un neurone nascosto nello strato k e in posizione i è indicato con $z_{k,i}$:

$$z_{k,i} = g\left(w_{0,i}^{(k)} + \sum_{j=1}^{n_{k-1}} z_{k-1,j}\, w_{j,i}^{(k)}\right) \tag{2.1.3}$$

dove $w_{0,i}^{(k)}$ è il bias associato al neurone, $z_{k-1,j}$ sono i valori in uscita dei neuroni dello strato precedente e $w_{j,i}^{(k)}$ sono i pesi delle connessioni corrispondenti.

2.1.2 Funzione di loss

A questo punto la nostra rete non conosce ancora nulla del compito che ci aspettiamo risolva. Per insegnarle dovremo chiederle di analizzare i dati e, nel caso in cui sbagli, correggerla in base all'errore fatto.

Per effettuare il processo di addestramento introduciamo la *loss function* o funzione di loss $\mathcal{L}$, che è calcolata per ogni vettore di dati $\mathbf{X} = (x_1, x_2, \ldots, x_m)$ e confronta il valore del vettore degli output della rete $\hat{Y}$ con quello delle risposte corrette Y, restituendo un numero che quantifica l'errore fatto:

$$\mathcal{L}(\hat{Y}(\mathbf{X}; \mathbf{W}; \mathbf{B}), Y) \tag{2.1.4}$$

dove $\hat{Y}$ dipende dal vettore di input $\mathbf{X}$, da $\mathbf{W}$ che è il vettore che contiene i pesi e da $\mathbf{B}$ che è il vettore contenente i bias della rete. $\mathcal{L}$ è la loss su un singolo vettore di dati ma

ciò che è di fatto utilizzata è la J che è la loss media su più campioni di dati, che sono indicati con l'indice (i):

$$J(\mathbf{W};\mathbf{B}) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(\hat{Y}^{(i)}(\mathbf{X}^{(i)};\mathbf{W};\mathbf{B}), Y^{(i)}) \tag{2.1.5}$$

Possiamo considerare la J come dipendente unicamente dai pesi $\mathbf{W}$ e bias $\mathbf{B}$ della rete, infatti sia i dati che le risposte corrette sono costanti e non dei parametri che la rete può variare per migliorarsi.

La funzione di loss è parte integrante del processo di addestramento perché, oltre a misurare la prestazione della rete, influenza le dinamiche di apprendimento e la sua velocità. Visto che ogni funzione di loss premia e scoraggia la rete in modo diverso, occorre quindi selezionare la più appropriata in base al compito specifico da svolgere.

Vedremo ora due tra le funzioni di loss più comunemente usate: la *Mean Squared Error* (MSE) loss e la *Cross-Entropy* (CE) loss.
La MSE loss, utilizzata molto nei problemi di regressione dove la rete deve predire un solo numero ma con valore continuo, è la media del quadrato dei residui, termine con cui si indica la differenza tra la previsione del modello e quella effettiva:

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{2.1.6}$$

dove $\hat{y}_i$ è il valore previsto e y_i il valore atteso. Elevare al quadrato i residui evita che il modello interpreti erroneamente di avere una buona performance anche in presenza di errori. I residui, infatti, possono essere sia positivi che negativi, e la loro somma potrebbe risultare prossima a zero anche quando i singoli errori siano significativi. Questa funzione inoltre penalizza il modello più pesantemente per gli errori più grandi, aiutandolo a convergere più velocemente verso il valore minimo.

La CE loss è utilizzata per i problemi di classificazione multiclasse, nei quali la rete ha più neuroni di output ma deve accenderne solo uno, ed il risultato è una distribuzione di probabilità su più classi. La Cross-Entropy è definita come:

$$CE = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k} y_j^{(i)} \log(\hat{y}_j^{(i)}) \tag{2.1.7}$$

dove k è il numero totale di classi, $\hat{y}_j^{(i)}$ è il valore predetto per la classe numero j e $y_j^{(i)}$ è il valore atteso per quella classe (1 se il dato da classificare appartiene alla classe j, 0 in caso contrario). È importante notare che solo il termine corrispondente all'y_j uguale a 1 contribuisce alla perdita, poiché tutti gli altri termini sono moltiplicati per 0. Il logaritmo penalizza fortemente il modello quando sbaglia classe con sicurezza cioè quando prevede una probabilità quasi nulla per la classe corretta, in quanto il logaritmo diverge per argomenti vicini a zero, mentre è meno severo per gli errori più incerti.

A questo punto occorre trovare i pesi e i bias che minimizzano la nostra funzione di loss media $J(\mathbf{W};\mathbf{B})$ e siccome il processo per la minimizzazione di pesi e bias è analogo, ci concentreremo in dettaglio solo sul primo. J può essere rappresentata nello spazio dei pesi come in Fig. 2.4.

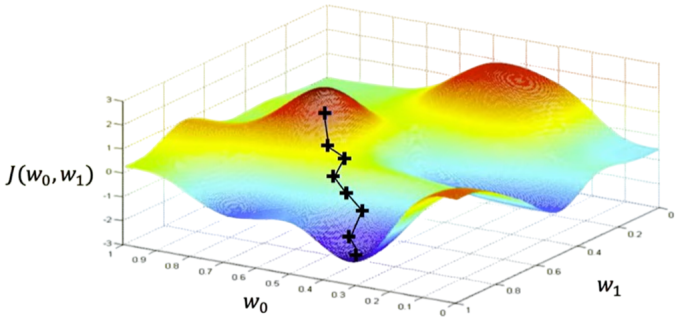


Figura 2.4: *Grafico della funzione di loss per una rete neurale con 2 soli pesi w_o e w_1 [20].*

Per trovare il minimo della funzione non è possibile calcolarla in ogni punto perché lo spazio dei parametri non è chiuso (ogni peso e bias va da $-\infty$ a $+\infty$) e perché il numero di parametri si aggira per i sistemi più complessi nell'ordine dei miliardi. Si utilizza quindi come strategia la cosiddetta discesa del gradiente, che non garantisce di trovare il minimo assoluto ma permette di trovare velocemente un minimo locale. Per questo motivo le funzioni di loss sono scelte spesso anche in base alla facilità con cui è calcolabile il loro gradiente. La discesa del gradiente è un algoritmo che è composto da tre fasi:

1. Inizializzazione in maniera casuale dei pesi
2. Calcolo del gradiente $\frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$
3. Aggiornamento dei pesi $\mathbf{W} \rightarrow \mathbf{W} - \eta \frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$

Dato che il gradiente fornisce la direzione in cui la funzione di loss cresce più ripidamente, spostandosi in direzione opposta di un piccolo quantitativo si ottiene una sua riduzione. Ripetendo i passaggi 2 e 3, se la scelta del passo è adeguata, è possibile stabilizzarsi in un minimo locale della funzione dei loss.

2.1.3 Calcolo del gradiente

Il gradiente è calcolato con l'algoritmo di backpropagation che applica la regola per il calcolo della derivata di una funzione composta in modo strutturato ed efficiente. Questo algoritmo infatti struttura il calcolo del gradiente e memorizza alcuni risultati di calcolo intermedi per evitare operazioni ridondanti. Lo scopo della backpropagation è il calcolo delle derivate parziali $\partial J/\partial w$ e $\partial J/\partial b$ della funzione di loss media J rispetto a ogni peso w e bias b nella rete, anche se in realtà ciò che la backpropagation ci permette di calcolare è il calcolo delle derivate parziali per un singolo vettore di dati $\partial \mathcal{L}/\partial w$ e $\partial \mathcal{L}/\partial b$, da cui possiamo passare a $\partial J/\partial w$ e $\partial J/\partial b$ mediando su più dati [21]. Per semplicità immaginiamo come esempio di avere una rete con un solo input, un solo strato nascosto fatto da un neurone e un solo output (Fig. 2.5) e di ignorare la distinzione tra J e $\mathcal{L}$.

Figura 2.5: *Rete neurale a singolo strato con un solo neurone nascosto, un input e un output [20].*

Abbiamo quindi due connessioni e quindi due pesi w_1 e w_2, dove w_1 è il peso della connessione tra il dato e il neurone nascosto e w_2 è il peso associato alla connessione tra il neurone nascosto e il neurone di output. Le due componenti del gradiente saranno quindi:

$$\frac{\partial J(\mathbf{W})}{\partial w_2} = \frac{\partial J(\mathbf{W})}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_2} \quad \frac{\partial J(\mathbf{W})}{\partial w_1} = \frac{\partial J(\mathbf{W})}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_1} \tag{2.1.8}$$

Notiamo come sia più facile calcolare le derivate di J rispetto ai pesi delle connessioni vicine allo strato di output, perché con la derivazione è sufficiente fermarsi prima, invece nel caso di $\partial J(\mathbf{W})/\partial w_1$, visto che $\hat{y}$ non dipende direttamente da w_1, occorre procedere fino a z_1.

Lo stesso procedimento, come già accennato, è poi ripetuto per il calcolo del gradiente di J rispetto ai bias, in quanto ad ogni addestramento vengono corretti sia i pesi che i bias della rete.

2.1.4 Iperparametri

Gli iperparametri sono gli elementi della rete che definiscono l'architettura della rete e il processo di addestramento, che è necessario impostare. Essi non sono ottimizzati dalla rete neurale durante il processo di addestramento. Tra essi vi sono il numero di strati e di neuroni della rete, la funzione di attivazione, la dimensione della batch, il numero di epoche e il *learning rate*. La dimensione della batch è il numero di campioni n usato per il calcolo della funzione di loss media J, mentre il numero di epoche indica quanto a lungo viene addestrato il modello.

Il learning rate invece è un iperparametro che abbiamo introdotto parlando della discesa del gradiente quando abbiamo immaginato di fare un passo η nella direzione opposta a quella del gradiente, ed è uno degli iperparametri che influenza maggiormente l'addestramento della rete. Un learning rate grande permette al modello di imparare velocemente ma con il rischio di non stabilizzarsi in un minimo, mentre un learning rate piccolo garantisce una discesa meticolosa ma ha un'alta probabilità di bloccare il modello in un minimo locale. Per questo motivo si utilizza spesso un learning rate adattativo che cambia durante l'addestramento. Uno dei learning rate più intuitivi si basa sulla ripidità della discesa e diminuisce man mano che il modulo del gradiente si avvicina a zero.

2.1.5 Overfitting

Un problema caratteristico dell'addestramento delle reti neurali è l'*overfitting*, che avviene quando la rete si adatta troppo bene a dati di addestramento e non riesce a fare previsioni accurate su altri dati. Questo può succedere in caso di addestramento troppo lungo o quando il modello è troppo complesso e inizia a imparare informazioni irrilevanti dai dati. Nel primo caso l'addestramento deve essere interrotto quando la funzione di loss, calcolata sui dati di testing, raggiunge un minimo stabile (il cosiddetto *early stopping*), nel secondo caso invece si può o ridurre la complessità della rete o utilizzare degli strati di *dropout* che spengono casualmente una percentuale prestabilita di neuroni in un certo strato, rendendo la rete più robusta e meno tendente a memorizzare del rumore.

2.2 Reti neurali convoluzionali (CNN)

In questa sezione vedremo come fare a dare a un computer la capacità di vedere. Un'immagine non è altro che una matrice di pixel ossia di numeri e nel caso di immagini *greyscale* ogni pixel corrisponde a un solo numero, mentre per immagini RGB (a colori) ogni pixel ha tre numeri che indicano quanto è rosso, blu e verde.

Il primo problema da affrontare è in che modo fornire le immagini al computer per permettergli di analizzarle e fornire una risposta. Per passare un'immagine a una rete neurale multistrato occorre appiattirla in un vettore, ma questo comporta la perdita delle informazioni sulle posizioni reciproche dei pixel e soprattutto richiede l'utilizzo di tantissimi parametri, visto che ogni singolo neurone è collegato a tutti i neuroni dello strato precedente. Per esempio, con un'immagine in ingresso di dimensioni $28\times28\times1$ (28 pixel di altezza, 28 pixel di larghezza, e un pixel di profondità) ogni singolo neurone nel primo strato nascosto di una rete neurale multistrato deve avere 784 pesi che può sembrare un numero ragionevole, ma già per immagini che hanno 100 pixel per lato, ci sarebbe un carico di 10^4 pesi per singolo neurone.

È in questo contesto che si inseriscono le reti neurali convoluzionali (CNN), che sono oggi una delle architetture di deep learning più diffuse per l'elaborazione di dati con topologia simile a una griglia. Le CNN hanno infatti il vantaggio di assumere che gli input siano rigorosamente immagini e con questo presupposto, mentre una rete neurale riceve l'input come un vettore, una CNN ha i neuroni dei primi strati disposti in tre dimensioni: altezza, larghezza e profondità.

Vi sono diverse applicazioni che le CNN possono svolgere. Nell'ambito dei compiti relativi alla visione artificiale possiamo distinguere quattro tipologie principali di problemi: classificazione, localizzazione, segmentazione e controllo probabilistico [22].

Per il task di classificazione, centrale per questa tesi, il compito della rete è quello di classificare un oggetto, ovvero effettuare una mappatura dallo spazio delle immagini a quello delle classi, dove con il termine classe si intende un'etichetta che fornisce una conoscenza certa della tipologia dell'oggetto nell'immagine. Per decidere cosa rappresenti un'immagine, ad esempio se una persona o una automobile, quello che implicitamente facciamo noi stessi è il riconoscimento di pattern caratteristici per ogni tipo di immagine: se vi sono occhi, naso e bocca allora è probabile sia una persona mentre se sono visibili ruote, fari e la targa allora è probabile si tratti di una automobile. Purtroppo vista la grande variabilità tra i soggetti di una stessa classe risulta essere problematico definire manualmente i *pattern* da riconoscere e vogliamo che sia la rete stessa a trovarli e che lo faccia in maniera gerarchica, cioè partendo da *feature* come bordi o angoli fino a riconoscere gli elementi di un viso e infine la sua composizione.

Le CNN presentano alcune caratteristiche in più rispetto alle reti neurali multistrato, come lo strato di convoluzione e l'operazione di *pooling*, illustrate nella Fig. 2.6, che mostra l'architettura di un'intera rete neurale convoluzionale:

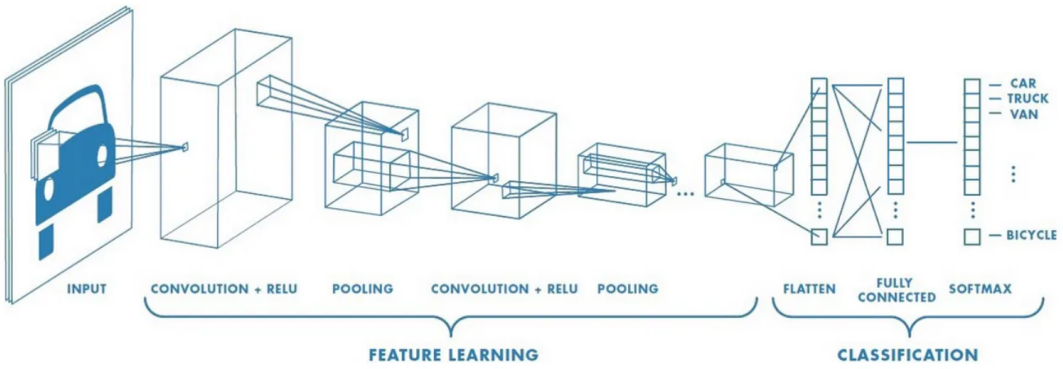


Figura 2.6: *Architettura di una CNN [23].*

Ogni CNN è divisa in due parti: la parte responsabile dell'identificazione dei pattern e la parte di classificazione. La prima parte è composta dalla ripetizione di strati convoluzionali, a cui è sempre applicata una funzione di non linearità, e operazioni di pooling, fino a quando l'immagine è sufficientemente piccola per essere passata alla seconda parte. La seconda parte è costituita da strati densi e non è diversa dalla rete neurale multistrato già incontrata, in cui l'ultimo strato denso mantiene l'output, come i punteggi delle classi.

2.2.1 Strato di convoluzione

Nello strato di convoluzione l'immagine è analizzata un pezzo alla volta in quanto ogni neurone è connesso solo a una regione locale dell'input. Possiamo immaginare il processo come lo scorrere di una finestra immaginaria dove, dopo ogni spostamento, i pixel contenuti nella finestra sono passati a un nuovo neurone dello stesso strato di convoluzione. La finestra è chiamata filtro, e nel caso in Fig. 2.7 cattura a ogni spostamento 16 pixel; il neurone dello strato di convoluzione ha quindi 16 pesi.

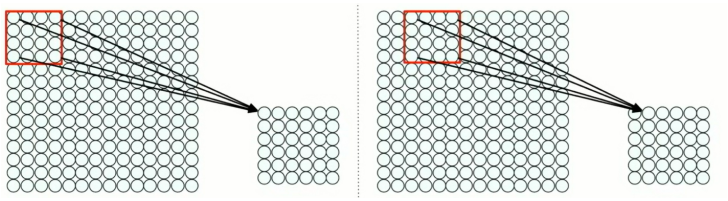


Figura 2.7: *Illustrazione, nel caso di un filtro 4×4, delle connessioni tra l'input e i neuroni di uno strato convoluzionale, di cui è evidente la struttura bidimensionale [22].*

I filtri sono ciò che è usato per rilevare i pattern in un'immagine e funzionano facendo il prodotto di convoluzione tra l'intera immagine I e la matrice del filtro F. L'operazione di convoluzione consiste, per ogni spostamento, in una moltiplicazione elemento per elemento tra la matrice dei pixel passati al neurone e quella dei pesi del filtro. I termini della moltiplicazione sono sommati e maggiore il risultato, maggiore è la corrispondenza tra quella parte di immagine e il filtro. L'operazione di convoluzione produce quindi una matrice di output O detta *feature map* i cui elementi $O_{i,j}$ sono [24]:

$$O_{i,j} = \sum_m \sum_n I_{i+m,\, j+n} \cdot F_{m,n} + b \tag{2.2.1}$$

dove I è l'immagine in input allo strato, F è la matrice del filtro e b il bias da aggiungere prima dell'applicazione della ReLU. Dopo l'operazione di convoluzione l'immagine è ora diventata una feature map (Fig. 2.8), i cui elementi indicano in quali punti dell'immagine il filtro ha ottenuto maggiore o minore corrispondenza.

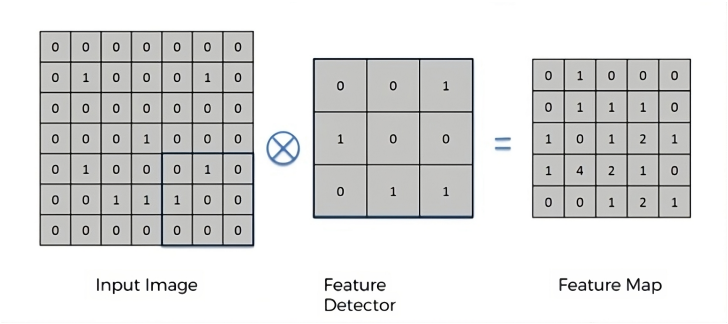


Figura 2.8: *Esempio di feature map ottenuta applicando un filtro 3×3 (feature detector) a un'immagine (input image), con bias nullo.*

I filtri, essendo pesi, sono imparati dalla rete come quelli nelle reti neurali multistrato. In un esempio concreto come il riconoscimento di una X (Fig. 2.9), la rete giungerebbe probabilmente all'uso di almeno 3 filtri: uno per ogni orientamento della linea obliqua e uno per il centro della X dove le due linee si intersecano.

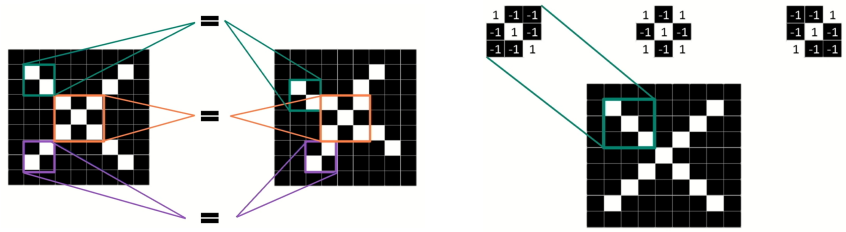


Figura 2.9: *A sinistra sono evidenziate tre pattern chiave per riconoscere una X e a destra un esempio di filtri 3×3 per rilevare tali caratteristiche [22].*

Il numero di neuroni in uno strato di convoluzione non è deciso direttamente ma è calcolato in base a tre iperparametri: la *depth*, la *stride* e il *padding*.

- Depth: corrisponde al numero di filtri N_F di uno strato, cioè quante feature può individuare.
- Stride: indica il passo con cui il filtro scorre, cioè il numero di pixel da saltare tra uno spostamento e l'altro. Essa regola anche la dimensione della feature map che sarà in generale più piccola quanto più è grande la stride.
- Padding: sistema che permette di mettere un bordo di zeri (zero-padding) all'immagine, in modo da influenzare le dimensioni delle feature map o evitare anomalie durante le operazioni di convoluzione.

La larghezza (O_W) e l'altezza (O_H) della feature map in uscita da uno strato di convoluzione può essere calcolata in funzione delle dimensioni dell'input (I_W e I_H), della dimensione

del filtro (F), della stride (S) applicata per il movimento dei filtri, e del quantitativo di padding (P) usato sui bordi [24]:

$$O_W = \frac{(I_W - F + 2P)}{S} + 1 \qquad O_H = \frac{(I_H - F + 2P)}{S} + 1 \tag{2.2.2}$$

2.2.2 Pooling

Quando è rilevata una feature specifica nell'immagine, la sua posizione esatta perde di importanza; ciò che conta è infatti la sua posizione relativa alle altre feature. Per questo motivo sono effettuate operazioni di pooling che riducono la dimensione dell'immagine mantenendone le feature significative. In sostanza, questo è fatto perché riducendo le informazioni spaziali si guadagna in prestazioni di calcolo e perché meno informazioni spaziali significa meno parametri e quindi una minore possibilità di overfitting. L'operazione di pooling è effettuata dopo l'operazione di convoluzione e l'applicazione della ReLU e, diversamente dal prodotto di convoluzione, non utilizza filtri.

Ci sono due tipi di operazioni di pooling:

- *max-pooling* (più usato) che prende il massimo valore dalla finestra analizzata;
- *mean-pooling* che fa la media di tutti gli elementi contenuti nella finestra.

In Fig. 2.10 possiamo osservare un esempio di max-pooling.

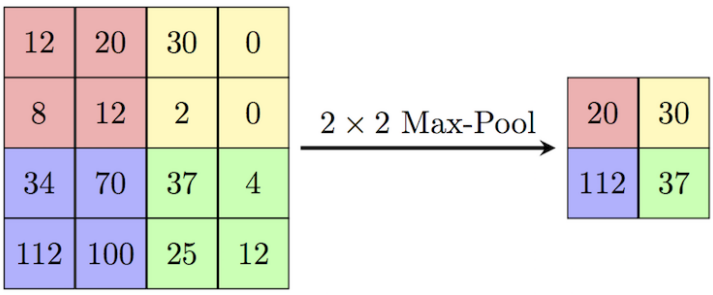


Figura 2.10: *Esempio di un max-pooling 2×2 con stride pari a 2 [25].*

Nell'esempio la larghezza e l'altezza della matrice su cui è fatto il pooling sono riscalate di un fattore 2, e dato che ogni massimo è preso su 4 numeri, il 75% della feature map è scartato, alleggerendo la computazione per gli strati successivi. Inoltre è possibile prima dell'operazione di pooling applicare del padding; in tal caso la dimensione della feature map in uscita è calcolabile con le Eq. 2.2.2.

2.2.3 Strati densi

Gli strati densi servono infine per completare la struttura della rete e la loro funzione principale è quella di eseguire una sorta di raggruppamento delle informazioni prodotte dagli strati di convoluzione per classificare l'immagine. Gli strati di convoluzione e le operazioni di pooling riducono la dimensione dell'input e producono feature map che, dopo essere state appiattite, sono passate tutte insieme agli strati densi che procedono a fornire tanti output quante sono le classi delle immagini da classificare.

Capitolo 3

Implementazione della rete neurale

In questo capitolo è descritto lo sviluppo di una rete neurale convoluzionale (CNN) per la classificazione delle immagini provenienti dal tracciatore di neutroni innovativo RIPTIDE. L'obbiettivo della rete è discriminare tra immagini contenenti tracce luminose provenienti da reazioni di doppio scattering n-p, singolo scattering n-p o background (vedi Fig. 3.1).

3.1 Sviluppo e applicazione della rete neurale

Questa sezione illustra le fasi principali del lavoro, dalla definizione dell'architettura alla fase di addestramento, fino all'analisi dei risultati.

3.1.1 Definizione dell'architettura della rete

La progettazione dell'architettura di una rete neurale dipende da vari fattori, come il numero di ingressi, il numero di uscite, la complessità del compito da svolgere e le dimensioni del dataset. Non esistono regole universali per la scelta di un'architettura, pertanto occorre trovare un equilibrio tra conoscenza teorica e sperimentazione empirica. In questa tesi, sono state testate diverse architetture, sia seguendo ragionamenti intuitivi sia ispirandosi a modelli affermati in letteratura, come AlexNet [26] e VGG [27]. Durante la fase di sperimentazione è stata seguita la semplice regola proposta da Yoshua Bengio [28], che suggerisce di aumentare gradualmente il numero di strati fino a quando l'accuratezza della rete non smetta di migliorare.

La rete è stata implementata in C++17 utilizzando LibTorch, l'API C++ di PyTorch versione 2.4.1. L'architettura della CNN utilizzata in questo lavoro è composta da 5 strati di convoluzione e da 2 strati densi, i cui dettagli sono riassunti in Tab. 3.1. All'uscita di ogni strato della rete, sia di convoluzione che denso, è applicata la funzione di attivazione ReLU, e dopo ogni strato di convoluzione è effettuata una operazione di max-pooling 2x2. Poiché l'ordine in cui vengono eseguite queste due operazioni non influisce sul risultato, si è deciso di seguire lo standard nelle moderne reti neurali, che prevede l'applicazione della funzione di attivazione prima della riduzione delle dimensioni. Inoltre per migliorare la capacità di generalizzazione e ridurre il rischio di overfitting è stato associato a ogni strato, fatta eccezione per quello di output, uno strato di dropout, che spegne casualmente a ogni batch il 10% dei neuroni, disattivandosi durante la fase di testing.

Tabella 3.1: *Modello dell'architettura.*
F=dimensione filtro, S=stride, P=padding.

Strato	**Configurazione**	**Pesi**	**Bias**
conv1	convoluzione(N.filtri=64, F=3, S=1, P=0)	576	64
dropout2d1	Dropout2d(10%)	-	-
MaxPool1	MaxPooling(F=2, S=2, P=0)	-	-
conv2	convoluzione(N.filtri=128, F=3, S=1, P=0)	73 728	128
dropout2d2	Dropout2d(10%)	-	-
MaxPool2	MaxPooling(F=2, S=2, P=1)	-	-
conv3	convoluzione(N.filtri=192, F=3, S=1, P=0)	221 184	192
dropout2d3	Dropout2d(10%)	-	-
MaxPool3	MaxPooling(F=2, S=2, P=0)	-	-
conv4	convoluzione(N.filtri=256, F=3, S=1, P=0)	442 368	256
dropout2d4	Dropout2d(10%)	-	-
MaxPool4	MaxPooling(F=2, S=2, P=1)	-	-
conv5	convoluzione(N.filtri=320, F=3, S=1, P=0)	737 280	320
dropout2d5	Dropout2d(10%)	-	-
MaxPool5	MaxPooling(F=2, S=2, P=1)	-	-
fc1	denso(input=1280, neuroni=500)	640 000	500
dropout	Dropout(10%)	-	-
fc2	denso(input=500, neuroni=3)	1 500	3
Parametri Totali		2 118 099	

La scelta di filtri di dimensione 3×3 per ciascuno strato di convoluzione è dovuta alla presenza nelle immagini da classificare di pattern unicamente di basso livello, quali angoli e linee, maggiormente riconoscibili con filtri di piccole dimensioni [24]. I tentativi di catturare feature di più alto livello nel primo strato di convoluzione, applicando un filtro 7×7 e stride 2 o un filtro 11×11 e stride 4 [26], non hanno portato a miglioramenti dell'accuratezza e sono stati quindi abbandonati.

Per ottimizzare la rete, è stato deciso di aumentare linearmente il numero di filtri con ogni strato di convoluzione, anziché raddoppiarli, in modo da mettere a disposizione del primo strato convoluzionale un maggiore numero di filtri e massimizzare così l'elaborazione prima del primo max-pooling.

L'uso di 5 strati di convoluzione ha permesso di ridurre al minino le dimensioni delle feature map da passare agli strati densi. Partendo da immagini con dimensioni relativamente elevate (100×100×1) sono state ottenute alla fine dei processi di convoluzione e maxpooling delle feature maps di dimensioni 2×2. Presentare delle feature maps di piccole dimensioni all'ingresso del primo strato denso ha un duplice vantaggio: in primo luogo il loro appiattimento non complica il riconoscimento delle informazioni spaziali; in secondo luogo permette l'applicazione di un grande numero di filtri (320) nell'ultimo strato di convoluzione, pur mantenendo piccolo il numero di valori di input (1280) che ogni neurone del primo strato denso deve gestire. Questo ha permesso l'uso di due soli strati densi, uno strato nascosto con 500 neuroni e uno strato di output con 3, riducendo così il rischio di overfitting che cresce all'aumentare del numero di strati densi. Per cercare di accelerare il processo di riduzione delle dimensioni delle feature map sono state provate

anche architetture con meno strati di convoluzione utilizzando una stride di 2, ma un simile approccio si è rivelato inadeguato a causa della perdita durante la convoluzione di troppe informazioni. La riduzione delle feature map attraverso gli strati è esplicitata in Tab. 3.2, dove nella terza colonna sono indicati il numero e le dimensioni delle feature map in uscita da ogni strato.

Tabella 3.2: Riduzione della dimensione delle feature map.

Strato	**Operazione**	**Feature map in uscita**
Input	-	(1, 100, 100)
conv1	Convolution (1 → 64, filtro 3×3, stride 1)	(64, 98, 98)
MaxPool1	MaxPooling (filtro 2×2, stride 2, padding 0)	(64, 49, 49)
conv2	Convolution (64 → 128, filtro 3×3, stride 1)	(128, 47, 47)
MaxPool2	MaxPooling (filtro 2×2, stride 2, padding 1)	(128, 24, 24)
conv3	Convolution (128 → 192, filtro 3×3, stride 1)	(192, 22, 22)
MaxPool3	MaxPooling (filtro 2×2, stride 2, padding 0)	(192, 11, 11)
conv4	Convolution (192 → 256, filtro 3×3, stride 1)	(256, 9, 9)
MaxPool4	MaxPooling (filtro 2×2, stride 2, padding 1)	(256, 5, 5)
conv5	Convolution (256 → 320, filtro 3×3, stride 1)	(320, 3, 3)
MaxPool5	MaxPooling (filtro 2×2, stride 2, padding 1)	(320, 2, 2)
Le feature map in uscita sono appiattite in un vettore 1D		(1280)

3.1.2 Fasi di addestramento e ottimizzazione

L'addestramento della rete è il processo iterativo con cui il modello impara a migliorare le sue previsioni sui dati di addestramento, aggiornando pesi e bias attraverso la minimizzazione di una funzione di loss.

Attraverso le simulazioni descritte in sezione 1.2.5, sono stati generati 10^4 eventi, in modo da ottenere un dataset composto dal 40% di eventi di singolo scattering, 20% di doppio scattering e 40% di background. Sebbene tale distribuzione non rappresenti accuratamente una raccolta dati realistica di RIPTIDE, è stata ritenuta più idonea per ottimizzare il processo di addestramento della rete neurale. Dal momento che l'efficienza nella ricostruzione delle tracce cala significativamente in caso di tracce corte, si è scelto di etichettare come background anche le proiezioni di singoli o doppi scattering in cui sono presenti protoni a energia inferiore ai 5 MeV, in modo che la rete classifichi come singolo o doppio scattering solo immagini che permettano una ricostruzione ottimale della traccia.

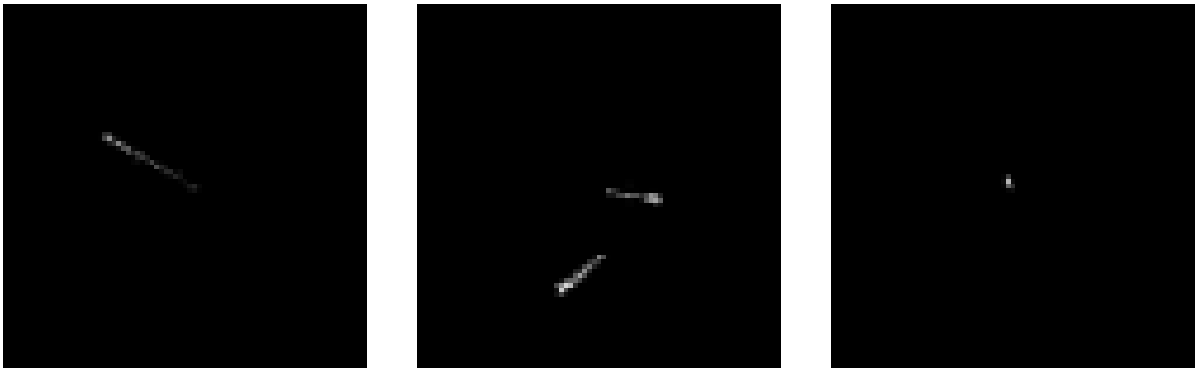


Figura 3.1: Proiezione di un evento di singolo scattering (sinistra), di doppio scattering (centro) e di background (destra).

Il dataset, composto da immagini 100×100×1, è stato fornito alla rete sotto forma di file CSV, in cui ogni immagine corrisponde a una riga contenente i valori dei pixel e dove il primo numero rappresenta l'etichetta. I background sono stati indicati con "0", i singoli scattering con "1" e i doppi scattering con "2". Il dataset è stato quindi mescolato e diviso in un dataset di training (80%) e di testing (20%), che sono utilizzati per inizializzare la classe personalizzata `RIPTIDEDataset`, che eredita dalla classe base `torch::data::datasets` offerta da LibTorch. La classe riceve in argomento il percorso del file CSV e la dimensione dell'immagine e restituisce un tensore di ordine 1 con le etichette e uno di ordine 3 con le immagini. Questi tensori sono poi divisi in batch usando la funzione `torch::data::make_data_loader` che riceve in argomento la dimensione della batch e i tensori con le etichette e le immagini, e provvede anche al mescolamento del dataset a ogni epoca.

Come funzione di loss è stata scelta la Cross-Entropy loss, introdotta in Sez. 2.1.2 mentre per l'aggiornamento dei pesi e dei bias è stato scelto l'ottimizzatore Adam (*Adaptive Moment Estimation*) [29]. Adam adatta individualmente il learning rate per ciascun parametro, diminuendolo per i parametri rispetto a cui la derivata della funzione di loss è grande e aumentandolo per i parametri rispetto a cui la loss varia poco. Questo learning rate adattivo evita di saltare un minimo, e permette di esplorare se ci si trovi in un punto di minimo locale o assoluto, assicurando così una convergenza più stabile e rapida, anche in presenza di gradienti molto piccoli o molto grandi.

Il processo di addestramento si svolge all'interno di un ciclo caratterizzato da due livelli di iterazione: uno per le epoche e uno per le batch. Con il termine epoca si intende un passaggio completo attraverso l'intero dataset di immagini, mentre le batch rappresentano i raggruppamenti di immagini all'interno di tale dataset.

In ogni epoca, per ciascuna batch, sono calcolati il valore medio della funzione di loss e il suo gradiente, e i pesi della rete vengono aggiornati. Il numero di batch determina il numero di aggiornamenti dei pesi effettuati durante l'epoca; per ottimizzare i tempi di addestramento, sono state utilizzate batch composte da 16 immagini. All'inizio di ogni nuova epoca, il dataset è rimescolato, e l'addestramento riprende dai pesi ottenuti alla fine dell'epoca precedente. Durante il ciclo delle epoche, la rete è sia addestrata che testata, scorrendo le batch dei dataset di training e di testing. La principale differenza tra i due cicli di batch risiede nel fatto che, durante il testing, gli strati di dropout sono disabilitati e i parametri del modello non vengono aggiornati. La loss media per ogni

epoca è calcolata sia durante il training (*training loss*) che durante il testing (*validation loss*), e quest'ultima è il parametro usato nell'early stopping per decidere quando arrestare l'addestramento.

Il sistema di early stopping implementato ferma l'addestramento quando la validation loss non migliora per 10 epoche consecutive. Questo valore di *patience* è stato ritenuto un buon compromesso tra il prevenire l'overfitting e il concedere alla rete tempo sufficiente a imparare.

3.1.3 Analisi dei risultati

A ogni epoca sono calcolate per monitorare la performance della rete quattro quantità: training loss, validation loss, *training accuracy* e *validation accuracy*.

La training loss diminuisce a ogni epoca in modo continuo perché viene minimizzata durante l'addestramento, mentre la validation loss è utilizzata per monitorare la capacità di generalizzazione della rete e ha un andamento più irregolare, come si può notare in Fig. 3.2. In particolare, la validation loss diminuisce irregolarmente fino al raggiungimento di un minimo intorno all'epoca 25, dopo la quale ha iniziato lentamente a crescere.

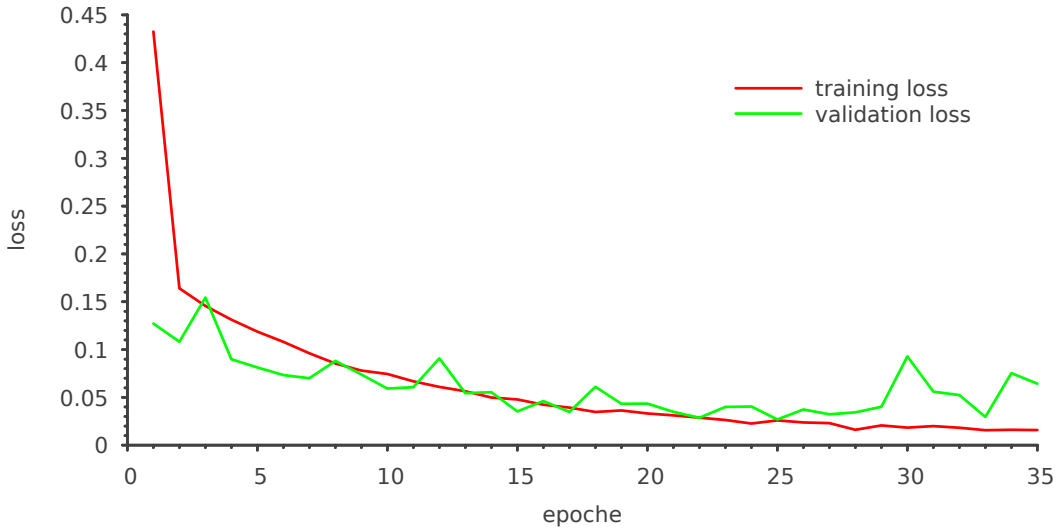


Figura 3.2: *Andamento della training loss (in rosso) e della validation loss (in verde) durante le epoche.*

I termini training accuracy e validation accuracy indicano, rispettivamente, le percentuali di classificazioni corrette effettuate dalla rete sul dataset di training e su quello di testing. Analizzando i loro andamenti durante le epoche, mostrati in Fig. 3.3, si riconosce come la training accuracy tenda asintoticamente al 100%, mentre la validation accuracy si stabilizzi intorno al 99% per poi iniziare a calare dopo circa 25 epoche.

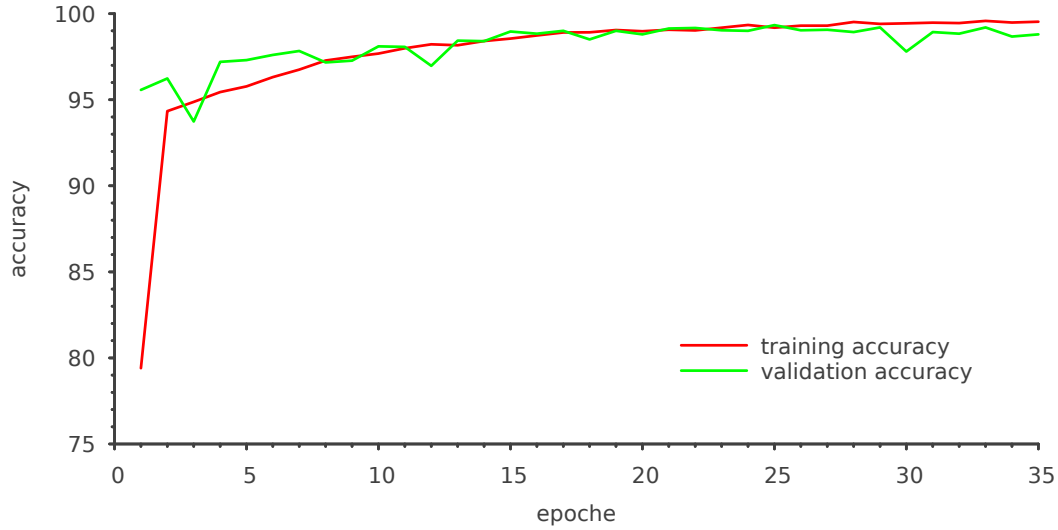


Figura 3.3: *Andamento della training accuracy (in rosso) e della validation accuracy (in verde) durante le epoche.*

Per decidere quando fermare l'addestramento si guarda alla validation loss e alla validation accuracy, e non alla training loss e alla training accuracy dal momento che migliorano asintoticamente, anche se non è semplice determinare il punto esatto nel quale queste grandezze iniziano a peggiorare a causa del loro andamento irregolare. Attraverso l'early stopping l'addestramento è stato interrotto all'epoca 35, 10 epoche dopo che i valori di validation loss e validation accuracy non sono stati migliorati. In particolare il maggior valore ottenuto per la validation accuracy è stato di 99.3%.

In aggiunta a loss e accuratezza, è possibile monitorare la performance della rete anche attraverso un grafico *Receiver Operating Characteristic* (ROC) [30]. Questo grafico confronta ciascuna classe contro tutte le altre e permette di capire quali casi la rete faccia più fatica a classificare. Nel grafico in Fig. 3.4, ogni curva rappresenta la performance del modello per una specifica classe. Ogni punto sulla curva rappresenta un compromesso tra la sensibilità, o tasso di veri positivi (TPR), e il tasso di falsi positivi (FPR). La quantità scelta per indicare la capacità del modello di discriminare tra le classi è l'area sotto la curva (AUC). Un valore AUC di 1 indica un modello perfetto, mentre un AUC di 0.5 indica che il modello non ha alcuna capacità discriminativa (equivalente a un modello casuale).

Nel grafico sono mostrate tre curve, ciascuna corrispondente a una delle tre classi di immagini: background, singolo scattering e doppio scattering. È evidente che il modello mostra un'ottima capacità di classificare correttamente le immagini appartenenti alla classe background, mentre manifesta maggiori difficoltà nell'identificare correttamente le immagini delle classi singolo e doppio scattering. Sebbene i valori di AUC siano molto vicini a 1 per tutte le classi, le differenze, seppur minime, indicano che gli errori di classificazione della rete sono principalmente dovuti alla confusione tra le immagini di singolo scattering e quelle di doppio scattering, che vengono talvolta erroneamente scambiate tra loro. Questo comportamento suggerisce che il modello potrebbe trarre vantaggio da un aumento del numero di immagini di queste due classi nel dataset.

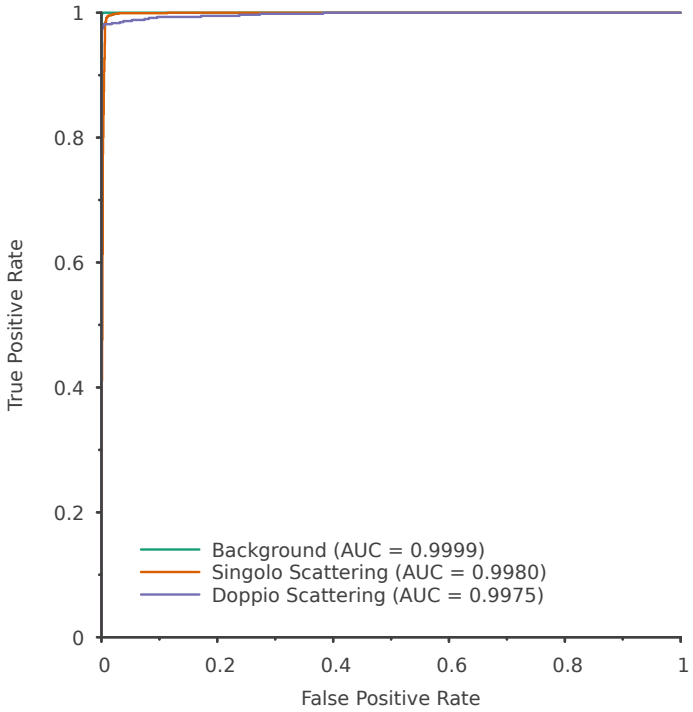


Figura 3.4: *Grafico ROC multiclasse che mostra le performance del modello per le tre classi: background (in verde), singolo scattering (in arancione) e doppio scattering (in violetto). Nella legenda è riportato il valore di AUC per ogni classe.*

Il tasso di veri positivi rappresenta la proporzione di osservazioni positive correttamente classificate come tali ed è calcolato come:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{3.1.1}$$

dove TP è il numero di veri positivi (le osservazioni positive classificate correttamente come positive) e FN è il numero di falsi negativi (le osservazioni positive erroneamente classificate come negative). Il tasso di falsi positivi rappresenta invece la proporzione di osservazioni negative che vengono erroneamente classificate come positive ed è calcolato come:

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} \tag{3.1.2}$$

dove FP è il numero di falsi positivi (le osservazioni negative classificate come positive) e TN è il numero di veri negativi (le osservazioni negative classificate correttamente come negative).

Conclusioni e sviluppi futuri

Il principale obbiettivo di questa tesi è dimostrare come le reti neurali convoluzionali (CNN) possano essere utilizzate con successo per la classificazione delle immagini generate dal tracciatore di neutroni RIPTIDE. Si è quindi proceduto a individuare l'architettura e i valori degli iperparametri che massimizzassero l'accuratezza della rete nella distinzione tra eventi di singolo scattering, doppio scattering e background.

I risultati hanno mostrato da parte della rete un'alta capacità di discriminazione tra le classi presentatele nel dataset, raggiungendo percentuali di accuratezza superiori al 99%. Si è inoltre visto, attraverso l'utilizzo di un grafico ROC, che la rete non ha avuto difficoltà a classificare le immagini appartenenti alla classe di background, mentre ha talvolta confuso i casi di singolo e doppio scattering.

La rete è stata unicamente addestrata e testata su dati generati con simulazioni Monte Carlo e, non essendo RIPTIDE ancora operativo, non è certo che le immagini utilizzate siano un campione rappresentativo di una reale presa dati, che potrebbe rilevare nuove sorgenti di background non considerate nella simulazione o un maggiore rumore. Solo l'addestramento e il testing della rete neurale su tali dati potrà quindi confermare la correttezza delle scelte fatte o indicarne di più adatte.

La rete sviluppata in questa tesi potrà essere applicata, non appena RIPTIDE entrerà in funzione, per selezionare tra le immagini acquisite, solo quelle relative agli eventi di nostro interesse. Questo permetterebbe di automatizzare il processo di analisi dati, migliorando la precisione e la scalabilità del sistema nel rilevamento dei neutroni. I risultati ottenuti aprono nuove prospettive per l'estensione del modello a dataset più complessi e per l'uso di reti neurali più profonde per migliorare ulteriormente la classificazione.

Bibliografia

[1] C. Pisanti et al. Riptide: a proton-recoil track imaging detector for fast neutrons. *Journal of Instrumentation*, 19(02):C02074, Feb. 2024.

[2] L. I. Dorman. *Solar Neutrons and Related Phenomena*. Springer, 2010.

[3] D. M. Hassler et al. Mars' surface radiation environmentmeasured with the mars sciencelaboratory's curiosity rover. *Science*, page 1244797, 2014.

[4] John W. Norbury. Nuclear physics and space radiation. In Sean Freeman et al., editors, *J. Phys. Conf. Ser.*, volume 381, page 012117, 2012.

[5] F. Natale. Adroterapia: la nuova frontiera della radioterapia, 2016.

[6] C. Guerrero et al. Performance of the neutron time-of-flight facility n tof at cern. *The European Physical Journal A*, 49(2):27, 2013.

[7] C. Massimi, A. Musumarra, and N. Patronis. Measurement of the neutron-neutron scattering length at the cern n tof facility. Technical report, CERN, 2020.

[8] M. Visvalingam and J. D. Tandy. The neutron method for measuring soil moisture content- a review. *Journal of Soil Science*, 23:499–511, 1972.

[9] J. M. Ryan et al. A scintillating plastic fiber tracking detector for neutron and proton imaging and spectroscopy. In *Nuclear Science Symposium*, 1999.

[10] S. M. Valle et al. The mondo project: A secondary neutron tracker detector for particle therapy. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 845:556–559, 2017.

[11] Chenyao Han et al. A background suppression detector array for fast neutron measurement in space science study. *Measurement*, 230:114479, 2024.

[12] W.R. Leo. *Techniques for Nuclear and Particle Physics Experiments: A How-To Approach*. Springer, 1981.

[13] Donald E. Groom and Spencer R. Klein. Passage of particles through matter. *The European Physical Journal C - Particles and Fields*, 15:163–173, 2000.

[14] J. Hu, J. Liu, Z. Zhang, et al. Recoil-proton track imaging as a new way for neutron spectrometry measurements. *Scientific Reports*, 8(1):13363, Sep. 2018.

[15] S. Agostinelli et al. GEANT4–a simulation toolkit. *Nucl. Instrum. Meth. A*, 506:250–303, 2003.

[16] David Wechsler. *The Measurement of Adult Intelligence*. Williams & Wilkins, Baltimore, 3rd edition, 1944.

[17] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408, 1958.

[18] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[19] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[20] A. Amini. Mit introduction to deep learning 6.s191: Lecture 1. Online, 2023. Foundations of Deep Learning.

[21] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.

[22] A. Amini. Mit introduction to deep learning 6.s191: Lecture 3. Online, 2023. Convolutional Neural Networks for Computer Vision.

[23] S. Patel and J. Pingel. Introduction to deep learning: What are convolutional neural networks?, 2017.

[24] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *ArXiv*, abs/1311.2901, 2013.

[25] Sarvachan Verma and Manoj Kumar. Automatic classification of melanoma using grab-cut segmentation & convolutional neural network. *SN Comput. Sci.*, 5:591, 2024.

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60:84 – 90, 2012.

[27] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[28] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. In Grégoire Montavon, Genevieve B. Orr, and Klaus-Robert Müller, editors, *Neural Networks: Tricks of the Trade*, pages 437–478. Springer Berlin Heidelberg, 2012.

[29] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[30] Mark R. J. Junge and Joseph R. Dettori. Roc solid: Receiver operator characteristic (roc) curves as a foundation for better diagnostic tests. *Global Spine Journal*, 8:424 – 429, 2018.