Una Rete Neurale per la classificazione di caratteri scritti a mano

Progetto finale - Corso di Intelligenza Artificiale I
Lorenzo Addazi (213772)
addazi.lorenzo@gmail.com

• *Introduzione*

Storicamente, i primi interessi nei confronti del modello delle Reti Neurali si registrarono tra gli anni cinquanta e sessanta del XX secolo con lo studio del funzionamento del sistema nervoso ed in particolare delle capacità di calcolo logico ed aritmetico binario di questo.

Il problema del riconoscimento di Pattern è stato uno dei primi campi d'applicazione pratica delle reti neurali, la sfida era quella di cercare di capire come potesse esser possibile che sistemi semplici come il sistema nervoso potessero esser in grado di riconoscere oggetti a prescindere da dimensione, orientazione e sfondo, a differenza dei potenti calcolatori che risultavano invece incapaci di far ciò in modo efficiente.

Il sistema realizzato fornisce un esempio di riconoscimento di Pattern finalizzato alla classificazione di caratteri scritti a mano, ciò per mezzo di un percettrone con apprendimento supervisionato ed algoritmo di backpropagation dell'errore.

• *Struttura del Progetto*

E' possibile suddividere il lavoro del sistema in 4 fasi fondamentali :

1. *Predisposizione del Data Set ;*
2. *Predisposizione dell'architettura della Rete Neurale ;*
3. *Addestramento Supervisionato della Rete Neurale ;*
4. *Validazione su Test Set della Rete Neurale ;*

Sulla base del sistema definito è stato svolto un lavoro di confronto delle prestazioni al variare della struttura della Rete Neurale, in particolare i fattori che sono stati considerati sono i seguenti :

1. *Tipologia di Percettrone, Semplice o Multistrato ;*
2. *Numero di neuroni nello strato Hidden in caso di Percettrone Multistrato ;*
3. *Valore di Coefficiente di Apprendimento della Rete Neurale, quanto la rete è influenzata da ogni passo di addestramento;*
4. *Valore di Momentum della Rete Neurale, quanto effettivamente l'apprendimento viene applicato sui pesi sinaptici della rete ;*

1. *Predisposizione del Data Set*

La fase di Apprendimento Supervisionato della Rete Neurale richiede un insieme di esempi di corretta classificazione detto Data Set, tale insieme è composto da coppie ***< Input Pattern, Class >*** *dove* ***Input Pattern*** *rappresenta una sequenza di valori in input alla Rete Neurale (tanti quanti sono i Neuroni nello Strato Input) e* ***Class*** *rappresenta il valore di Output Ideale che*

si vuole sia prodotto dalla Rete Neurale, in particolare questo sta ad indicare la corretta classificazione che ci si aspetta.

Nel caso particolare, il Sistema è stato implementato in modo che va ad acquisire delle immagini di caratteri scritti a mano dalla cartella "DataSet/Normal" per poi estrarre da queste una rappresentazione binaria e serializzarla in Array Unidimensionale.

Per l'estrazione di Immagini Binarie dall'insieme di Immagini iniziale è stato implementato un semplice algoritmo che presuppone di lavorare con immagini sufficientemente "pulite", nelle quali dunque v'è sufficiente contrasto tra sfondo e corpo del carattere :

```
Algorithm ConvertImageToBinary ( image Source, image Destination ) :
        int thresholdValue = 170;
        for each pixel in Source do :
                if pixel has RGB value < thresholdValue then
                    Destination( pixel ) = [ 0, 0, 0 ]
                else
                    Destination( pixel ) = [ 255, 255, 255 ]
```

Fig. 1 - Algoritmo di conversione delle immagini in immagini binarie

Al termine del processo di conversione delle immagini, si avranno in "DataSet/Binary" i Pattern corrispondenti ad ogni carattere sotto forma di Immagine Binaria come illustrato in figura 2.

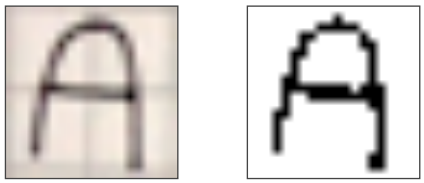

Fig. 2 - Esempio di immagine Input ed associata immagine binaria

L'input del Data Set sarà dunque una Matrice avente tanti record quante sono le immagini binarie ottenute (52 - caratteri minuscoli e maiuscoli) ognuno dei quali sarà un Array Unidimensionale di dimensione pari al numero di pixel dell'immagine (20 x 20 pixel per carattere).

Il valore di Output per ogni Classe è stato infine ottenuto stabilendo 26 valori (tanti quanti i caratteri dell'alfabeto) nell'intervallo [0, 1], tale limitazione trova spiegazione nella scelta della funzione Sigmoide (Fig. 3) come funzione di attivazione.

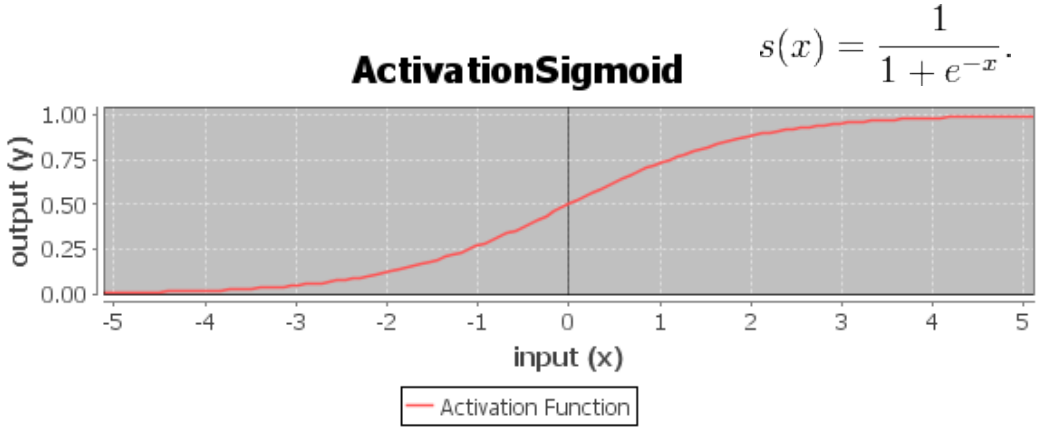

Fig. 3 - Definizione e grafico della funzione Sigmoide

2. *Predisposizione dell'architettura della Rete Neurale*

Si lavora con immagini di dimensione 20 x 20 pixel dunque il numero di Neuroni Input della Rete Neurale sarà pari a 400, uno per pixel.

Lo strato di Output sarà invece composto da un unico neurone, i valori associati alle classi sono infatti nell'intervallo [0, 1] e la Rete Neurale dovrà dunque produrre un unico valore.

La presenza o meno di uno strato di Neuroni Hidden ed il numero di questi, come detto, rientra tra gli aspetti dell'architettura della Rete Neurale modificati attraverso le varie verifiche, si rimanda dunque il discorso alle osservazione sperimentali.

3. *Addestramento Supervisionato della Rete Neurale*

La Rete Neurale viene addestrata sull'intero Data Set fino ad ottenere un valore di errore inferiore allo 0.001, proprio in base al tempo di convergenza sono state fatte le considerazioni d'efficienza delle varie architetture testate.

La metodologia di allenamento è gestita dal Framework "Encog" di Jeff Heaton, in esso l'errore considerato ad ogni passo è inteso come Errore Quadratico Medio (Root Mean Square - RMS), ed il metodo di addestramento è tramite algoritmo di Backpropagation per la propagazione dell'errore calcolato.

Al termine del processo di addestramento della rete, dunque al momento della convergenza ad un valore d'errore inferiore allo 0.001, il sistema produce un Plot dell'andamento del valore di errore nel numero delle epoche di addestramento passate utilizzando la libreria Open Source JFreeChart (http://www.jfree.org/jfreechart/ *).*

4. *Validazione su Test Set della Rete Neurale*

Terminata la fase di addestramento, la Rete Neurale viene testata su un insieme di codifiche di caratteri mostrando l'effettiva capacità di riconoscimento acquisita di classificare caratteri scritti a mano.

• *Osservazioni Sperimentali*

Come accennato, scopo del progetto è quello di fornire un confronto di prestazioni della Rete Neurale al variare dell'architettura; partendo dal modello di Percettrone Semplice si è arrivati fino ad una versione di Percettrone Multistrato più efficiente frutto delle osservazioni fatte su architetture intermedie.

Il primo modello architettonico sperimentato è stato quello di un Percettrone Semplice con 400 Neuroni Input ed 1 Neurone Output, con un fattore di apprendimento 0.001 ed un valore di momentum 0.9 quel che ci si aspettava era una Rete Neurale molto lenta nell'apprendimento.

La fase di addestramento è strutturata in modo da perseverare fino ad ottenere un valore di errore inferiore a 0.001, con questa prima architettura si ottiene tale obiettivo dopo circa 27.500 epoche.

Dalla figura 4 è possibile vedere come la Rete Neurale abbia notevoli difficoltà nel raggiungere un determinato livello di precisione, seppur riesca facilmente nella classificazione di Pattern altamente differenti; in una prima

fase l'errore decresce infatti esponenzialmente fino ad un valore circa pari a 0,025 da cui assume un comportamento decrescente quasi costante, causa dell'enorme numero di epoche risultante.

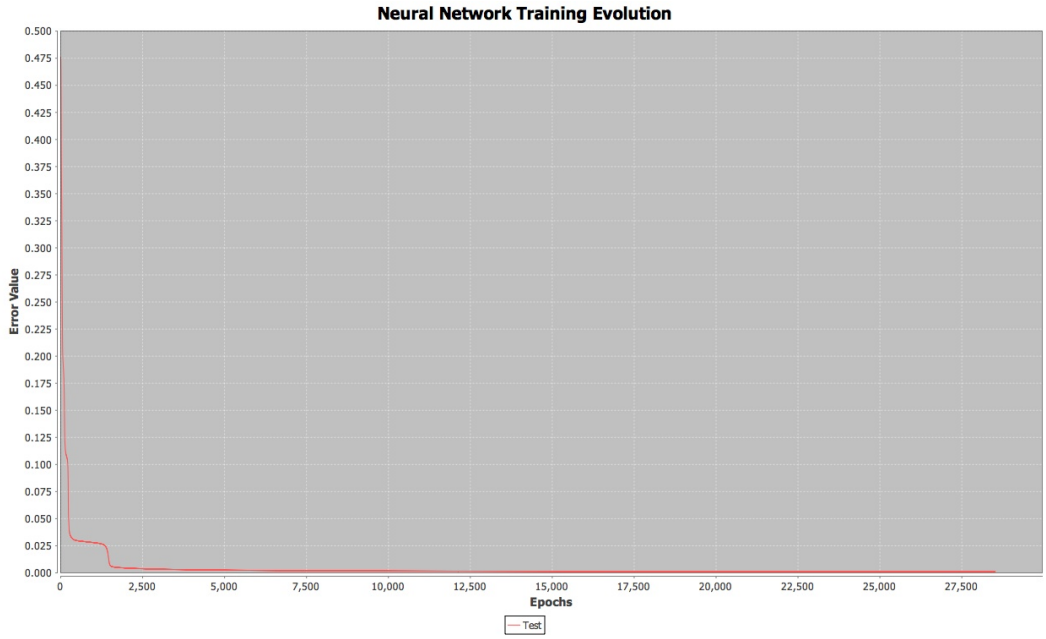

Fig. 4 - Grafico di andamento dell'errore durante l'addestramento

Passo successivo è stato quello di introdurre uno strato intermedio (Hidden Layer) tra Input ed Output, inizialmente composto da un unico neurone mantenendo stessi valori di coefficiente di apprendimento e momentum.

I risultati ottenuti sono stati deludenti, la Rete Neurale converge ad un errore inferiore a 0.001 in un numero di epoche circa pari a 700.000.

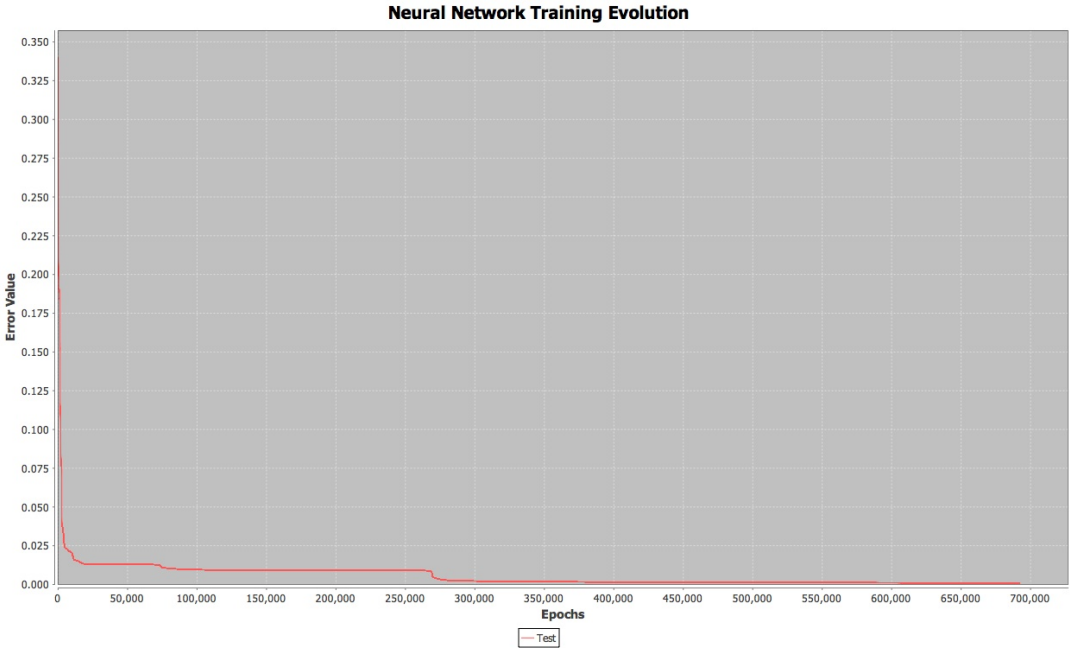

Fig. 5 - Grafico di andamento dell'errore durante l'addestramento

Dalla figura 5 è possibile vedere come il comportamento osservato nell'impostazione precedente si sia amplificato, anche in questo caso infatti l'errore ha un andamento inizialmente esponenzialmente decrescente fino ad un valore circa pari a 0.0015 per poi assumere un comportamento decrescente quasi costante che porta alla convergenza in un numero enorme di epoche.

Nonostante le prestazioni deludenti, l'introduzione di un Hidden Layer nella Rete Neurale, dunque l'utilizzo di un Percettrone Multistrato anziché Semplice, apre le porte ad una vasta scelta di possibili ottimali configurazioni variando il numero di Neuroni Hidden.

Il numero di Neuroni Hidden rappresenta un punto critico nella progettazione di una Rete Neurale, non è infatti stabilito un numero genericamente corretto di questi ma solo una serie di indicazioni da seguire per aver maggior probabilità di trovare la soluzione ottimale.

Nella serie di sperimentazioni, si è tenuto conto delle due seguenti indicazioni relativamente al numero di Neuroni Hidden :

1) Il numero di Neuroni Hidden dev'esser compreso tra la metà del numero dei Neuroni Input ed il numero di Neuroni Input stesso;

2) Il numero ottimale di Neuroni Hidden è pari alla somma di due terzi del numero di Neuroni Input ed il numero di Neuroni Output;

Seguendo la seconda indicazione, la terza prova è stata quella di modificare il numero di Neuroni Hidden da 1 a 267; le prestazioni rilevate sono state notevolmente migliori rispetto le due prove precedenti, la Rete Neurale converge infatti ad un valore di errore inferiore a 0.001 in poco meno di 10.000 epoche seppur vada comunque considerato che aumentando il numero di Neuroni Hidden la singola iterazione richiede maggior tempo di esecuzione.

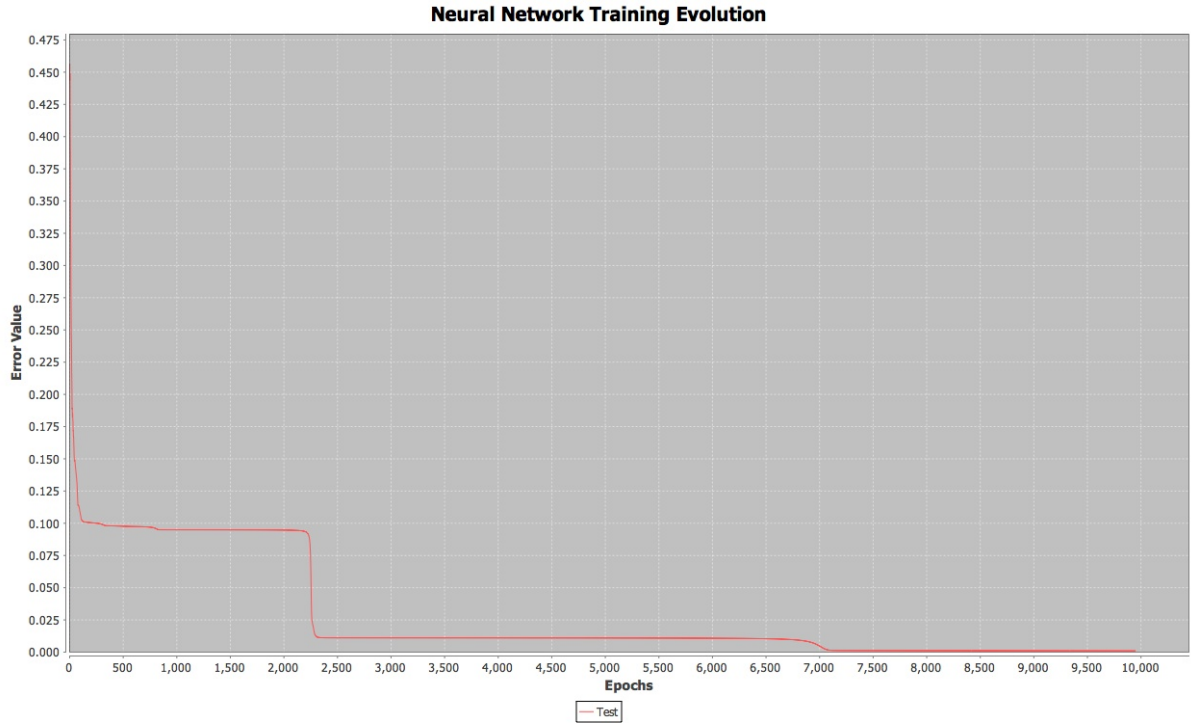

Fig. 6 - Grafico di andamento dell'errore durante l'addestramento

Dalla figura 6 è possibile osservare che continua ad esser presente il comportamento rilevato in precedenza, anche in questo caso l'errore tende ad assumere un comportamento decrescente quasi costante che comporta l'innalzamento del numero di iterazioni necessarie.

Pur avendo ottenuto risultati migliori, la prima indicazione fornisce ulteriori spunti di sperimentazione per migliorare le prestazioni del sistema; quarta prova è stata dunque quella di utilizzare un numero di Neuroni Hidden esattamente a metà tra il Numero di Neuroni Input e la metà di questo, 300.

Dalla figura 7 vediamo come il numero totale di epoche richieste per ottenere un errore inferiore a 0.001 è più alto del numero ottenuto al passo precedente, circa pari a 25.000; tuttavia l'andamento dell'errore commesso assume una forma leggermente differente rispetto ai casi precedenti, gli intervalli in cui la funzione tende quasi ad esser costante divengono più brevi e sono più frequenti gli intervalli di decrescenza esponenziale.

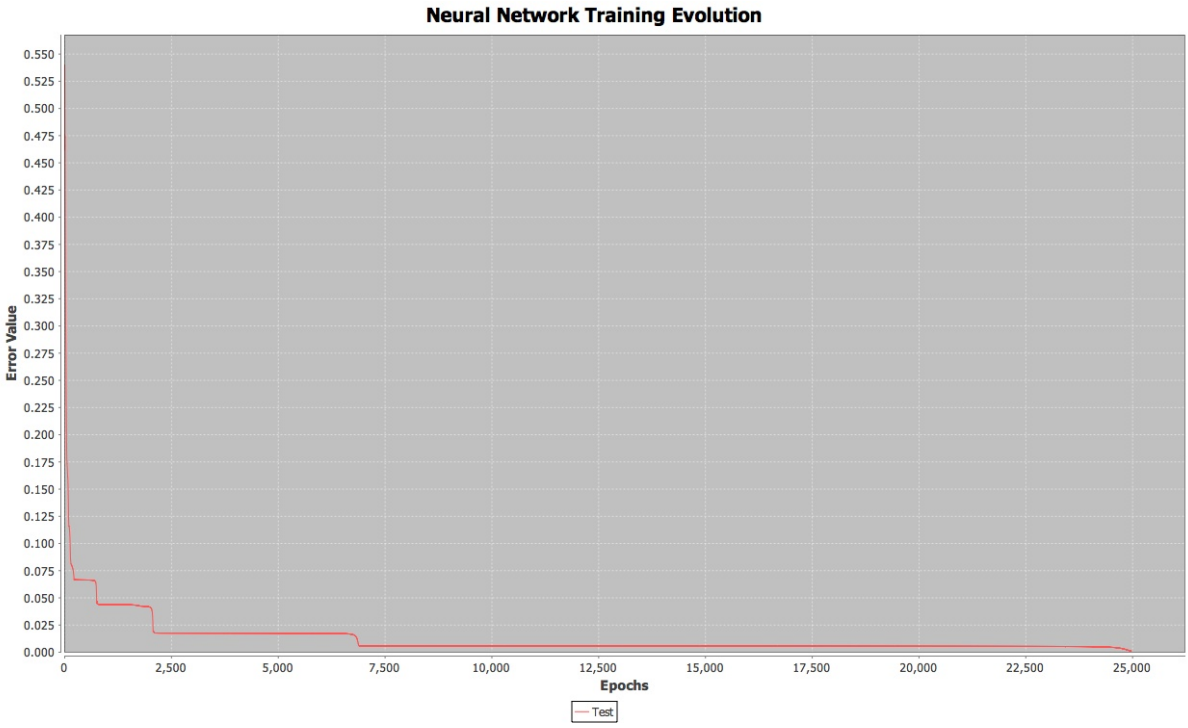

Fig. 7 - Grafico di andamento dell'errore durante l'addestramento

Prova successiva è stata invece con un numero di Neuroni Hidden esattamente pari alla metà dei Neuroni Input, dunque 200.

Il numero di epoche necessarie per ottenere un valore di errore inferiore a 0.001 diviene circa pari a 9.250 (Figura 8), valore radicalmente inferiore a quello ottenuto in precedenza e fino a questo punto il migliore ottenuto, tuttavia continua ad esserci un vasto intervallo di epoche nel quale l'errore decresce molto lentamente aumentando il numero delle iterazioni necessarie.

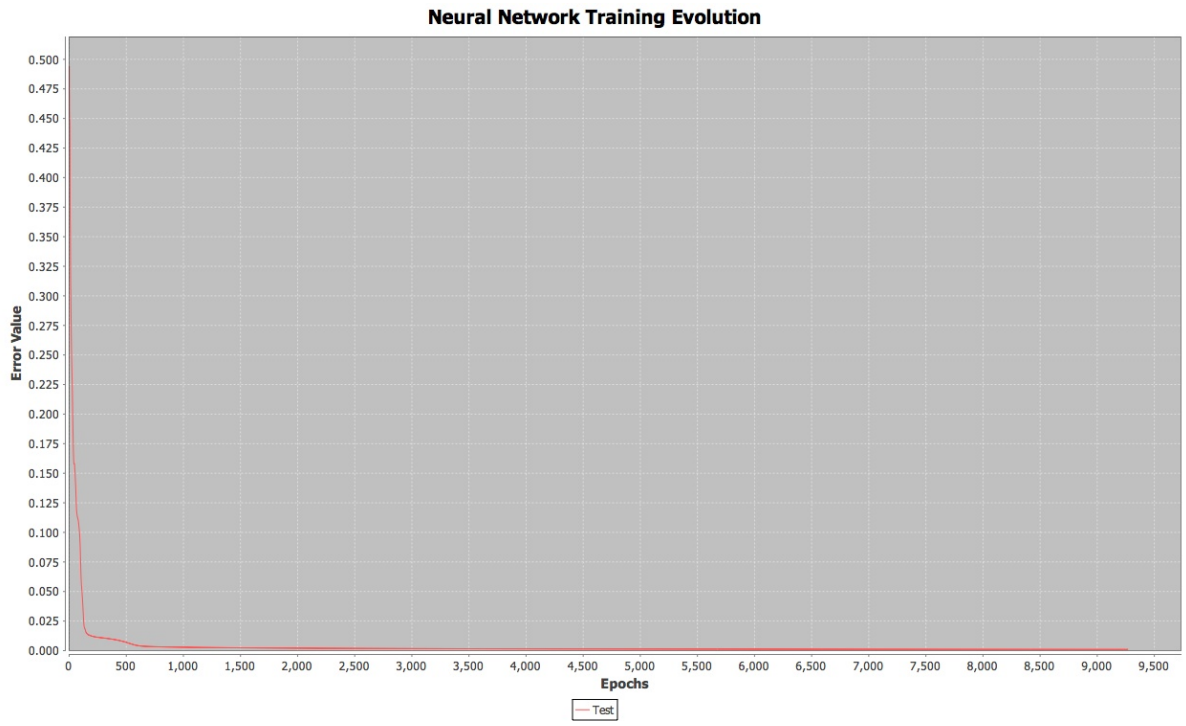

Fig. 8 - Grafico di andamento dell'errore durante l'addestramento

Dalle sperimentazioni precedenti è possibile osservare che molto probabilmente il numero di Neuroni Hidden ottimale si trova nell'intervallo compreso tra 200 e 267, estremi esclusi; passo successivo, è stato dunque quello di impostare il sistema con 230 Neuroni Hidden per la quinta prova.

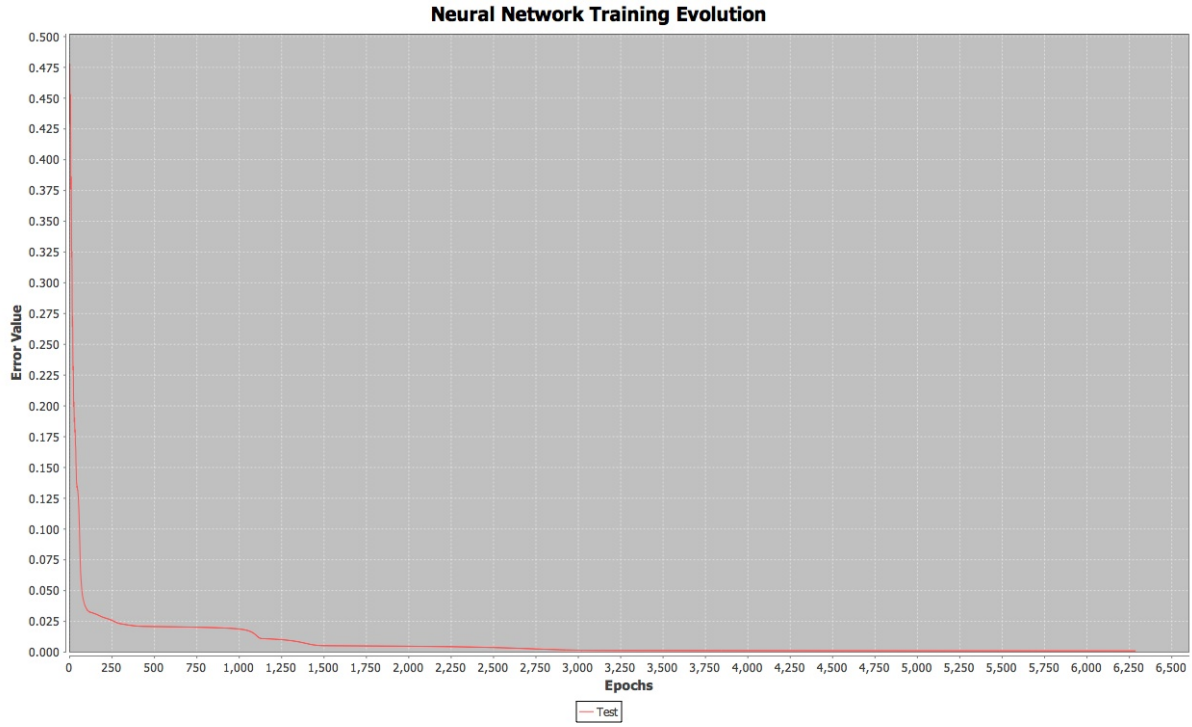

Fig. 9 - Grafico di andamento dell'errore durante l'addestramento

I risultati ottenuti sembrano confermare l'osservazione, il sistema ottiene un valore di errore inferiore a 0.001 in poco più di 6.250 epoche come mostrato dalla figura 9.

E' dunque possibile restringere l'intervallo precedente tra 200 e 231, estremi esclusi; passo successivi è stato allora quello di vedere cosa accade ponendo un numero di Neuroni Hidden pari a 220 per la sesta prova.

Con un numero di Neuroni Hidden pari a 220 si converge a valore di errore inferiore a 0.001 in poco più di

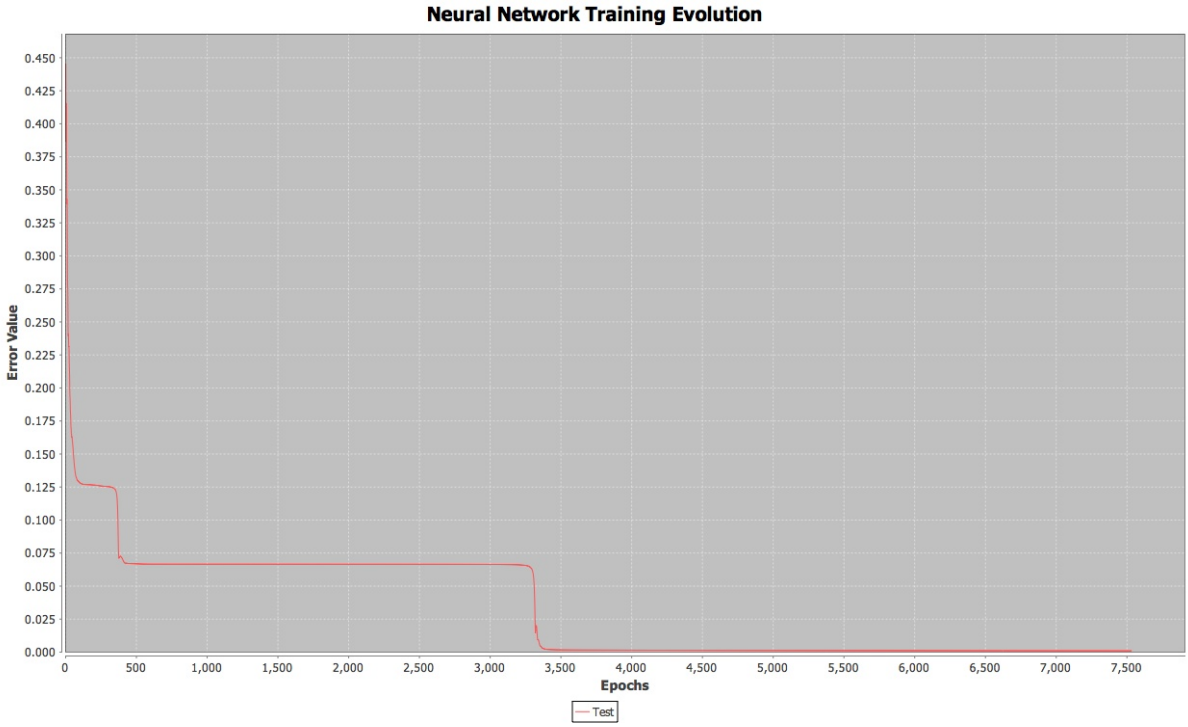

Fig. 10 - Grafico di andamento dell'errore durante l'addestramento

7.500 epoche come illustrato in figura 10, ciò conferma dunque che la scelta ideale è attualmente quella di 230 Neuroni Hidden come fatto nella quinta prova.

Pur essendo riusciti ad ottenere un numero di epoche di addestramento sempre inferiore modificando il numero di Neuroni Hidden, permane in ogni architettura un andamento sfavorevole del valore d'errore; in modo più o meno marcato, in ogni prova, il valore dell'errore tende inizialmente a decrescere esponenzialmente per poi rallentare vertiginosamente provocando un forte aumento del numero di iterazioni necessarie.

Per provare ad eliminare tal comportamento comune, si è passati ad effettuare diverse prove modificando il valore di coefficiente di apprendimento (Learning Rate) della Rete Neurale, prima prova è stata con coefficiente di apprendimento pari a 0.05.

Sono necessarie circa 2.700 epoche per ottenere un valore di errore inferiore a 0.001, il numero di iterazioni diminuisce dunque notevolmente rispetto alle prove pretendenti; é inoltre possibile notare da figura 11 come anche l'andamento del valore di errore sia radicalmente differente.

Immaginando il processo di addestramento della Rete Neurale come la ricerca di una porta in una stanza buia passo dopo passo, l'aumento del coefficiente d'apprendimento potrebbe esser paragonato all'aumento della lunghezza del passo fatto in ogni istante, un approccio meno cauto alla ricerca della soluzione dunque.

Nel grafico in figura 11 osserviamo infatti come la funzione non presenti più un unico andamento esponenziale decrescente iniziale bensì "rimbalzi" lungo serie di valori d'errore anche profondamente differenti per gran parte

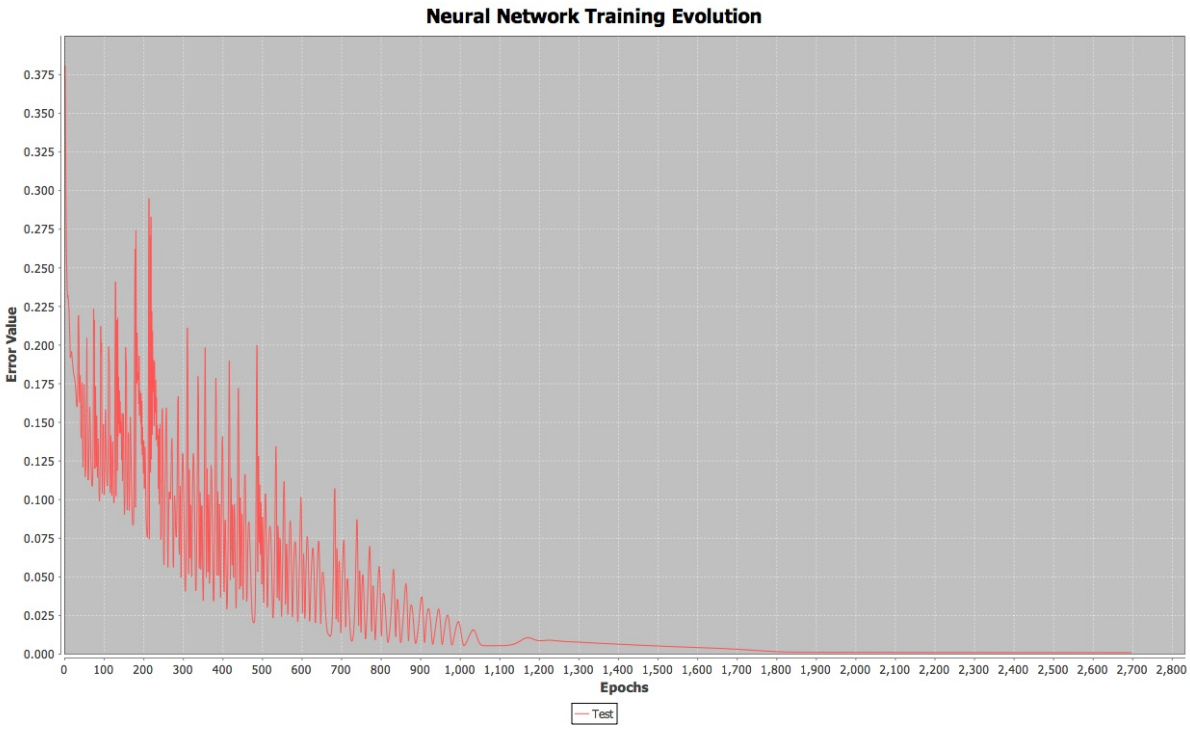

Fig. 11 - Grafico di andamento dell'errore durante l'addestramento

del processo di addestramento; solo nella parte finale è presente una fase di rallentamento tuttavia notevolmente ridotta in termini di epoche rispetto le prove precedenti.

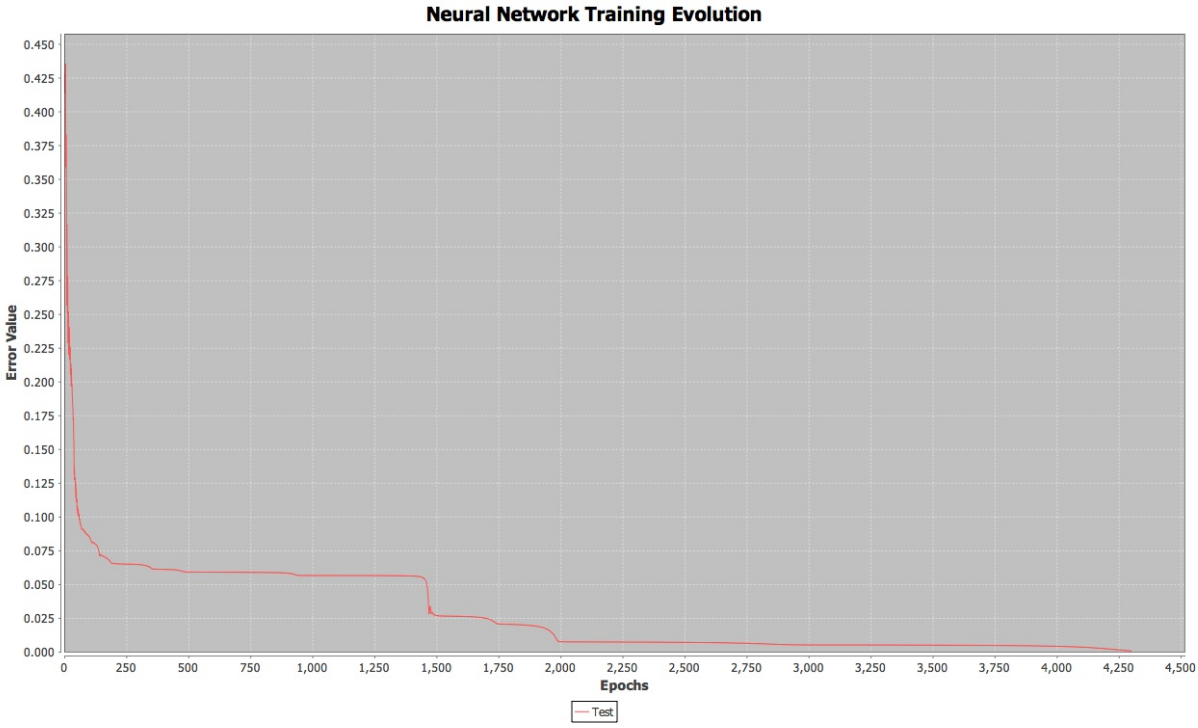

Fig. 12 - Grafico di andamento dell'errore durante l'addestramento

Dalle prove precedenti, potrebbe sembrare conveniente verificare vi sia la possibilità di ottenere risultati ancor più efficienti scegliendo un valore di coefficiente di apprendimento compreso tra 0.001 e 0.05, prova successiva è stata dunque quella di sceglier un valore di coefficiente di apprendimento pari a 0.03.

Dalla figura 12 è possibile osservare che, pur rimanendo relativamente basso, il numero di epoche necessario al fine di ottenere un valore di errore inferiore a 0.001 aumenta rispetto il passo precedente, ciò permette di innalzare l'estremo inferiore dell'intervallo definito a 0.03, escluso.

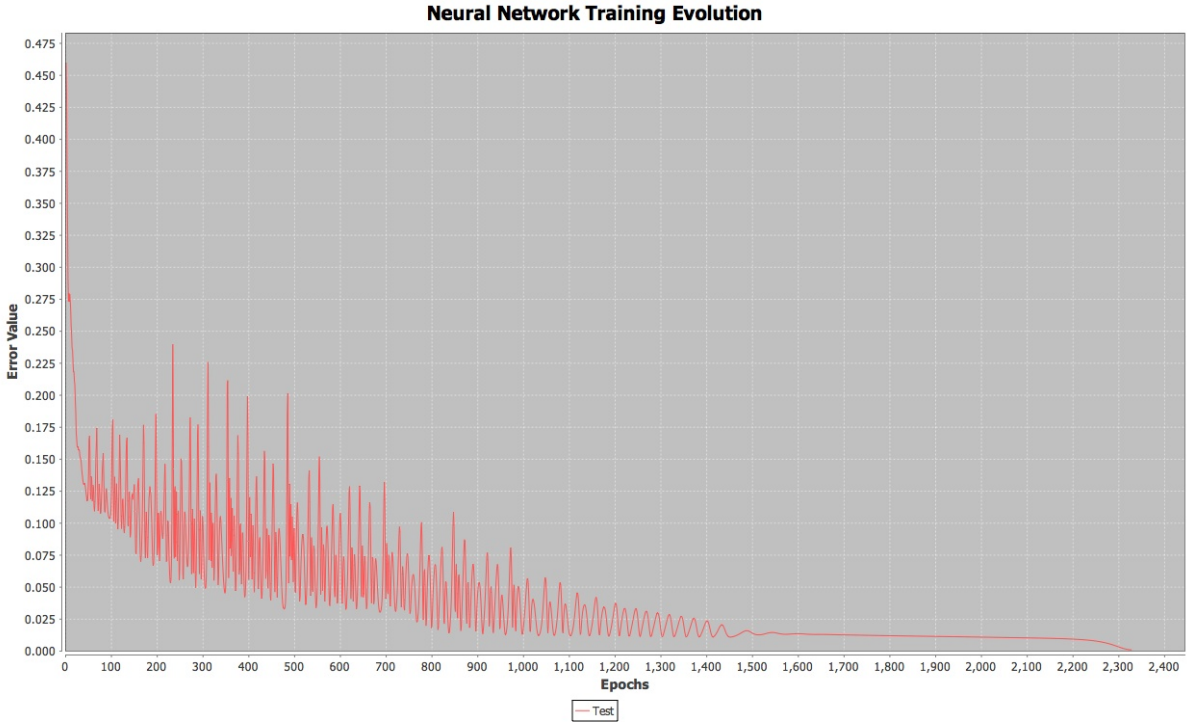

Fig. 13 - Grafico di andamento dell'errore durante l'addestramento

L'unico valore ancora da verificare nell'intervallo stabilito è dunque di 0.04 per il coefficiente di apprendimento, dalla figura 13 è possibile osservare come questo risulti effettivamente migliorare ancor più le prestazioni del sistema richiedendo circa 2.350 epoche per ottenere un valore di errore inferiore a 0.001.

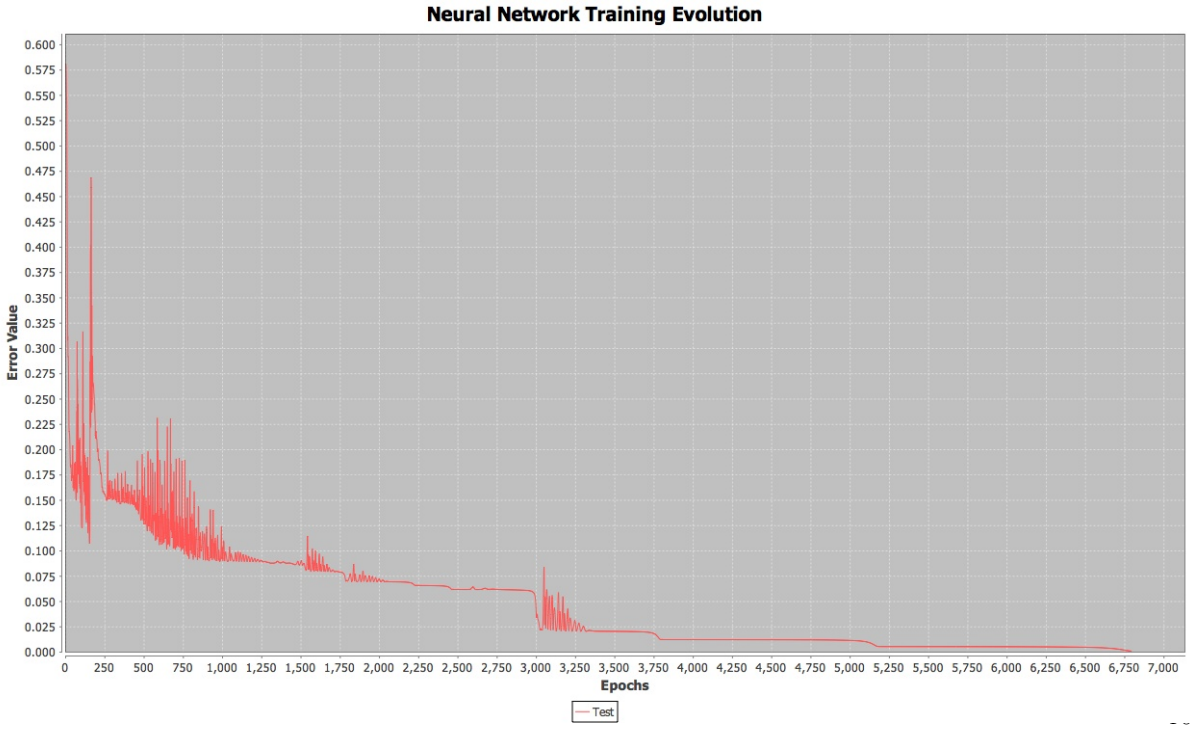

Fig. 14 - Grafico di andamento dell'errore durante l'addestramento

Dalla figura 13 è inoltre possibile osservare come il comportamento descritto relativamente a figura 11 sia in questo caso ancor più accentuato, l'intervallo di "rimbalzo" diviene infatti più ampio dell'intervallo in cui la funzione tende a rallentare visibilmente e ciò a mio parere è dimostrazione del motivo per cui tale impostazione della Rete Neurale risulti esser la più efficiente.

Ottenuta un'architettura efficiente considerando valori di apprendimento compresi tra 0.001 e 0.05, potrebbe esser possibile ottenere prestazioni migliori scegliendo un valore di coefficiente di apprendimento più grande dell'estremo superiore dell'intervallo suddetto.

Nella figura 14 è illustrato l'andamento del valore di errore ottenuto dal sistema con coefficiente di apprendimento 0.06, il numero di epoche necessarie cresce e l'andamento della funzione torna a presentare diffusi intervalli di rallentamento dell'apprendimento, ciò lascia dunque pensare che l'architettura ottimale continui ad esser con coefficiente di apprendimento 0.03.

• *Librerie utilizzate*

1. *Encog Machine Learning Framework*

L'implementazione dell'architettura della rete neurale, della fase di apprendimento supervisionato e della fase finale di validazione fa uso del framework "Encog" di Jeff Heaton (http://www.heatonresearch.com/about.html).

Tale scelta è dovuta non solo al fatto che il linguaggio utilizzato sia il Java, ma anche alla notevole semplicità di utilizzo del framework oltre che alla chiara ed estesa documentazione disponibile.

2. *JFree Chart Framework*

Per la produzione del grafico di andamento dell'errore in fase d'apprendimento, è stata utilizzata la libreria Open Source "JFreeChart" (http://www.jfree.org/jfreechart/), anche in questo caso la scelta è stata guidata dalla facilità d'utilizzo della libreria e dalla ampia documentazione a disposizione.

3. *Apache Commons I/O*

Nella fase di recupero delle immagini da utilizzare come samples per la fase di addestramento supervisionato della rete si è fatto uso della libreria "Apache Commons I/O" per filtrare i soli file visibili nella cartella "DataSet/Normal".