Università degli Studi di Padova

Dipartimento di Fisica e Astronomia "Galileo Galilei"

Corso di Laurea in Fisica

Tesi di Laurea

ALGORITMI DI NEURAL NETWORKS PER LA DISCRIMINAZIONE DELLA RADIAZIONE GAMMA/NEUTRONI IN UN NUOVO RIVELATORE BASATO SU SCINTILLATORI PLASTICI PER APPLICAZIONI DI FISICA AMBIENTALE

Candidato: LUCA MORSELLI

Relatore: PROF. MARCELLO LUNARDON
Correlatore: PROF. SAMIR SUWEIS, ALBERTO TESTOLIN, LUCA STEVANATO

Anno accademico 2016-2017

Ai miei nonni.

Indice

Capitolo 1

Introduzione

La misura ambientale dei neutroni ha registrato un crescente interesse per quanto riguarda la determinazione dell'umidità in contesti ambientali ed agronomici. Le sonde più utilizzate per questo tipo di misura sono basate su contatori proporzionali ad ^{3}He. A causa dei costi elevati di queste sonde si stanno studiando nuovi tipi di rivelatori ottenuti attraverso l'assemblaggio di diversi scintillatori coi quali è possibile costruire un rivelatore sensibile ai neutroni lenti, veloci ed ai raggi gamma. La discriminazione viene effettuata attraverso un'analisi della forma dell'impulso utilizzando algoritmi che confrontano l'integrale totale del segnale e l'integrale parziale. Questi algoritmi funzionano bene ad energie medio-alte, mentre a bassa energia sono via via meno efficienti. L'obiettivo della tesi è quello di implementare questa discriminazione attraverso algoritmi di reti neurali e di confrontarla con il metodo di analisi della forma di impulso, con il fine di migliorare la discriminazione nella regione di bassa energia.
L'elaborato inizia col presentare le reti neurali artificiali con particolare attenzione per quanto riguarda il paradigma di apprendimento supervisionato (Capitolo 2) per poi proseguire nei capitoli successivi con l'illustrare la misura sperimentale dei raggi gamma e dei neutroni e le sue problematiche (Capitolo 3) e i risultati di discriminazione attraverso l'utilizzo della rete neurale (Capitolo 4).

Capitolo 2

Reti Neurali Artificiali

Le reti neurali sono particolari tipi di algoritmi costruiti in modo da imitare le reti neurali biologiche. Queste reti sono formate da un insieme di connessioni tra unità di base chiamate neuroni artificiali. Una definizione più formale si può trovare in [Kriesel, 2009]:

Definizione 1 (Rete Neurale)**.** : Una rete neurale è una tripla ordinata (N, V, w) dove N è l'insieme dei neuroni artificiali e $V = \{(i, j) : i, j \in \mathbb{N}\}$ insieme i cui elementi vengono chiamati **connessioni**. La funzione $w : V \longrightarrow \mathbb{R}$ definisce i **pesi** ($w(i, j) = w_{ij}$ è il peso relativo alla connessione tra il neurone i-esimo e quello j-esimo).

2.1 Apprendimento

La caratteristica che ha reso popolare questo tipo di strutture è la loro capacità di apprendere. Attraverso l'apprendimento la rete cerca di trovare, dato un determinato compito (classificare dati, riconoscimento di immagini, ...), una certa funzione f^* che permetta di ottenere una mappa input-output (che chiameremo funzione di trasferimento) che realizza il compito assegnato. Per fare questo abbiamo bisogno di qualcosa che ci quantifichi la qualità della funzione trovata, si definisce allora una **funzione di costo**:

$$C : F \longrightarrow \mathbb{R} \tag{2.1}$$

tale che per la soluzione ottimale f^* valga:

$$C(f^*) < C(f) \quad \forall f \in F \tag{2.2}$$

dove F è una classe di funzioni assegnata al problema. L'apprendimento deriva dalla capacità di questi algoritmi di estrarre regolarità statistiche dai dati. Il paradigma di apprendimento che utilizzeremo è quello supervisionato.

2.1.1 Apprendimento supervisionato

Questo tipo di apprendimento si basa sul fornire alla rete un insieme di esempi durante l'allenamento, dove un esempio è costituito da una coppia $(\mathbf{x}, y)$ con **x** vettore di ingresso ed y etichetta associata. La rete elabora il vettore di ingresso e genera una $\hat{y}$, infine viene calcolato il valore della funzione di costo utilizzando i due valori $C(y, \hat{y})$. La rete dovrà quindi modificare i suoi parametri attraverso un determinato algoritmo di apprendimento cosi da minimizzare questo errore. In figura (2.1) viene rappresentato schematicamente il processo che vogliamo implementare: i segnali in ingresso verranno digitalizzati e forniti alla rete dopo un opportuno preprocessamento (Capitolo3) dopodiché la rete predirà una classe che verrà confrontata con l'etichetta associata al segnale.

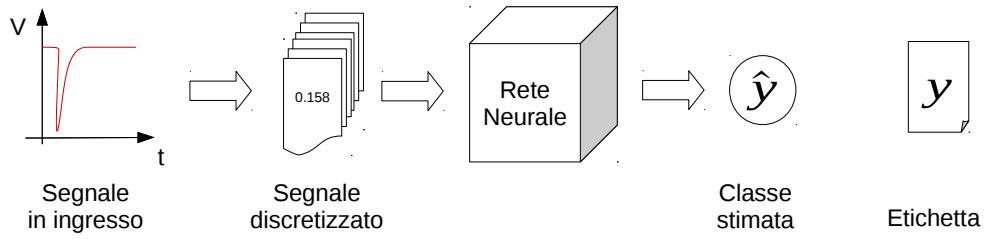


FIGURA 2.1: Paradigma supervisionato

Quello che si assume quando si trattano problemi supervisionati è che i dati in ingresso appartengano ad un numero discreto di classi prestabilite[1]. Per addestrare un classificatore supervisionato è necessario fornirgli un grande numero di esempi etichettati cosi da permettere il riconoscimento di regolarità statistiche all'interno dei dati.

2.2 Struttura delle reti neurali

Esistono diverse tipologie di reti neurali, quella che utilizzeremo è una rete di tipo **Feed-Forward** in cui le connessioni permesse sono solamente quelle da neuroni in uno strato l a quelli in uno strato $l+1$. La rete che andremo ad implementare è chiamata *MultiLayer Perceptron* (fig. 2.2). In questa rete ogni neurone in uno strato l è connesso a tutti i neuroni nello strato $l+1$, reti con questa proprietà si dicono **completamente connesse**. I neuroni dello strato nascosto hanno la funzione di identificare i

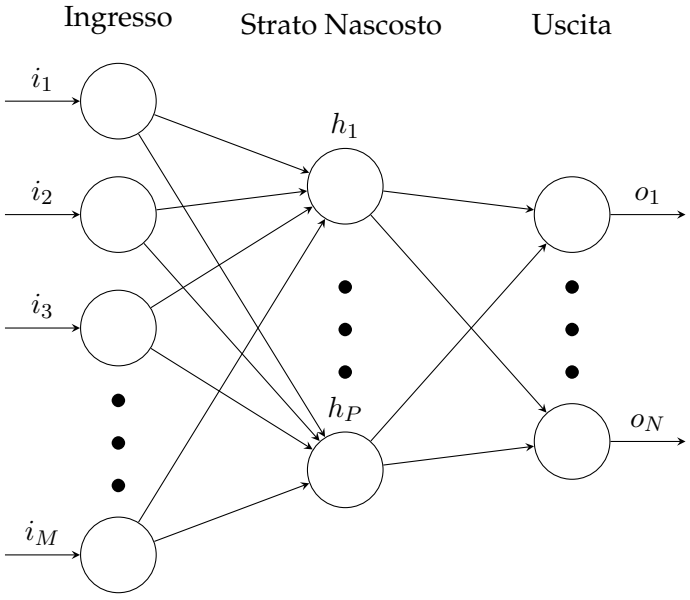
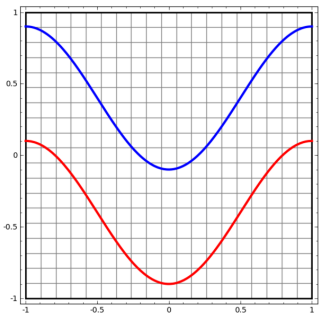


FIGURA 2.2: Esempio di Multilayer Perceptron con un solo strato nascosto

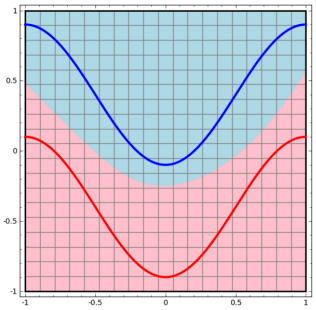
parametri che ben caratterizzano i dati di allenamento, questo viene fatto attraverso

[1]Questa assunzione può essere molto discutibile in alcuni casi

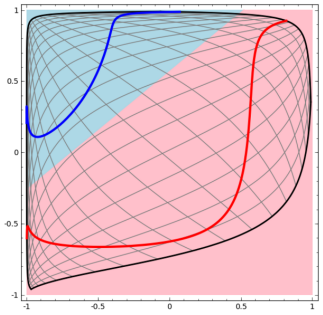
una trasformazione non lineare dei dati in ingresso.In questo nuovo spazio, chiamato *feature space*, le classi che ci interessa distinguere possono essere facilmente separabili rispetto allo spazio originario degli ingressi. In figura(2.3) è riportato l'esempio di una rete a due ingressi in cui si vogliono classificare punti che appartengono a due diverse curve in un piano. Si nota come la trasformazione, effettuata dai neuroni dello strato nascosto, renda linearmente separabili le due classi che vengono passate allo strato finale.



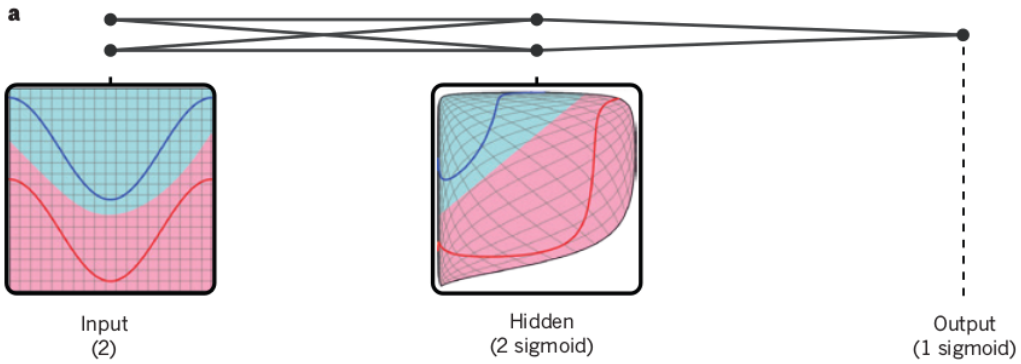
(A) Dati in ingresso.



(B) Classificazione spazio input



(C) *Feature Space*.



(D) Esempio di rete a due ingressi

FIGURA 2.3: Trasformazione degli ingressi operata dallo strato nascosto. Fonte: colah.github.io/ (prima riga), LeCun, Bengio, and Hinton, 2015 (seconda riga). I dati in ingresso (figura A) sono formati da punti appartenenti a due classi : punti blu e rossi. I colori nelle figure (B) e (C) delimitano gli iperpiani di separazione trovati dalla rete.

2.2.1 Il neurone artificiale

Il neurone artificiale è l'unità di base delle reti neurali (fig. 2.4). Quest'ultimo è composto da un numero N di ingressi che ricevono un segnale dai neuroni dello strato precedente o direttamente i dati da processare(nel caso dello strato iniziale).

Ogni ramo di ingresso è connesso al corpo del neurone attraverso una collegamento che dipende da un parametro w_{ij}^{l} chiamato *peso* che quantifica l'importanza dello specifico ramo di ingresso in relazione all'output (vedere figura 2.5). I segnali ven-

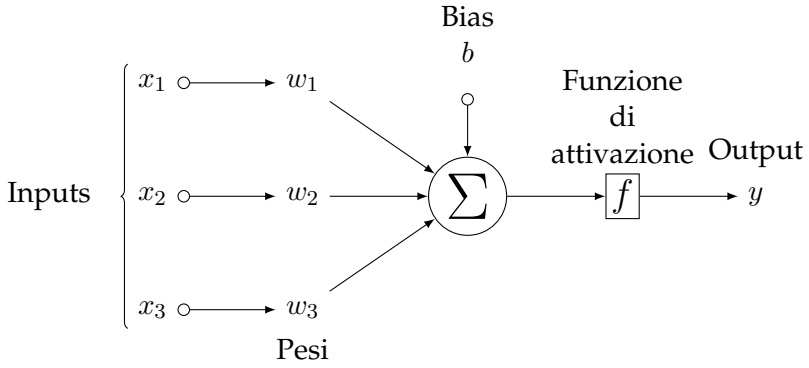


FIGURA 2.4: Il neurone artificiale

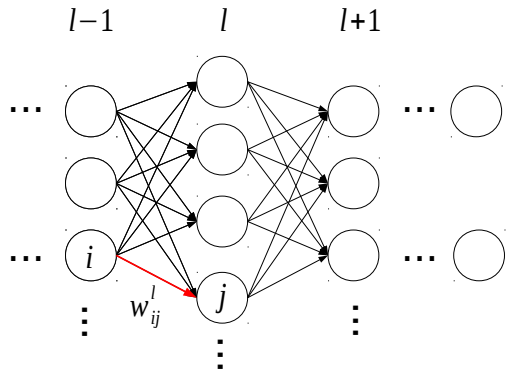


FIGURA 2.5: Notazione usata per indicare i pesi tra due neuroni

gono poi sommati emulando il funzionamento del *soma* nel neurone biologico:

$$z_j^l = \sum_{i \in l-1} w_{ij}^l x_i + b_j^l \tag{2.3}$$

che espresso in forma vettoriale diventa:

$$\mathbf{z}^l = (W^l)^T \mathbf{X} + \mathbf{b}^l \tag{2.4}$$

dove W^l è la matrice dei pesi tra lo strato $l-1$ e l:

$$W = \begin{bmatrix} w_{11}^l & w_{12}^l & w_{13}^l & \dots & w_{1n}^l \\ w_{21}^l & w_{22}^l & w_{23}^l & \dots & w_{2n}^l \\ \dots & \dots & \dots & \dots & \dots \\ w_{N1}^l & w_{N2}^l & w_{N3}^l & \dots & w_{NM}^l \end{bmatrix}$$

mentre **X** rappresenta il vettore dei dati di ingresso (nel caso stessimo considerando un strato di input) o il vettore degli output dei neuroni presenti nello strato $l-1$. Nella (2.4) si applica anche una traslazione dovuta ai bias dei singoli neuroni[2]. La somma di rete viene poi passata alla funzione di attivazione che rappresenta l'analogo biologico dello stato di attivazione del neurone. L'output del neurone è

[2]I bias possono essere visti come ingressi con collegamento $w = 1$

dunque rappresentato da:

$$a_j^l = f(z_j^l) \tag{2.5}$$

che possiamo scrivere come:

$$\mathbf{a}^l = F(\mathbf{z}^l) \tag{2.6}$$

dove:

$$F : \mathbb{R}^M \longrightarrow \mathbb{R}^M \tag{2.7}$$
$$F : z_j^l \mapsto f(z_j^l) = a_j^l \quad \forall j \in M$$

con M numero di neuroni nello strato l.

Le funzioni di attivazione

A seconda della funzione di attivazione scelta possiamo distinguere diversi tipi di neurone. Le funzioni di attivazione più utilizzate sono (2.6):

- Funzione di Fermi (o funzione logistica)
- Tangente Iperbolica

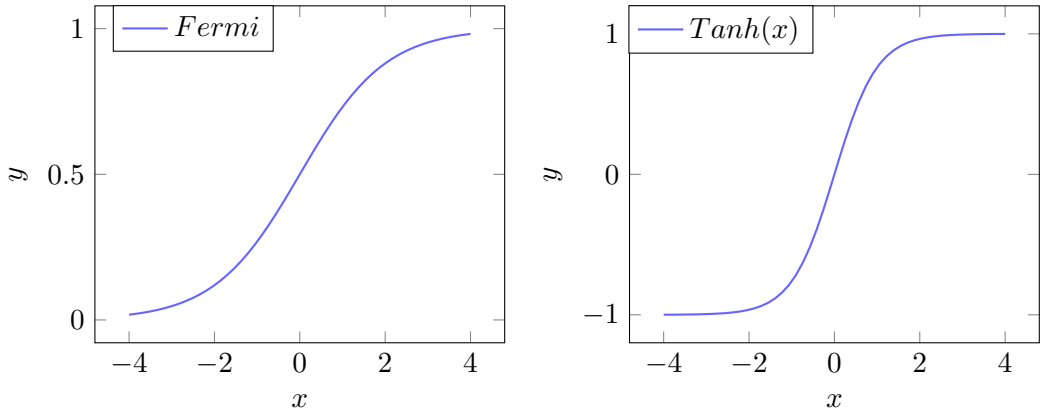


FIGURA 2.6: Funzioni di attivazione comuni

Una proprietà importante che hanno queste funzioni è quella di essere differenziabili: essenziale per poter applicare l'apprendimento attraverso la backpropagation. La tangente iperbolica è preferibile rispetto alla funzione logistica in quanto dispari, quindi il neurone produrrà in media un segnale vicino a 0 rendendo più semplice l'apprendimento.

2.3 Discesa del gradiente

Per automatizzare l'apprendimento abbiamo bisogno di un metodo che ci permetta di trovare il minimo di una funzione del tipo

$$F : \mathbb{R}^m \longrightarrow \mathbb{R}$$

dove nel nostro caso F sarà la funzione di costo $C(w, b)$. Il metodo della discesa del gradiente è uno degli approcci più utilizzati per l'apprendimento automatico, si basa sul fatto che per una funzione $F(\mathbf{x})$ definita e differenziabile in un intorno

di $\mathbf{x}_1$ la direzione di massima decrescita è data dalla direzione opposta al gradiente calcolato nel punto stesso. Se sviluppiamo in serie di Taylor al prim'ordine la F:

$$F(\mathbf{x}) = F(\mathbf{x}_1) + \nabla F(\mathbf{x}_1)(\mathbf{x} - \mathbf{x}_1) + O(||\mathbf{x} - \mathbf{x}_1||^2) \tag{2.8}$$

Pensiamo ora di spostarci da $\mathbf{x}_1$ di una quantità h in una certa direzione definita da un versore $\mathbf{u}$, quello che cerchiamo è la direzione che minimizzi $F(\mathbf{x}_1 + h\mathbf{u})$. Usando (2.8):

$$F(\mathbf{x}_1 + h\mathbf{u}) - F(\mathbf{x}_1) = h\nabla F(\mathbf{x}_1) \cdot \mathbf{u} + h^2 O(1) \tag{2.9}$$

ignorando il termine di second'ordine otteniamo:

$$F(\mathbf{x}_1 + h\mathbf{u}) - F(\mathbf{x}_1) = h\nabla F(\mathbf{x}_1) \cdot \mathbf{u} \tag{2.10}$$

per fare in modo che la quantità a primo membro decresca più rapidamente dobbiamo minimizzare $h\nabla F(\mathbf{x}_1) \cdot \mathbf{u}$, si vede che il versore che minimizza questa quantità è:

$$\mathbf{u} = -\frac{\nabla F}{||\nabla F||}$$

quindi il punto di minimo di F si ottiene iterando la seguente regola:

$$\mathbf{x}^{i+1} = \mathbf{x}^i - \eta \nabla F(\mathbf{x}^i) \tag{2.11}$$

in componenti:

$$x^{i+1} = x^i - \eta \frac{\partial F}{\partial x}(x^i) \tag{2.12}$$

dove η prende il nome di **learning rate**[3]. L'iterazione della (2.11) si può fermare nel caso in cui $||\mathbf{x} - \mathbf{x}_1|| < \epsilon$ con ϵ valore di soglia prefissato.

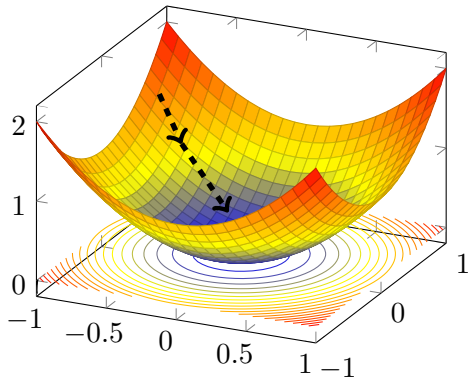


FIGURA 2.7: Due iterazioni della discesa del gradiente nel caso di $F(x, y) = x^2 + y^2$

[3]Quando si parla di metodi numerici η indica il cosiddetto *step-size* ovvero di quanto ci muoviamo lungo la direzione opposta al gradiente ad ogni iterazione. Quando si parla di reti neurali questo coefficente cambia nome in quanto ci indica quanto veloce è l'apprendimento.

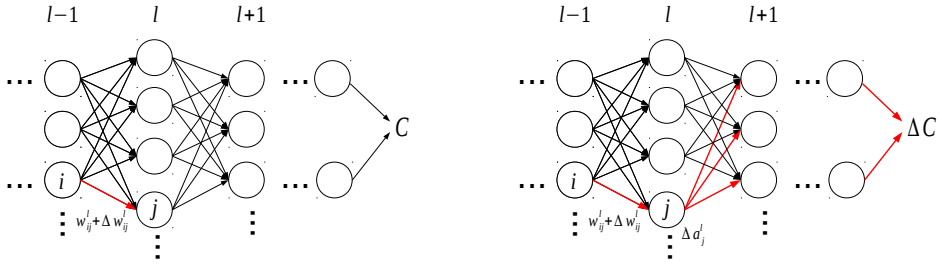
2.4 Backpropagation

Questo algoritmo di apprendimento è riportato in un articolo del 1988 (Rumelhart, Hinton, and Williams, 1988), è uno degli algoritmi più utilizzati per allenare reti neurali in quanto risulta essere computazionalmente efficiente rimanendo concettualmente semplice. Questo algoritmo ci permette di calcolare il gradiente della funzione di costo cosi da permetterci di aggiornare i pesi secondo la (2.11) che nel caso di una rete neurale del tipo considerato sarà nella forma:

$$w_{ij}^l(t+1) = w_{ij}^l(t) - \eta \frac{\partial C}{\partial w_{ij}^l}(t) \tag{2.13}$$

$$b_j^l(t+1) = b_j^l - \eta \frac{\partial C}{\partial b_j^l}(t) \tag{2.14}$$

Pensiamo di applicare una piccola variazione ad un certo peso in uno strato della rete $w_{ij}^l \mapsto w_{ij}^l + \Delta w_{ij}^l$ (fig. 2.8 (A)). Questa variazione porta conseguentemente ad una variazione nella funzione di attivazione Δa_j^l che a sua volta si propaga lungo la rete (fig. 2.8 (B)). La variazione di C sarà allora:



(A) Piccola variazione in un peso nella rete (B) Propagazione della variazione lungo la rete

FIGURA 2.8: Variazione dei pesi in una rete generica.

$$\Delta C \sim \frac{\partial C}{\partial w_{ij}^l} \Delta w_{ij}^l \tag{2.15}$$

Per calcolare $\frac{\partial C}{\partial w_{ij}^l}$ dobbiamo allora ricostruire come una piccola variazione di w_{ij}^l porta ad una variazione ΔC. E' facile osservare che:

$$\Delta a_j^l \sim \frac{\partial a_j^l}{\partial w_{ij}^l} \Delta w_{ij}^l \tag{2.16}$$

a sua volta per a_j^l:

$$\Delta a_q^{l+1} \sim \frac{\partial a_q^{l+1}}{\partial a_j^l} \Delta a_j^l = \frac{\partial a_q^{l+1}}{\partial a_j^l} \frac{\partial a_j^l}{\partial w_{ij}^l} \Delta w_{ij}^l \tag{2.17}$$

questo si ripete fino che si arriva al livello L:

$$\Delta C \sim \frac{\partial C}{\partial a_m^L}\frac{\partial a_m^L}{\partial a_n^{L-1}}\frac{\partial a_n^{L-1}}{\partial a_p^{L-2}}\cdots\frac{\partial a_q^{l+1}}{\partial a_j^l}\frac{\partial a_j^l}{\partial w_{ij}^l}\Delta w_{ij}^l \tag{2.18}$$

Fin ora abbiamo considerato un percorso che passa attraverso $a_j^l, a_q^{l+1}, \ldots, a_n^{l+1}, a_m^L$ come in figura (2.9)

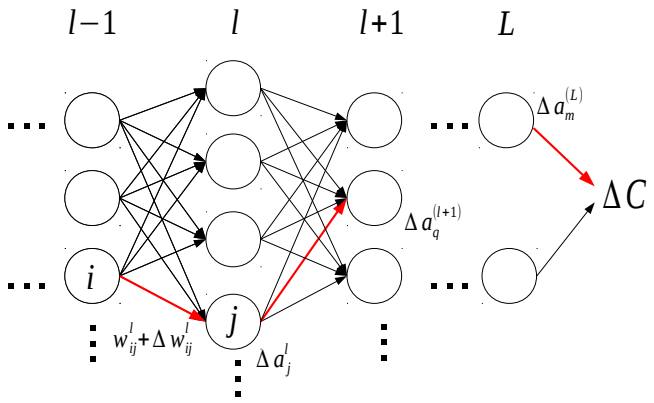


FIGURA 2.9: Variazione lungo un determinato percorso attraverso la rete.

Sommando su tutti i possibili percorsi otteniamo:

$$\Delta C = \sum_{mnp\ldots q}\frac{\partial C}{\partial a_m^L}\frac{\partial a_m^L}{\partial a_n^{L-1}}\frac{\partial a_n^{L-1}}{\partial a_p^{L-2}}\cdots\frac{\partial a_q^{l+1}}{\partial a_j^l}\frac{\partial a_j^l}{\partial w_{ij}^l}\Delta w_{ij}^l \tag{2.19}$$

Facendo un'identificazione con la (2.15) si ottiene:

$$\frac{\partial C}{\partial w_{ij}^l} = \sum_{mnp\ldots q}\frac{\partial C}{\partial a_m^L}\frac{\partial a_m^L}{\partial a_n^{L-1}}\frac{\partial a_n^{L-1}}{\partial a_p^{L-2}}\cdots\frac{\partial a_q^{l+1}}{\partial a_j^l}\frac{\partial a_j^l}{\partial w_{ij}^l} \tag{2.20}$$

che notiamo essere l'applicazione della regola della catena. Per ottenere l'algoritmo in una forma iterativa calcoliamo una generica delle derivate:

$$\frac{\partial a_j^l}{\partial a_q^{l-1}} = \frac{\partial}{\partial a_q^{l-1}} f\left(\sum_{i\in l-1} w_{ij}^l a_i^{l-1}\right) = f'(z_j^l) w_{qj}^l \tag{2.21}$$

e definiamo:

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} \tag{2.22}$$

questa può essere riscritta come:

$$\delta_j^l = \sum_k \frac{\partial C}{\partial z_k^{l+1}}\frac{\partial z_k^{l+1}}{\partial z_j^l} \tag{2.23}$$

dove esplicitando la seconda derivata:

$$\frac{\partial z_k^{l+1}}{\partial z_j^l} = \sum_i w_{ik}^{l+1} \frac{\partial a_i^l}{\partial z_k^l}(z_i^l) = w_{ik}^{l+1} f'(z_k^l) \tag{2.24}$$

dunque:

$$\delta_j^l = \sum_k \frac{\partial C}{\partial z_k^{l+1}} w_{ik}^{l+1} f'(z_k^l) = \sum_k \delta_k^{l+1} w_{ik}^{l+1} f'(z_k^l) \tag{2.25}$$

scrivendo esplicitamente le derivate nella (2.20) secondo la (2.21) si ottiene:

$$\frac{\partial C}{\partial w_{ij}^l} = \sum_{mnp\cdots q} \frac{\partial C}{\partial a_m^L} f'(z_m^L) w_{nm}^L f'(z_n^{L-1}) w_{pn}^{L-1} \cdots f'(z_q^{l+1}) w_{jq}^{l+1} \frac{\partial a_j^l}{\partial w_{ij}^l} = \tag{2.26}$$

$$= \sum_{np\cdots q} \left(\underbrace{\sum_m \delta_m^L f'(z_m^L) w_{nm}^L}_{\delta_n^{L-1}} \right) f'(z_n^{L-1}) w_{pn}^{L-1} \cdots f'(z_q^{l+1}) w_{jq}^{l+1} \frac{\partial a_j^l}{\partial w_{ij}^l} = \cdots = \delta_j^l a_i^{l-1}$$

ed analogamente per i bias:

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \tag{2.27}$$

Se quindi conosciamo δ^L è possibile "propagare all'indietro" l'errore attraverso questo algoritmo. All'interno della nostra rete avremo allora due tipi di "segnali" (fig. 2.10):

- Segnale di andata prodotto dagli esempi (Nero)
- Segnale di errore (Blu)

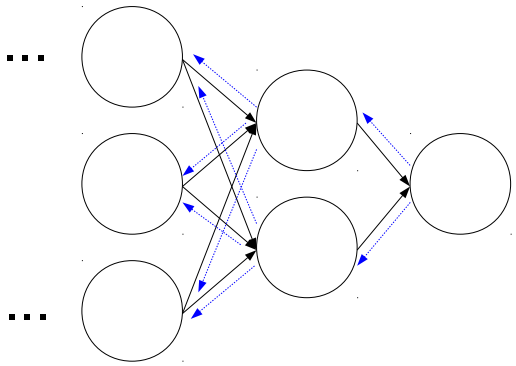


FIGURA 2.10: I due diversi tipi di segnali nella rete

2.5 Linee guida per implementare la backpropagation

L'algoritmo della backpropagation è uno dei più utilizzati per l'apprendimento automatico, tuttavia allenare una rete neurale attraverso questo algoritmo richiede la configurazione di un certo numero di iperparametri la cui scelta non è sempre chiara, non esiste infatti una ricetta da seguire poiché spesso quest'ultimi sono fortemente dipendenti dal tipo di problema e di dati. Esistono però alcune utili linee guida che si possono seguire per velocizzare l'apprendimento [LeCun et al., 2012].

2.5.1 Tipi di apprendimento

Consideriamo un insieme di allenamento contenente un numero N di esempi:

$$\mathbf{T} = \{\mathbf{x}_n, \mathbf{y}_n\}_{n=1...N}$$

per ogni esempio fornito i neuroni nello strato di output (che possiamo indicare come i neuroni appartenenti ad un insieme O) produrranno un "segnale" $y_i(n)$ da cui poi si ottiene un segnale di errore quantificato dalla funzione di costo:

$$e_i(n) = C(d_i(n), y_i(n)) \tag{2.28}$$

sommando su tutti i neuroni nello strato di uscita si ottiene l'errore totale:

$$e(n) = \sum_{i \in O} C(d_i(n), y_i(n)) \tag{2.29}$$

ad esempio nel caso dello scarto quadratico medio come funzione di costo la (2.28) diventa:

$$e(n) = \frac{1}{2} \sum_{i \in O} (d_i(n) - y_i(n))^2 \tag{2.30}$$

Se disponiamo di N esempi di allenamento allora possiamo definire l'errore totale mediato (spesso chiamato anche **rischio empirico**):

$$\mathcal{E} = \langle e(n) \rangle = \frac{1}{N} \sum_{n=1}^{N} e(n) = \frac{1}{N} \sum_{n=1}^{N} \sum_{i \in O} C(d_i(n), y_i(n)) \tag{2.31}$$

A seconda di come effettuiamo l'allenamento possiamo distinguere due diversi tipi di apprendimento:

- Apprendimento per batch[4]
- Apprendimento stocastico

Batch Learning

In questo tipo di apprendimento l'aggiornamento dei pesi attraverso la (2.12) viene effettuato una volta che sono stati processati tutti gli N esempi dell'insieme di allenamento, i pesi vengono quindi aggiornati alla fine di ogni epoca. Il gradiente è

[4] deriva dall'inglese **batch learning** dove *batch* significa lotto, in questo caso vengono mediati tutti gli esempi di allenamento.

ottenuto dunque mediando su tutti gli esempi:

$$\nabla F \sim \frac{1}{N} \sum_{x \in X} \nabla F_x \tag{2.32}$$

dove ∇F_x è il gradiente stimato per l'esempio x. La stima del gradiente attraverso questo metodo risulta più accurata e questo garantisce, sotto alcune condizioni, la convergenza del metodo della discesa del gradiente ad un minimo locale. Possiamo quindi riassumere i vantaggi di questo metodo:

- Condizioni di convergenza di facile valutazione
- Possibilità di implementare metodi del second'ordine per velocizzare l'apprendimento [LeCun et al., 2012].
- Possibilità di parallelizzare l'apprendimento

Questo modo di aggiornare i pesi porta ad un aumento del tempo impiegato per il calcolo del gradiente con l'aumentare della dimensione dell'insieme di apprendimento.

Apprendimento Stocastico

In questo tipo di apprendimento l'aggiornamento dei pesi viene effettuato dopo ogni esempio attraverso la (2.12):

$$w(t+1)^l_{ij} = w(t)^l_{ij} - \eta \frac{\partial \mathcal{E}_i}{\partial w^l_{ij}}(t) \tag{2.33}$$

La stima del gradiente utilizzando solamente un esempio risulta "rumorosa", questo è utile in quanto spesso le reti neurali che utilizzano funzioni di attivazione non lineari hanno più di un minimo locale di differente profondità. Usando il batch learning si trova il minimo locale relativo al bacino in cui si inizia la discesa del gradiente (dipendenza dai pesi iniziali), nell'apprendimento stocastico, invece, grazie al "rumore" presente nella stima del gradiente c'è la possibilità che i pesi passino da un bacino all'altro con un possibile minimo locale più profondo. Esiste una variazione di questo tipo di apprendimento che viene chiamato **mini-batch learning**. In questo caso vengono aggiornati i pesi dopo aver passato alla rete un sottoinsieme prefissato $\mathbf{B} \subset \mathbf{T}$ di esempi. Il gradiente viene quindi mediato su un numero minore di esempi. Questo permette di velocizzare molto l'apprendimento nel caso in cui i dati siano ridondanti. Consideriamo un insieme di allenamento di 1000 esempi composto da 10 copie di un insieme di 100 esempi, il gradiente calcolato sui primi cento esempi darà lo stesso risultato del gradiente su tutti e 1000 gli esempi ma risulta molto più veloce. Difficilmente si trovano dati con questo livello di ridondanza ma sono sempre presenti degli insiemi di regolarità che rendono la stima del gradiente su un piccolo sottoinsieme vantaggiosa. Riassumendo, le qualità di questo tipo di apprendimento sono:

- Velocità di apprendimento
- Spesso si ottiene in una soluzione migliore (la dimostrazione è presente solo per alcuni casi semplificati).
- Facilità di implementazione

Per questi motivi si utilizza spesso un apprendimento stocastico nei problemi di classificazione[Haykin et al., 2009].

2.5.2 Normalizzazione degli ingressi

La convergenza risulta migliorata se la media degli ingressi è vicina a zero. Infatti se avessimo ad esempio tutti gli input positivi i pesi nel primo strato nascosto potrebbero solo crescere o decrescere insieme, se un vettore dei pesi di un neurone deve cambiare direzione può farlo solo procedendo in modo discontinuo rallentando l'apprendimento [LeCun et al., 2012]. Un'altra linea guida da seguire per quanto riguarda gli ingressi è quella di scalarli in modo tale che la media sull'intero insieme di allenamento sia vicina a zero.

2.5.3 Scelta del *learning rate*

Un basso *learning rate* riduce le fluttuazioni dovute a valori diversi del gradiente ma rende più lento l'apprendimento. Quello che si fa di solito è abbassare il *learning rate* dopo un certo numero di epoche.

2.5.4 Regolarizzazione

La regolarizzazione consiste nell'aggiungere un termine alla funzione di costo in modo da stabilizzare la soluzione [Haykin et al., 2009]:

$$C(w,b) = \mathcal{E}(w,b) + \lambda R(w,b) \tag{2.34}$$

dove λ è il parametro di regolarizzazione. La scelta più comune per il fattore di regolarizzazione è la norma di L_2. Aggiungendo un termine alla funzione di costo vengono di fatto imposti dei vincoli sui valori che possono assumere i pesi. Minimizzare la somma di questi due termini ci permette di trovare un compromesso tra fittare bene i dati di allenamento e generalizzare il più possibile la soluzione trovata evitando l'*overfitting* dei dati di allenamento.

2.6 Discriminazione Gamma/Neutroni

La discriminazione gamma/neutroni che vogliamo implementare rientra nella classe dei problemi supervisionati. Per risolvere questo tipo di problemi si seguono i seguenti passi:

1. Scegliere il tipo di esempi da usare per l'allenamento.
2. Raccogliere un grande numero di esempi. Gli esempi usati per l'allenamento devono rappresentare in modo completo i dati reali.
3. Determinare il tipo di input da fornire alla rete.
4. Eseguire l'allenamento attraverso l'algoritmo di apprendimento.
5. Valutare l'accuratezza della funzione su un insieme di dati non utilizzati durante l'apprendimento. Questo insieme è chiamato **insieme di test**.

Nel prossimo capitolo andremo a costruire il *dataset* etichettato che utilizzeremo per addestrare la rete neurale. Le etichette che andremo ad associare ai segnali devono essere determinate in modo accurato in quanto queste rappresentano il "*ground truth*".

Capitolo 3

Misura sperimentale dei segnali di neutroni e raggi gamma

3.1 Rivelatori a scintillazione

Le misure sono state fatte utilizzando degli scintillatori organici. Il meccanismo di scintillazione di questi rivelatori si basa su transizioni nella struttura dei livelli energetici di una singola molecola rendendo il processo indipendente dallo stato fisico della stessa. Questo scintillatori utilizzano molecole organiche con alcune proprietà di simmetria che portano ad una struttura dei livelli energetici del tipo in figura (3.1). Nei livelli energetici sono presenti stati ($S_0, S_1, \dots$) di singoletto e stati ($T_1, T_2, \dots$) di tripletto. La separazione in energia tra lo stato S_0 ed S_1 è di $3 - 4eV$ mentre tra S_1 e stati successivi c'è una minor separazione, inoltre ogni stato presenta una ulteriore suddivisione dovuta ai diversi stati vibrazionali, questi stati sono separati tra loro di qualche frazione di elettronvolt.

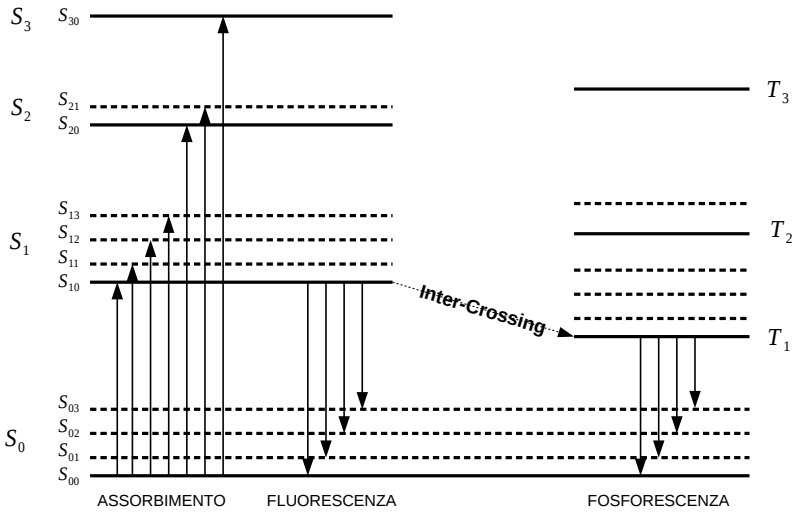
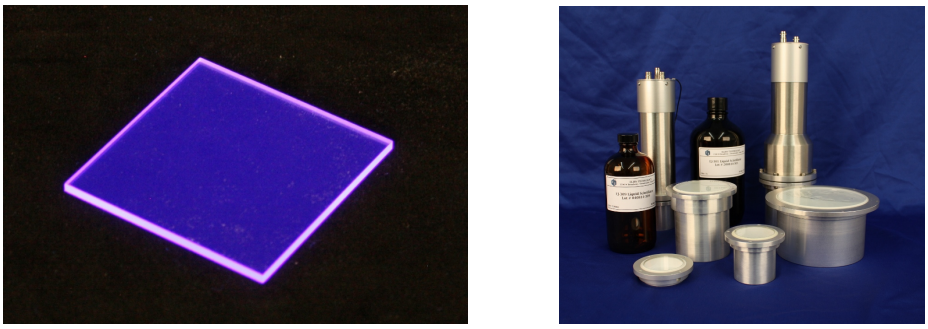


FIGURA 3.1: Struttura dei livelli energetici delle molecole di uno scintillatore organico. Questo tipo di struttura prende il nome di $\pi - electron$. [Knoll, 2010]

L'assorbimento dell'energia cinetica di una particella carica che passa nelle vicinanze della molecola porta gli elettroni dallo stato S_0 a stati eccitati (S_1,S_2,...) dopodiché attraverso conversioni interne gli elettroni in stati $S_2, S_3, \ldots$ vengono de-eccitati allo stato S_1 mentre gli stati con energia vibrazionale in eccesso, non essendo in equilibrio termico con gli elettroni vicini, la perdono arrivando a S_{10}. Dopo un tempo dell'ordine di qualche picosecondo si hanno dunque un certo numero di molecole eccitate nello stato S_{10}. Queste molecole producono, attraverso un decadimento rapido, luce di **fluorescenza** nella transizione da $S_{10} \rightarrow S_{0i}$ (dove S_{0i} indica uno degli i stati S_0 a diversa energia vibrazionale). Il tempo tipico di questo processo è di qualche nanosecondo. Alcune molecole però possono essere convertite in uno stato di tripletto T_1 attraverso un processo che prende il nome di *inter-system crossing*. Questo stato ha un tempo di decadimento molto maggiore in quanto la transizioni dirette allo stato S_0 sono proibite, questo rende più probabile una riconversione $T_1 \rightarrow S_{10}$ attraverso eccitazione termica a cui segue l'emissione di luce di fosforescenza dovuta a $S_{10} \rightarrow S_{0i}$. Questo secondo tipo di emissione viene chiamata **fluorescenza ritardata**, il tempo tipico di questo processo è di 100-500ns. Per le misure effettuate sono stati utilizzati degli scintillatori organici prodotti dalla ELJEN:

- EJ301, scintillatore liquido con proprietà di PSD.
- EJ228 scintillatore plastico usato come rivelatore di *start*.



(A) EJ228

(B) Scintillatori organici liquidi prodotti dalla ELJEN

FIGURA 3.2: Scintillatori organici plastici e liquidi.

Si è deciso di cominciare lo studio con uno scintillatore liquido dalle proprietà molto ben conosciute (l'EJ301), per poter valutare nel modo migliore le prestazioni della rete neurale rispetto a un caso standard. In un secondo tempo questa analisi verrà estesa al nuovo detector basato sullo scintillatore plastico.

3.2 Protocollo sperimentale

I dati sono stati acquisiti attraverso un digitizer della CAEN (V1730, 500MS/s, 14 bit) che digitalizza l'input e nel caso in cui la differenza tra il valore di riferimento e l'altezza del segnale digitalizzato supera un valore di soglia prefissato, viene raccolto l'evento. Per allenare la rete neurale abbiamo bisogno di dati con una precisa etichetta che li identifichi come neutroni o come gamma. L'identificazione del tipo di segnale è stata fatta applicando la tecnica della discriminazione di forma di impulso (*Pulse Shape Discrimination* o PSD). Per quanto riguarda scintillatori organici come

EJ301 e EJ228 questa differenza in forma è dovuta ai diversi meccanismi di interazione con la materia per neutroni e raggi gamma. I neutroni depositano la loro energia attraverso scattering elastico con protoni e carbonio che sono responsabili del meccanismo di eccitazione delle molecole e dunque della conseguente produzione di luce di scintillazione per fluorescenza. Per quanto riguarda i raggi gamma il meccanismo di fluorescenza è azionato dagli elettroni ionizzati dai gamma stessi. La differenza nei segnali è dovuta alla componente ritardata dell'emissione, infatti la densità di stati T_1 (responsabili della componente ritardata) lungo la scia della particella ionizzante è legata allo *stopping power* $-dE/dx$: particelle con uno stopping power maggiore presentano una maggior concentrazione di questi stati con conseguente aumento di luce di fluorescenza ritardata.

Per ottenere la PSD si definiscono i seguenti parametri (fig. 3.3):

- Pre-gate: Punto di inizio dell'integrazione
- Long Gate: numero di bin per l'integrazione totale del segnale
- Short Gate: numero di bin per l'integrazione parziale

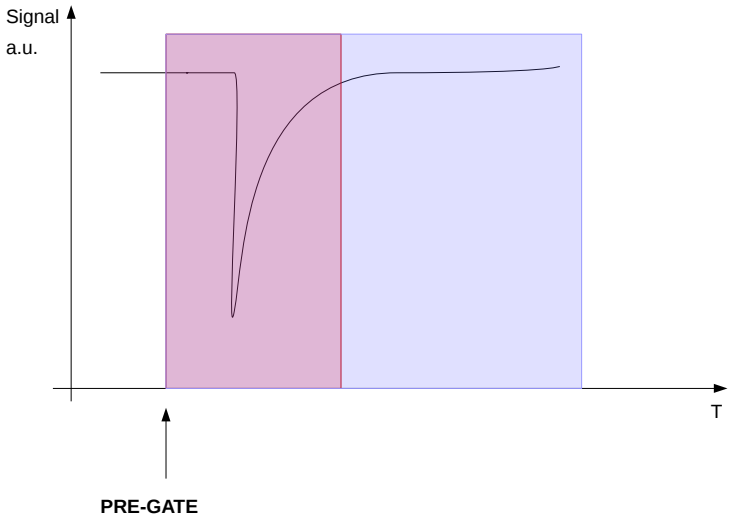


FIGURA 3.3: Aree di integrazione: in rosso il long gate e in blu lo short gate.

Il grafico utilizzato per la PSD si ottiene graficando il rapporto tra la differenza tra le due integrazioni e l'integrazione totale in ordinate:

$$PSD = \frac{Q_{long} - Q_{short}}{Q_{long}} \tag{3.1}$$

e l'energia dell'evento sulle ascisse.

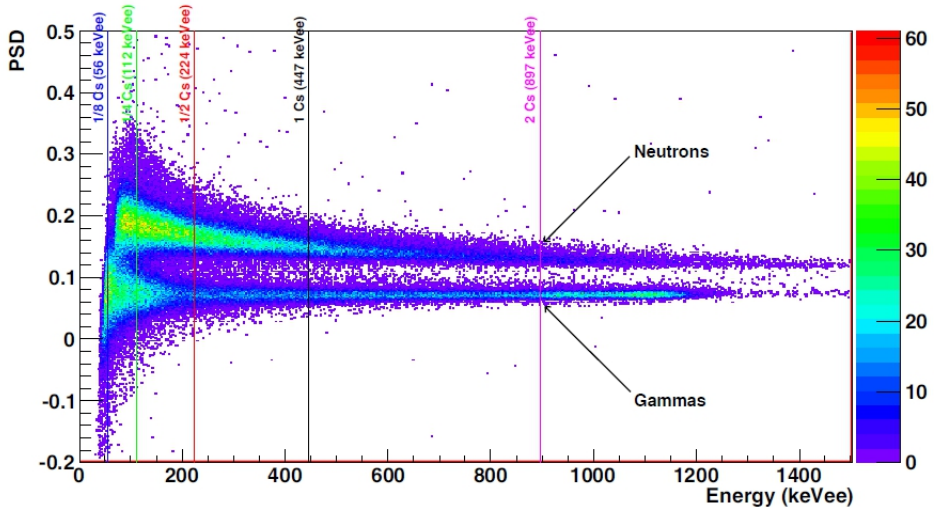
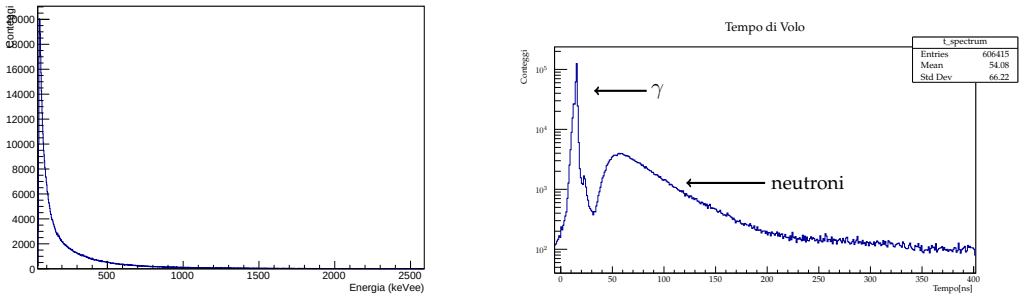


FIGURA 3.4: Grafico della psd. Fonte: http://www.caen.it/

3.3 Misura del tempo di volo

A basse energie è difficile discriminare i segnali attraverso la PSD. Con riferimento alla figura (3.4) notiamo che per energie minori di 447 keVee i segnali cominciano a mischiarsi. Per migliorare l'identificazione dei segnali si può utilizzare la misura dei tempi di volo: i γ infatti avranno una distribuzione piccata sul valore $t = d/c$ con d distanza tra il rivelatore e la sorgente di ^{252}Cf (fig. 3.6) mentre i neutroni avranno una distribuzione in tempo a seconda della loro energia (figura 3.5). Sarà possibile dunque definire una soglia sul tempo di volo così da distinguere i due tipi di radiazione.



(A) Spettro in energia ^{252}Cf

(B) Spettro temporale EJ301

FIGURA 3.5: EJ301

La misura del tempo di volo è fatta in coincidenza utilizzando l'EJ228 come rivelatore di start e l'EJ301 come rivelatore di stop , queste coincidenze vengono fatte via software all'interno dell'FPGA presente sulla scheda del digitizer. Per migliorare la stima del tempo in cui avviene l'interazione tra la radiazione e i rivelatori si utilizza un Constant Fraction Discriminator (o CFD) digitale. Il CFD consta nell'eseguire le seguenti operazioni sul segnale digitalizzato:

1. Duplicare il segnale

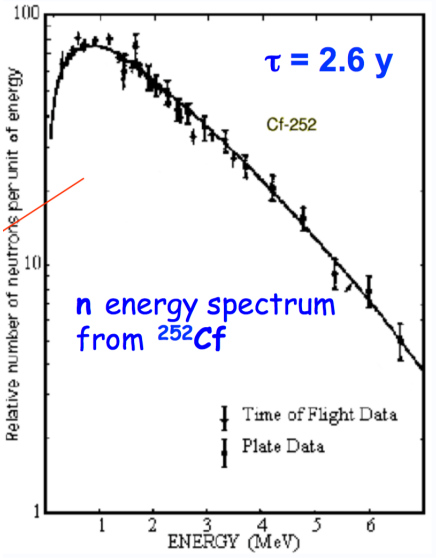


FIGURA 3.6: Spettro in energia dei neutroni del ^{252}Cf.

2. Attenuare il primo segnale
3. Invertire e ritardare il secondo segnale
4. Sommare i due segnali

i parametri di attenuazione sono da determinare sperimentalmente, nel caso dell'EJ301 si sono utilizzati i seguenti parametri:

- Fattore attenuazione 20%
- Ritardo 4ns

Il ritardo viene scelto in modo che il punto di attraversamento dello zero sia in corrispondenza al massimo del segnale in ingresso in questo modo il segnale del CFD risulta essere indipendente dall'ampiezza del segnale stesso. Il punto di attraversamento dello zero è una frazione costante del tempo di salita. In figura[1] (3.7) è evidente il vantaggio nell'utilizzo del CFD.

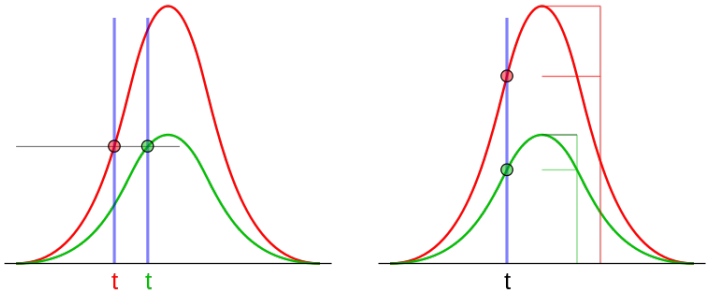


FIGURA 3.7: Differenza di triggering attraverso Leading Edge (sinistra) e CFD (destra)

[1]Fonte immagine: en.wikipedia.org/wiki/Constant_fraction_discriminator

3.4 Calibrazione in energia

La calibrazione in energia degli scintillatori è stata effettuata mediante l'analisi dei Compton edge dovuti a fotoni da 511 e 1275 keV prodotti da una sorgente di ^{22}Na. L'uso dell'analisi dei compton edge è motivata dal fatto che gli scintillatori utilizzati hanno un basso numero atomico medio e per questo non risultano sensibili al fotopicco, infatti la radiazione interagisce principalmente per scattering Compton. L'energia del Compton Edge è data da:

$$E_{CE} = \frac{2E_\gamma^2}{m_e c^2 + 2E_\gamma} \tag{3.2}$$

Però a causa della risoluzione limitata degli scintillatori il *Compton Edge* è soggetto ad uno *shift* a basse energie a seconda della risoluzione del rivelatore stesso. Simulando eventi di scattering Compton ed includendo una dispersione in energia è possibile osservare lo shift del Compton Edge (fig. 3.8). Quello che si osserva è che all'aumentare della dispersione in energia lo shift aumenta inoltre si nota anche un aumento degli eventi nella coda ad alte energie. Attraverso lo studio delle funzioni di risposta (fig. 3.9) di vari rivelatori si può correlare il valore del parametro σ/C al valore in keV della σ. Sempre da analisi di funzioni di risposta è possibile risalire allo shift del Compton Edge in (keV). Per calibrare gli spettri degli scintillatori si procede dunque in questo modo:

1. Fit gaussiano sul picco ad alte e basse energie
2. Calcolo del Parametro σ/C per ogni picco.
3. Ricavare σ in keV e da questo lo shift del Compton Edge
4. Fit lineare tra i due compton edge relativi al fotone da 511 keV e 1275keV.

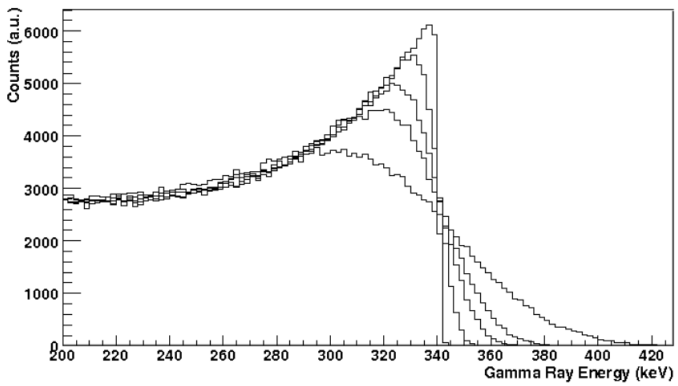


FIGURA 3.8: Distribuzione teorica degli eventi Compton con dispersione in energia Gaussiana

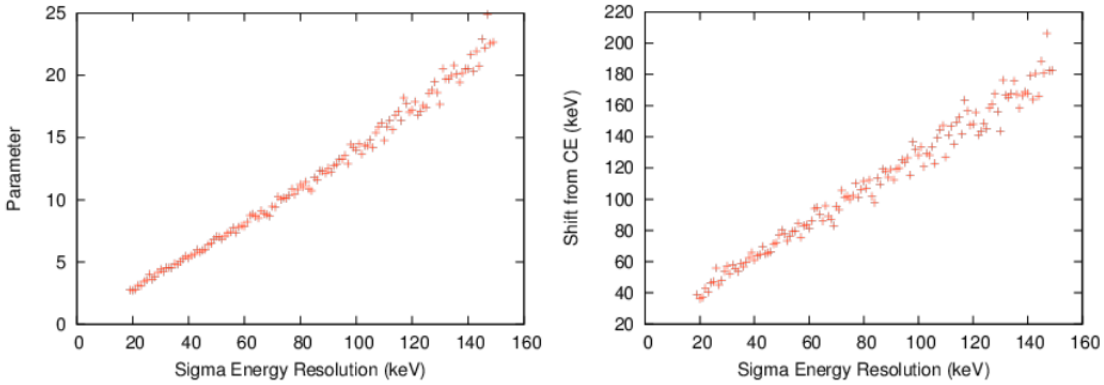


FIGURA 3.9: Funzioni di risposta per il fotone a 1275keV.
Fonte: Stevanato, 2016

3.4.1 Analisi dei segnali dell'EJ301

L'*EJ301* è uno scintillatore liquido prodotto dalla ELJEN utilizzato principalmente per PSD in presenza di γ e neutroni. Lo spettro in energia del ^{22}Na è:

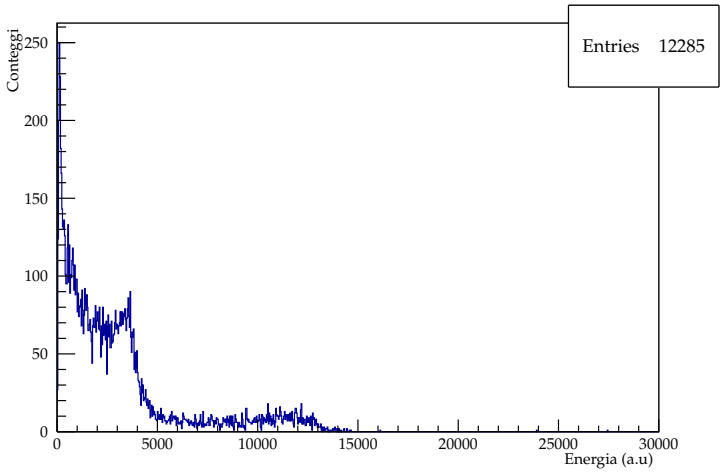


FIGURA 3.10: Spettro EJ301 non calibrato

Energia Fotone	Centroide	σ [a.u.]	σ/C [%]
511keV	3197	767.4	24%
1275keV	11200	1350	12%

TABELLA 3.1: Parametri fit EJ301

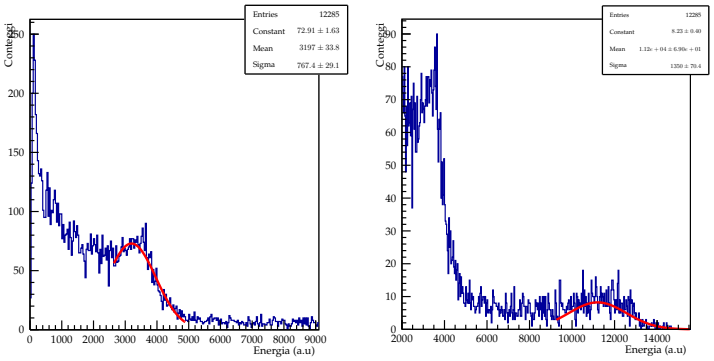


FIGURA 3.11: Fit gaussiano dei compton edge

Confrontando i parametri con le funzioni di risposta (fig. 3.9) si ottengono i seguenti valori:

Energia Fotone	σ [keV]	Shift C.E. [keV]
511 keV	40	25
1275 keV	80	60

TABELLA 3.2: Shift Compton EJ301

Enegia Fotone	C.E. nominale [keV]	Shift C.E. [keV]	C.E. Effettivo [keV]
511 keV	340	25	315
1275 keV	1062	60	1002

TABELLA 3.3: C.E. effettivi

Attraverso un fit lineare si è ottenuto un fattore di conversione di 11.6 canale/keVee.

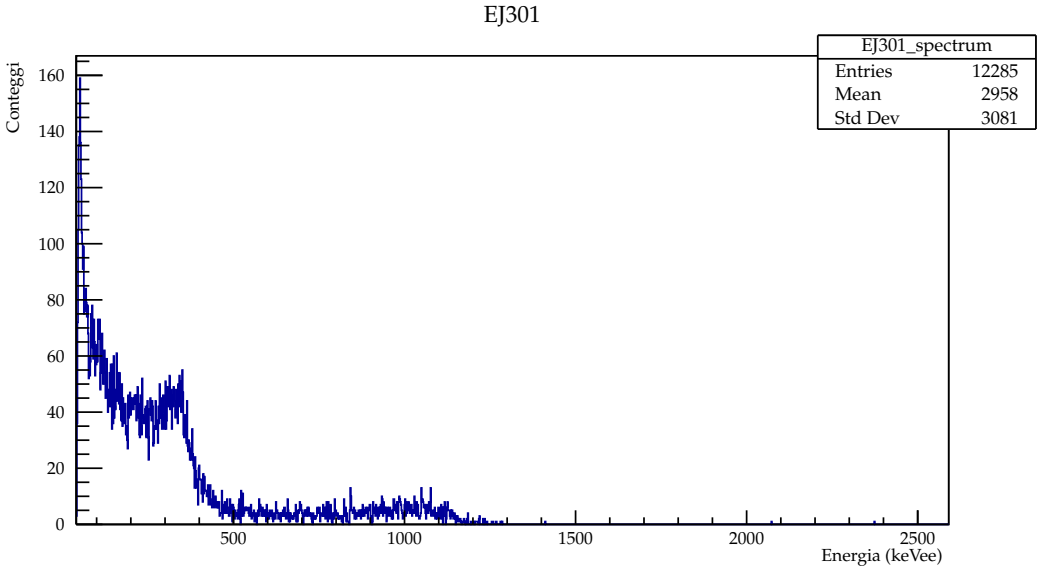


FIGURA 3.12: Spettro EJ301 calibrato

3.5 Preprocessamento dei dati

Prima di fornire alla rete i dati acquisiti dobbiamo preprocessarli in modo da eliminare alcune tipologie di eventi.

3.5.1 Eventi di Pileup

Gli eventi di Pileup sono eventi in cui oltre all'impulso principale è presente anche un secondo impulso di altezza maggiore del valore di soglia.Un esempio di evento di pileup è riportato in figura (3.13). Gli eventi di pilep risultano essere circa il 2%.

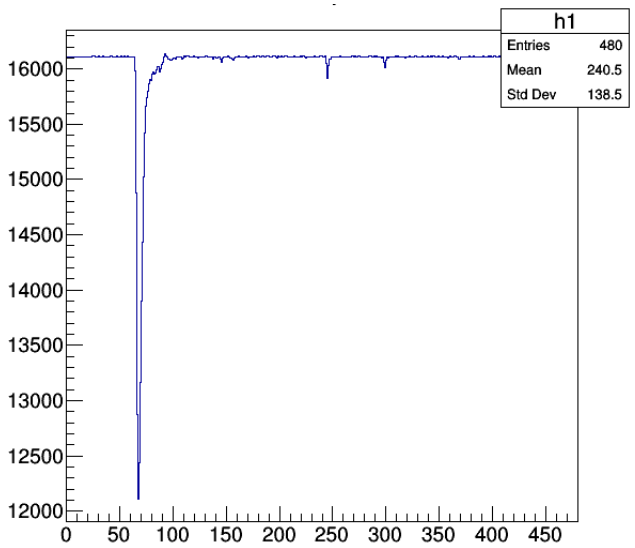


FIGURA 3.13: Evento di *pileup*.

3.5.2 Eventi cinematicamente proibiti

Abbiamo visto che il neutrone interagisce attraverso scattering elastico con protoni e nuclei di carbonio presenti nello scintillatore, a questi viene dunque fornita un'energia cinetica di rinculo data da:

$$E_R = \frac{4A}{(1+A)^2} cos^2(\theta) E_n \tag{3.3}$$

dove E_n l'energia del neutrone incidente e θ è l'angolo di deviazione del nucleo colpito rispetto alla linea di volo del neutrone. L'energia massima si ha per $\theta = 0$:

$$E_R = \frac{4A}{(1+A)^2} E_n \tag{3.4}$$

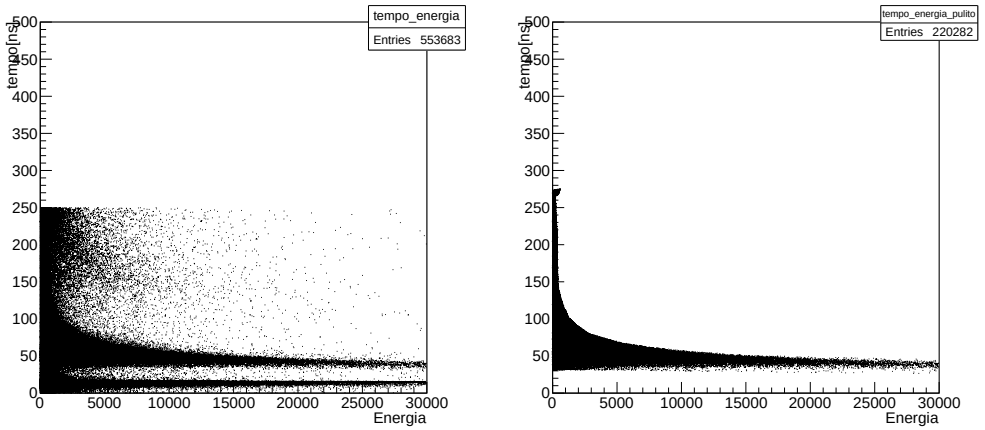
Il tempo di volo di un neutrone è dato da:

$$T = \frac{\sqrt{m}L}{\sqrt{2E}} \propto \frac{1}{\sqrt{E}} \tag{3.5}$$

Un neutrone con una certa energia $\hat{E}$ potrà allora rilasciare al nucleo colpito un'energia massima pari a:

$$E_R^{max} = \frac{4A}{(1+A)^2}\hat{E}_n \tag{3.6}$$

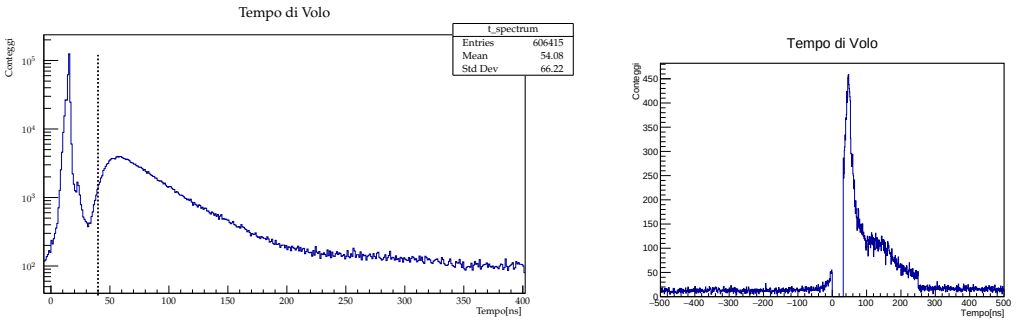
saranno da scartare tutti gli eventi che a T fissato hanno un'energia $E' > E_R^{max}$. In figura (3.14 A) sono riportati tutti gli eventi mettendo il tempo di arrivo in funzione dell'energia (in unita di Qlong) mentre in (3.14 B) sono rappresentati gli eventi cinematicamente permessi.



(A) Eventi totali (B) Eventi cinematicamente possibili

FIGURA 3.14: Eliminazione di eventi cinematicamente proibiti. L'energia è in Qlong.

Riportiamo in figura (3.16) i dati della sorgente di ^{252}Cf etichettati attraverso l'uso dei tempi di volo, i puntini neri indicano i neutroni mentre i puntini rossi i gamma. Notiamo che nella zona dei gamma sono presenti molti punti neri relativi ai neutroni, questi sono dovuti al gate imposto sul tempo di volo. Scegliendo una soglia come quella riportata in figura (3.15 A) stiamo etichettando come neutroni anche alcuni eventi che sono nella coda della distribuzione temporale dei raggi gamma (3.15 B). Lo stesso problema si presenta nella zona dei neutroni. Per evitare questo tipo di classificazioni errate si unisce al gate sul tempo di volo un gate sulla PSD cosi da eliminare questa tipologia di eventi. In questa fase di preselezioe il gate sulla PSD è tenuto volutamente molto lasco in modo da non perdere possibili segnali buoni di neutrone. Gli eventi preprocessati vengono presentati nel prossimo capitolo.



(A) Gate sul tempo di volo

(B) Coda della distribuzione temporale dei gamma

FIGURA 3.15: Problematiche nell'uso dei tempi di volo

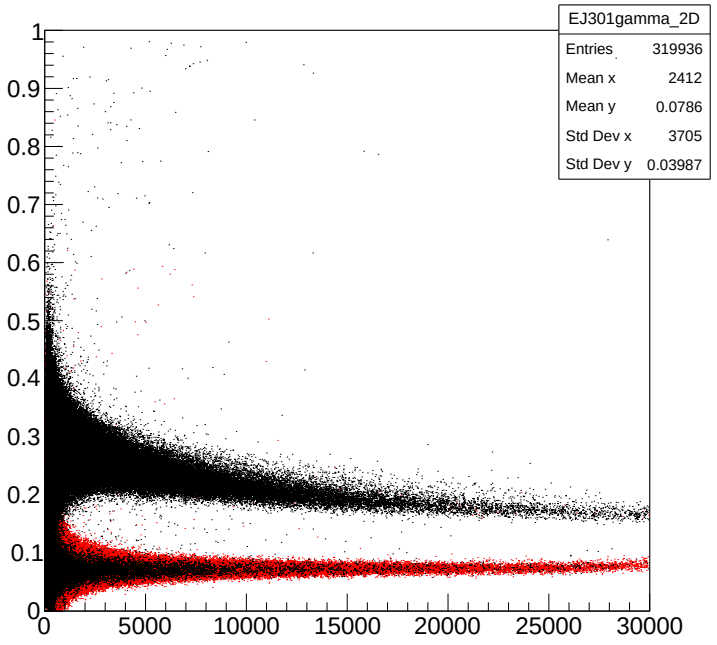


FIGURA 3.16: Grafico della PSD. In ascisse Energia [Qlong] e in ordinate il parametro PSD.

Capitolo 4

Discriminazione gamma/neutroni

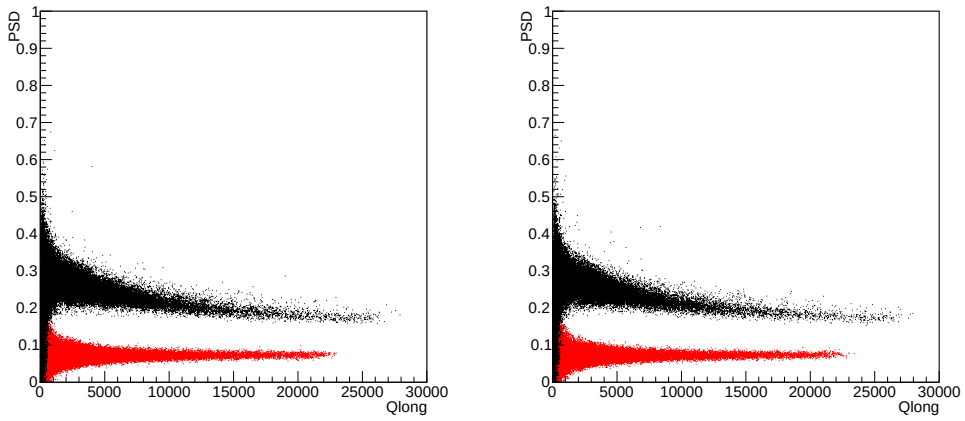
Per costruire una *baseline* si è implementato anche un semplice discriminatore lineare cosi da contestualizzare le prestazioni della rete neurale. Il metodo di discriminazione attraverso l'utilizzo della PSD verrà quindi confrontato con entrambi i classificatori.

4.1 Suddivisione dei dati

I dati preprocessati sono stati divisi nel seguente modo:

- Insieme di allenamento:
- Insieme di validazione
- Insieme di test
- Quattro insiemi di test suddivisi in 5 diverse classi di **Qlong** (0-350,350-500,500-1000,1000-1500,1500-2000).

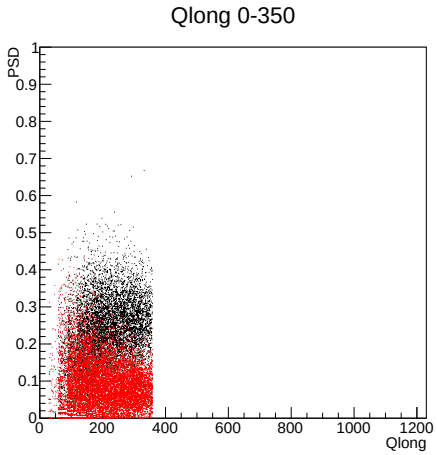
I quattro insiemi di test sono stati utilizzati per monitorare il comportamento della rete nella zona di energie dove la PSD classica non può essere applicata.



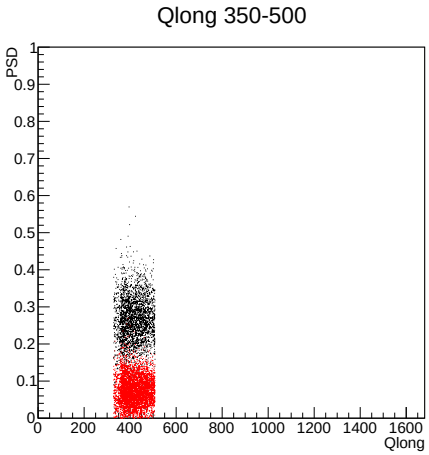
(A) Insieme di allenamento

(B) Insieme di test e validation

FIGURA 4.1: .

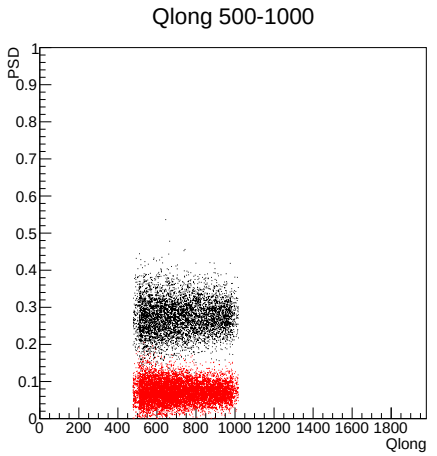


(A) Insieme di test 0-350

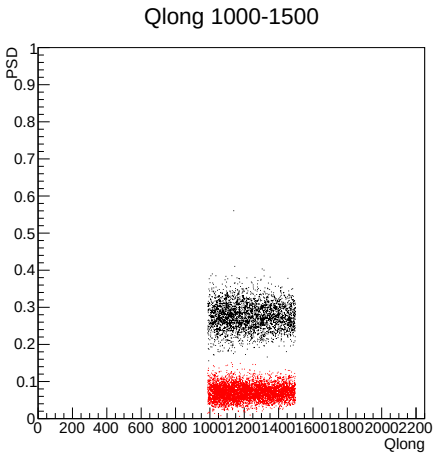


(B) Insieme di test 350-500

FIGURA 4.2: .



(A) Insieme di test 500-1000



(B) Insieme di test 1000-1500

FIGURA 4.3: .

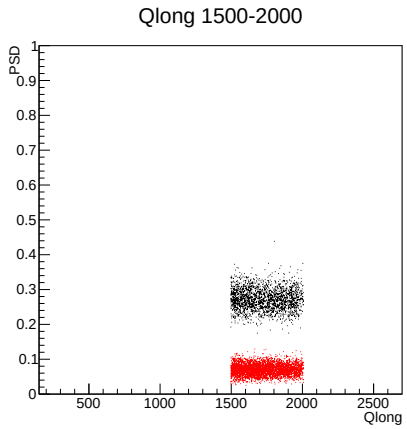


FIGURA 4.4: Insieme di test 1500-2000

4.2 Classificatore lineare

Il learning rate per questo classificatore è stato impostato al valore di $\eta = 0.1$. La dimensione dei mini-batch usata è pari a 600. In figura (4.1) è riportato il grafico di apprendimento.

Energia [keVee]	Numero eventi	Errore[%]	Efficienza Classificatore Lineare [%]	Efficienza PSD[%]
0-30.7	18224	27.02	97.58	71.20
30.7-43.85	6503	1.53	98.47	73.95
43.85-87.71	13062	0.24	99.75	96.36
87.71-131.6	7206	0.01	99.98	99.90
131.6-175.44	5005	0	99.97	99.99

TABELLA 4.1: Confronto classificatore lineare e PSD

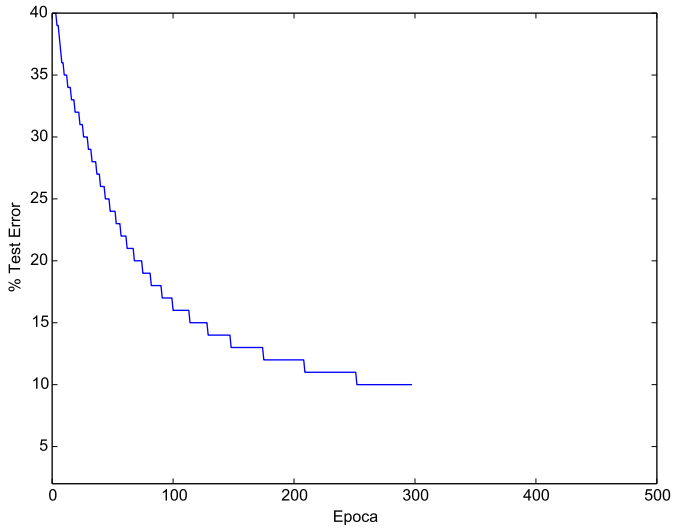


FIGURA 4.5: Grafico dell'apprendimento per il classificatore lineare

4.3 Rete neurale

La rete neurale utilizzata è governata dai seguenti iperparametri:

- Neuroni di ingresso: 480 (numero di campionamenti del segnale operata dal digitizer)
- Neuroni strato nascosto: 800
- Neuroni strato finale: 2 (classe neutroni e classe raggi gamma)
- Learning rate iniziale: 10^{-1}
- Grandezza mini-batch: 50
- Fattore di regolarizzazione (L_2): 10^{-4}

Il learning rate è stato abbassato secondo il seguente schema:

- Dopo 5 epoche: 10^{-2}
- Dopo 10 epoche: 10^{-3}
- Dopo 25 epoche: 10^{-4}

L'allenamento è stato svolto in 24 minuti, in figura (4.6) è riportato l'andamento dell'apprendimento mentre le prestazioni di discriminazione della rete sono presentate nella tabella (4.2). I risultati sono commentati nelle conclusioni.

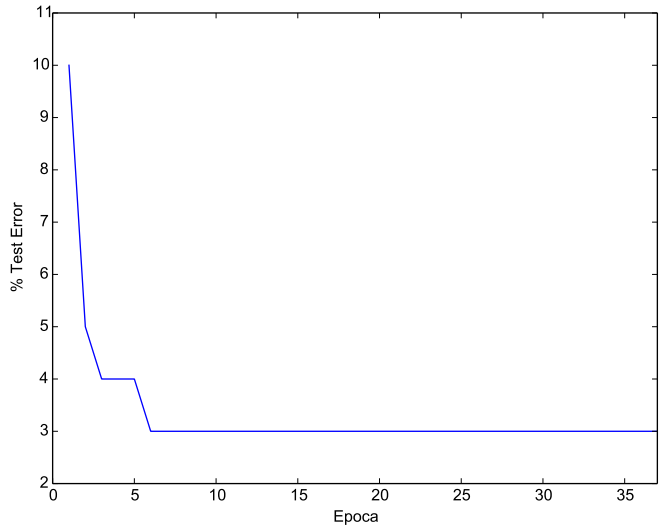


FIGURA 4.6: Andamento dell'apprendimento

Energia [keVee]	Numero eventi	Errore[%]	Efficienza Rete Neurale [%]	Efficienza PSD[%]
0-30.7	18224	6.92	90.00	10.80
30.7-43.85	6503	0.52	96.00	54.10
43.85-87.71	13062	0.14	99.30	94.50
87.71-131.6	7206	0.01	99.80	99.80
131.6-175.44	5005	0	100	99.99

TABELLA 4.2: Confronto rete neurale PSD.

4.4 Conclusioni

Il classificatore lineare, seppur più efficiente rispetto alla PSD (a parità di errore), a basse energie presenta un elevato errore che cala rapidamente salendo in energia (tabella 4.1). Per quanto riguarda la rete si vede dalla tabella (4.2) che a basse energie (0-44 keVee) presenta una migliore capacità di discriminare i due tipi di radiazione rispetto al metodo usuale mentre quando l'energia della radiazione aumenta, si vede che entrambi i metodi presentano ottime efficienze. Questi risultati giustificano l'introduzione di questo nuovo metodo, dal momento che nelle misure di carattere ambientale lo spettro dei neutroni del fondo ambientale presenta una forte componente di neutroni di bassa energia.

Gli sviluppi futuri si diramano su due fronti. Il primo, riguardante la parte strumentale, consta nella sostituzione dell'EJ301, che presenta problematiche ambientali in quanto tossico, con un rivelatore plastico, prodotto dalla ELJEN ed assemblato presso i Laboratori Nazionali di Legnaro, le cui caratteristiche si avvicinano a quelle dello scintillatore finora utilizzato. Il secondo fronte, rivolto alla parte software, consiste nello studio di diversi tipi di rete neurale: un primo miglioramento potrebbe essere quello di aggiungere un ulteriore strato nascosto e valutarne il comportamento. Per inquadrare meglio i risultati ottenuti si potrebbe compiere uno studio sistematico di diverse tipologie di reti neurali e un indagine approfondita degli iperparametri che le regolano[1].

[1] Attraverso l'uso di centri di calcolo si potrebbe implementare un ciclo sui vari iperparametri ed ottenere per ogni configurazione un grafico di apprendimento così da trovare la configurazione ottimale

Bibliografia

Haykin, Simon S et al. (2009). *Neural networks and learning machines*. Vol. 3. Pearson Upper Saddle River, NJ, USA:

Knoll, Glenn F (2010). *Radiation detection and measurement*. John Wiley & Sons.

Kriesel, David (2009). "A Brief Introduction to Neural Networks. 2007". In: *Url: http://www. dkriesel. com*.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015). "Deep learning". In: *Nature* 521.7553, pp. 436–444.

LeCun, Yann A et al. (2012). "Efficient backprop". In: *Neural networks: Tricks of the trade*. Springer, pp. 9–48.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1988). "Learning representations by back-propagating errors". In: *Cognitive modeling* 5.3, p. 1.

Stevanato, Luca (2016). *Dispense per il corso di Laboratorio di Fisica IV*.