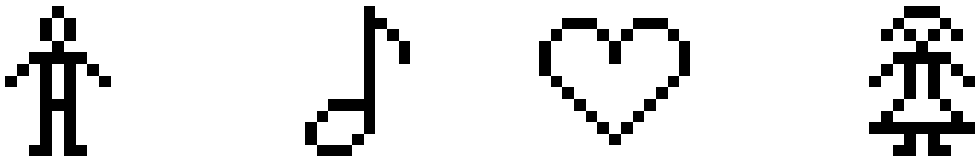
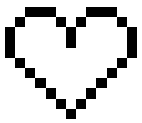
RETE NEURALE MULTISTRATO ED APPRENDIMENTO DI FORME MEDIANTE L'ALGORITMO DI BACKPROPAGATION

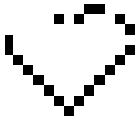
Mario Cimino e Giovanni D'Alessandro

Pisa, 2001

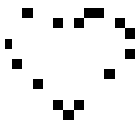




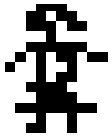
L'immagine somiglia ad un(a) cuore



L'immagine somiglia ad un(a) cuore



L'immagine sembra vagamente un(a) cuore
E' una nuova forma (s/n)?
n



L'immagine sembra vagamente un(a) donna
E' una nuova forma (s/n)?
n

Indice

1. ANALISI DEL PROBLEMA E OBIETTIVI DI PROGETTO

1.1 Il dominio del sistema

Desideriamo un'applicazione che permetta di creare facilmente una rete di neuroni multistrato, della quale si possa indicare il numero di strati, il numero di ingressi della rete, il numero di neuroni di ogni strato. Che l'addestramento – mediante algoritmo di backpropagation – possa configurarsi mediante il numero di esempi di un'epoca, l'errore minimo, il massimo numero di epoche da presentare. Gli ingressi siano delle immagini bianco e nero, possano essere forniti mediante un formato grafico portabile, disegnati con l'ausilio di un'applicazione grafica, e si possa variare le dimensioni dell'immagine a proprio piacimento; vogliamo inoltre che gli ingressi possano essere dati anche come vettori binari in un file di testo, di qualsiasi dimensione, così da generalizzare al massimo l'uso della rete. Le uscite della rete siano dei nomi, associati ad ogni esempio del Training Set, di cui si possa dare anche il massimo numero di caratteri. Vogliamo infine che - in caso di una immagine molto diversa da quelle apprese - il sistema se ne accorga e possa, su conferma dell'utente , automaticamente inserirla nel suo insieme di apprendimento.

1.2 Il processo di sviluppo

Vogliamo un codice c++ orientato agli oggetti, in cui ogni entita' a se' sia raggruppata in una classe, incapsulata in un modulo, e realizzata in modo da garantire una riusabilita' del codice, una buona leggibilita' e modificabilita'. Daremo quindi importanza al processo di sviluppo piuttosto che al prodotto, laddove i criteri suddetti potessero produrre un lieve peggioramento della velocita' di esecuzione.

A tale proposito cercheremo, se possibile, di usare le funzioni inline, tenendo presente che "Scrivere la definizione di una funzione membro nella dichiarazione di una classe rende i programmi meno leggibili e meno modificabili." *(pag. 215,* [Domenici-Frosini, 1996] *);* oppure il passaggio dei parametri di ingresso per riferimento (mediante attributo const), per una chiamata di funzione piu' snella.

Vogliamo che il codice sia portabile su tutte le piattaforme (WINDOWS, MAC, OS/2 …), quindi eviteremo l'uso di librerie specifiche di sistema (esempio per il Bitmap di Windows, o per operazioni sul file system), creando invece delle classi o procedure opportune, a meno che queste non risultino troppo grandi rispetto alle strutture proprie della rete multistrato.

2. PROGETTAZIONE

2.1 Identificazione degli oggetti

Si vuole utilizzare la classe Neurone creata nella prima esercitazione, ma con un vettore di pesi di dimensione variabile a tempo di esecuzione, poiche' il numero di ingressi e' deciso dall' utente; sebbene questo comporti l'uso dei puntatori, non se ne avra' traccia sulle dichiarazioni delle funzioni pubbliche, i quanto verranno usati solo i riferimenti, di piu' facile leggibilita'.

Nell'algoritmo di backpropagation, molte operazioni coinvolgono vettori; per cui possiamo pensare ad un concetto "Vettore" come idea separata, quindi rendendolo con una classe; lo stesso algoritmo inoltre presenta dei cicli di operazioni eseguite strato per strato, per cui e' necessario anche il concetto di "Strato" come insieme di "Neuroni"; infine, quasi in maniera naturale, la classe "Rete" esprime il concetto di insieme di "Strati".

2.2 Identificazione dei servizi

I servizi offerti da ogni classe sono stati estratti dalla descrizione dell'algoritmo di backpropagation, quindi operazioni varie sui vettori, oppure funzioni di manipolazione della Rete, implementate spesso con un ciclo di funzioni delle classi via via inferiori delle quali e' composta la struttura.

Ad esempio, la funzione "attivazione" della Rete provvede ad attivare tutti gli strati di cui e' composta; la funzione omonima dello strato attivera' ogni suo neurone, ed infine l' "attivazione" di un neurone comportera' la somma pesata degli ingressi, realizzata a sua volta con un prodotto scalare tra Vettori, memorizzato nello stato di attivazione.

Similmente avviene con le funzioni "carica da File", "salva su File", o "PesiCasuali" o "PesiZero", "Stampa", di evidente significato, in cui ogni operazione ad un livello e' un' iterazione di operazioni sulla classe sottostante, fino alla classe vettore, che esegue la funzione vera e propria , tipicamente sul peso di una connessione.

Le funzioni "Uscita" calcolano la funzione uscita dell'oggetto, che a livello di neurone e' una sigmoide; l'uscita di uno strato sara' un "Vettore" di "Scalari" ognuno dei quali e' l'uscita del neurone corrispondente.

La funzione "Osserva" di una Rete, legge il vettore di scalari dell'uscita della rete e ne approssima gli elementi a zero od uno; in particolare, valori sopra 0.9 e sotto 0.1 , indicanti un livello di rumore accettabile, vengono rispettivamente portati ad 1 e 0;

valori intermedi vengono approssimati all'intero piu' vicino, ma la funzione restituisce "false" indicando un riconoscimento parziale.

Tale vettore binario verra' convertito, tramite la funzione "intero", in un numero intero, ed il suo valore costituira' l'indice di una tabella delle uscite, contenente le stringhe di risposta.

La "apprendiEsempio" esegue un passo di apprendimento, relativo ad un singolo esempio, del backpropagation; "aggiornaErrore" somma l'errore sul singolo Esempio all'errore globale dell'epoca; "backpropagation" esegue tante volte le epoche, provvedendo , mediande la "permuta" a permutare in ordine casuale gli esempi (tale permutazione di indici e' stata realizzata, con complessita' lineare, mediante scambi di posto tra un indice iterato su tutto il vettore ed un altro scelto a caso).

I nomi usati dovrebbero essere abbastanza esplicativi, e la struttura delle classi si puo' vedere nel diagramma della classi a pagina seguente, scritto in Unified Modeling Language (UML), la notazione usata dalle principali aziende di software mondiali.

E' inevitabile che sin da adesso i servizi offerti dalle classi contengano dei dettagli implementativi, poiche' non abbiamo rappresentato le varie fasi intermedie di sviluppo. Di queste funzioni parleremo nel prossimo paragrafo.

3. SVILUPPO DEL CODICE IN C++

3.1 Il problema degli array di oggetti classe

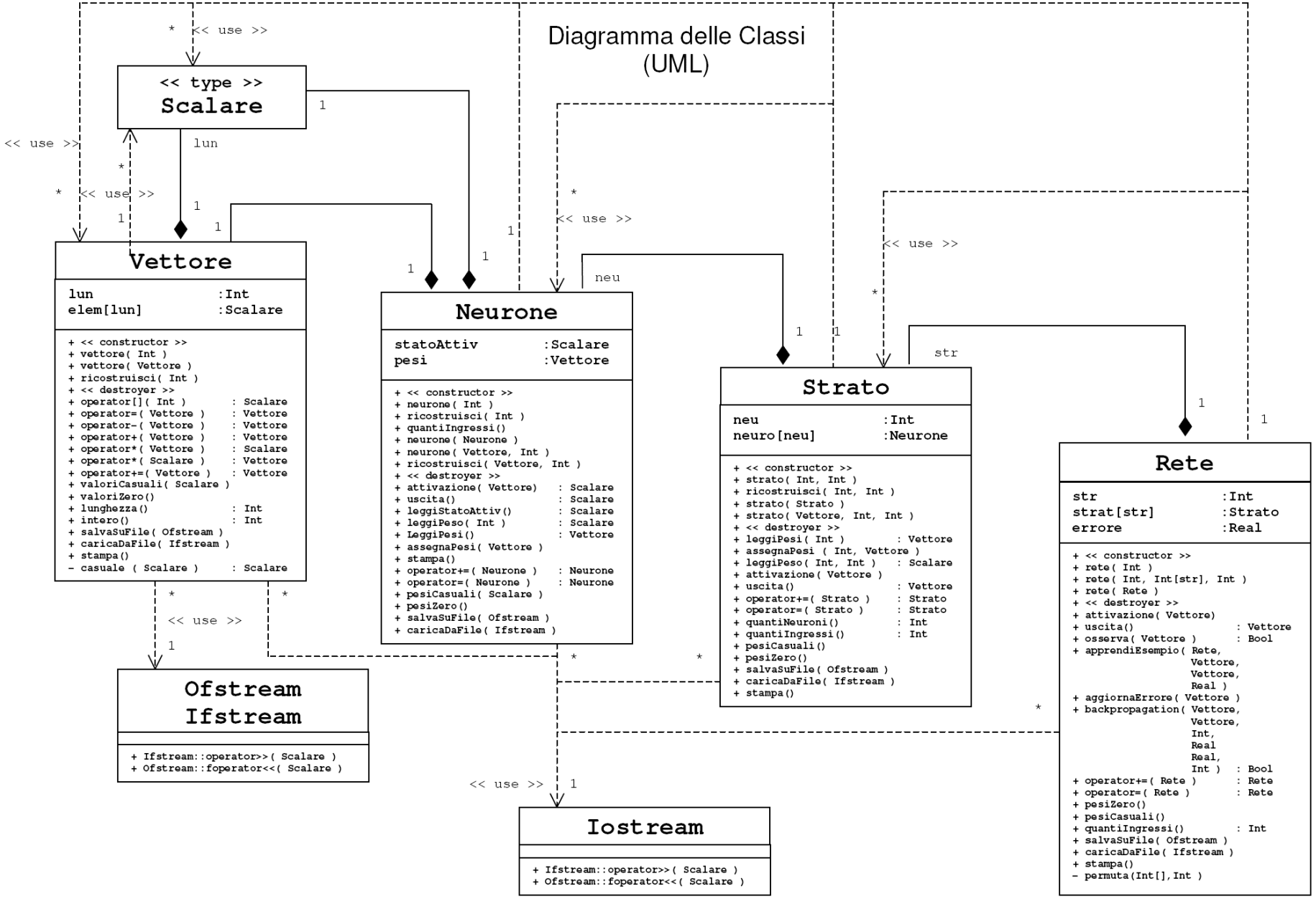
Proseguiamo descrivendo le funzioni membro legate all' implementazione.

Il C++ non permette l'uso di costruttori diversi da quello di default per array di oggetti classe. Se ad esempio si scrive " *miooggetto miaclasse[10]; ",* verra' eseguito il costruttore standard miaclasse(); non e' permesso *"miooggetto miaclasse(n,dim)[10];".*

La soluzione da noi proposta consiste nel dichiarare l'array con il costruttore standard e di ricostruire subito dopo l'oggetto con una funzione "ricostruisci", realizzata ovviamente per tutte le classi via via sottostanti.

D'altra parte, poiche' gli oggetti Vettore usati sono spesso di dimensioni variabili, ad ogni passo di un ciclo di apprendimento dell'algoritmo di B.P., in quanto variano le dimensioni dello strato, tale funzione puo' essere usata per modificare un "Vettore" - dichiarato in testa al ciclo - nelle dimensioni, a seconda delle necessita'; come - ad esempio - si modifica una variabile intero (i=val1; i=val2; …) , così sara' possibile modificare un oggetto vettore nelle dimensioni (vett(dim1); vett.ricostruisci(dim2); …)

Diagramma delle Classi (UML)



3.2 I costruttori di copia

Poiche' la parte dati di tutti gli oggetti contiene dei puntatori a memoria libera, e' stato necessario ridefinire i costruttori di copia.

3.3 Il problema della memorizzazione dei deltapesi

Nell'algoritmo di B.P., l'aggiornamento dei pesi avviene dopo il calcolo delle variazioni degli stessi su tutta la rete. Sorge quindi il problema di memorizzare queste informazioni in una struttura dati molto irregolare nelle dimensioni (e' un vettore di matrici, ciascuna di diversa altezza, e diversa larghezza) e costruire le opportune funzioni per aggiornare i pesi della rete.

Seguendo una la logica object oriented, verrebbe spontaneo separare la parte "pesi" dalla classe Neurone, e farne una classe base "PesiNeurone", che sara' ereditata dalla rimanente parte di "Neurone", e creare poi le opportune funzioni, senza nulla cambiare a livello "Strato" e "Rete". Queste funzioni pero' sarebbero utilizzate da due nuove classi "PesiStrato" e "Pesi Rete", da definire. In pratica, una tale soluzione implica l'aggiunta di tre nuove classi. E' giustificato un tale aumento di complessita' ?

D'altra parte, osserviamo che i pesi costituiscono la parte dati principale della classe (i ogni Neurone c'e' in piu' solo la variabile intera Stato di Attivazione"), e che nell'applicazione vi e' un solo oggetto Rete.

Abbiamo considerato inefficiente creare una classe apposita, scegliendo di memorizzare i pesi in una rete gemella, dove la variabile attivazione rimane inutilizzata. L'operatore += (implementato – come sempre – in tutte le classi sottostanti) tra le due reti realizzera' in maniera molto naturale l'aggiornamento dei pesi. La parte dei servizi sara' in comune tra i due oggetti, non producendo alcun overhead.

3.4 la suddivisione in moduli e le funzioni di ingresso-uscita

Seguendo i principi di incapsulamento, ogni classe ha un suo modulo ed una sua interfaccia.

Una serie di funzioni di ingresso-uscita, per acedere ai file, caricare strutture dati, parametri, sono state raggruppate in uno modulo "inout" usato dal programma principale. Nella pagina seguente rappresentiamo il Diagramma dei Moduli, in UML, dove compaiono anche i servizi offerti dal modulo inout, e l'interazione dell'utente con l'applicazione.

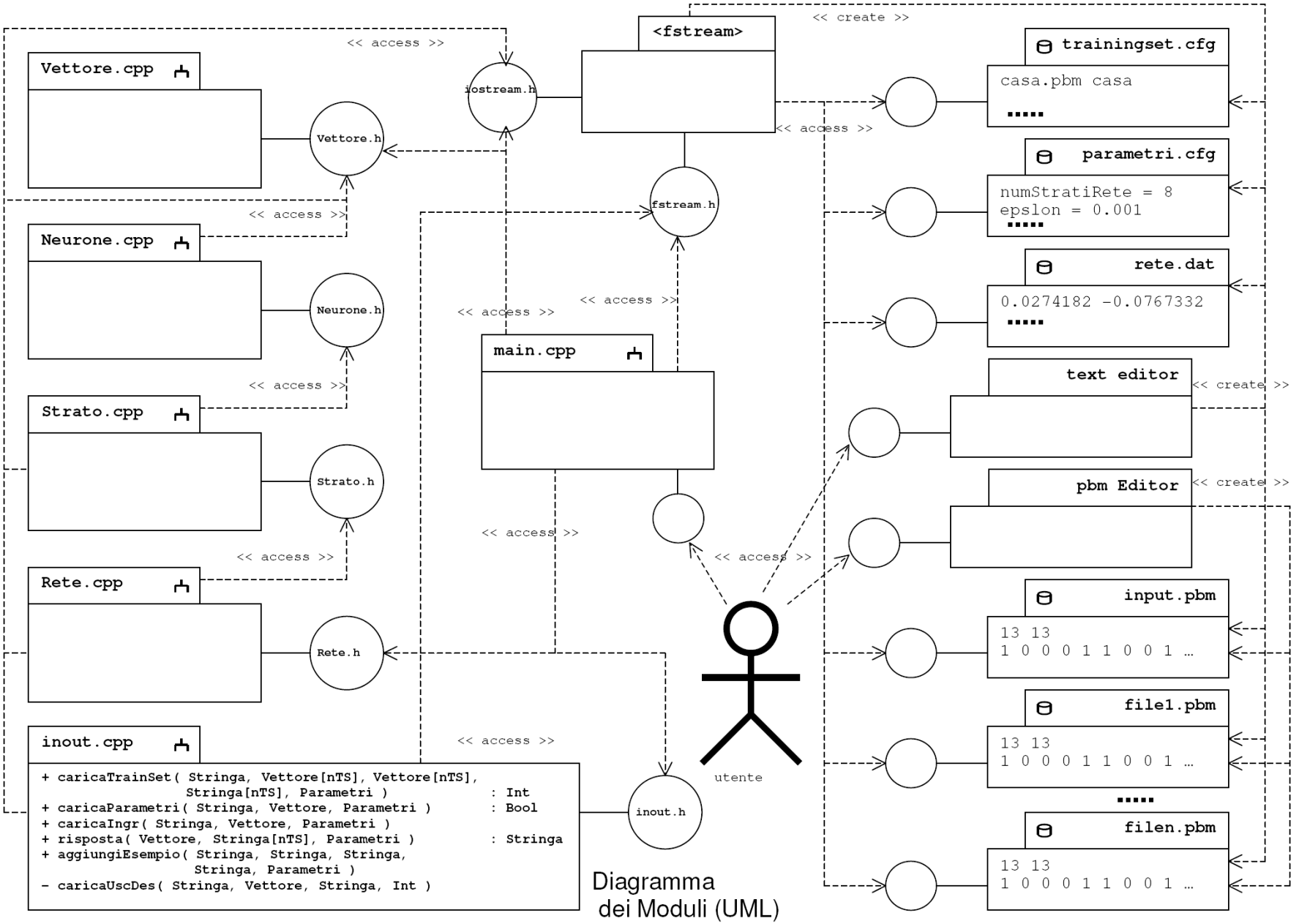


Diagramma dei Moduli (UML)

Dati M esempi del Training Set, la rete dovra' avere un numero di uscite pari almeno al logaritmo binario di M, per identificare ogni configurazione.

La i-esima immagine di esempio, sara' trasformata in un "Vettore" di ingressi, e questi caricato in ingrDes[i], mentre su risp[i] verra' caricata la stringa di risposta corrispondente; infine, su uscDes[i] viene caricato il "Vettore" che , in binario, rappresenta "i in base due" e che sara' l'uscita desiderata della rete.

La "caricaTrainset" costruisce, a partire dai file.pbm, le tre strutture suddette, utilizzando le funzioni casicaUscDes e caricaIngr, di ovvio significato.

La "caricaParametri" carica tutti i parametri stabiliti dall'utente in una struttura "parametri" passata per riferimento alle altre funzioni (cio' evita di passare troppi parametri, con facile leggibilita' delle dichiarazioni).

3.5 Le risposte "sembra vagamente a…" e "…non so cosa"

La funzione "risposta" ritorna la stringa di risposta i-esima, per un dato vettore di uscita; se questa e' ben identificata da "osserva", viene data all'utente come "somiglia a…".

Se il rumore e' eccessivo, e quindi la "osserva" trova qualche bit di uscita compreso tra 0.9 e 0.1, viene approssimata una risposta usando un arrotondamento al bit piu' vicino; cio' viene espresso come "sembra vagamente a…".

Infine, se tale arrotondamento produce un intero al di fuori dal rango del training set, viene espresso un "non so cosa".

In tal modo, abbiamo voluto dare all'utente un'idea della "distanza" dell' immagine con rumore, rispetto a quelle apprese.

3.6 L'apprendimento di una forma sconosciuta

In questi ultimi due casi, di riconoscimento incerto, il sistema chiede all'utente di poter apprendere la nuova forma. In caso di affermativo, entra nei file di configurazione (.cfg) ed incrementa il parametro "numero di esempi del training Set", aggiungendo alla lista degli esempi la nuova forma.

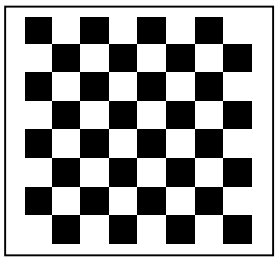
Per semplicita' non e' stato incrementato il numero di uscite nel caso in cui i bit fossero insufficienti, e cosi pure il numero di neuroni o di strati, cosa quest'ultima alquanto complessa. L'utente quindi dovra' riguardare il file di configurazione e decidere se aumentare o meno tali valori.

3.7 L'interfaccia utente ed il formato PBM (Portable Bitmap)

Il formato "Bitmap per OS/2 Windows" (".bmp" per intenderci) e' in realta' compresso, e necessita di librerie per la decodifica.

Il formato "Portable Bitmap" (".pbm") , in modalita' "Ascii Encoded" (esiste anche la "Binary Encoded") e' un file di caratteri ascii formato da un breve preambolo ed una sequenza di "0" ed "1", nel caso di immagini B/N, corrispondenti al pixel bianco e nero rispettivamente. L'ultima parte del preambolo e' costituita da due numeri indicanti altezza e larghezza. Praticamente qualsiasi sistema operativo al mondo, e qualsiasi text editor e' in grado di leggerlo.

Ad esempio un immagine 8x8 così fatta:



viene salvata come:

```
P1
# Created by Paint Shop Pro
8 8
1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 1 0 1 0
1 0 1 0 0 1 0 1 0 1 0 1 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1
```

In tal modo possiamo usare un editor grafico di pbm per disegnare le immagini col mouse, ritoccarle, oppure dare alla rete dei vettori, inseriti con un text-editor o con un semplice modulo in c++ che legge dati per esempio da un'applicazione medica.

L'assenza di compressione e' importante anche perche', nel caso del riconoscimento delle immagini, le compressioni devono essere fatte opportunamente, in maniera da conservare certe proprieta' topologiche della figura.

Le compressioni "gif" o "bmp" sono assolutamente inadatte in tal senso - pur essendo diffuse su Internet - perche' minimizzano l'occupazione di memoria su disco, ignorando la topologia dell'oggetto.

Invece le compressioni "per sintesi" (che memorizzano un'immagine come composizione di linee, triangoli, cerchi, etc…) sono buone per immagni geometriche, perche' risolvono il problema che la stessa immagine, ingrandita o rimpicciolita, o traslata, non verrebbe riconosciuta dalla nostra rete.

D’altronde, per immagini campionate, ad esempio foto di paesaggi naturali, occorrono compressioni molto piu’ sofisticate, ad esempio, dividendo l’immagine in “blocchetti piccoli”, e sintetizzando per ogni blocco la sola informazione cui e’ sensibile l’occhio umano; ad esempio, la “jpg” - che memorizza i coefficienti della trasformata di Fourier discreta del blocchetto – potrebbe essere opportunamente adattata; e per essere tollerante a movimenti e traslazioni dei soggetti, deve basarsi anche sulla interpretazone tridimensionale delle immagini piane.

Abbiamo voluto spendere qualche parola in piu’ su questo argomento, poiche’ lo riteniamo di importanza fondamentale per risolvere il problema del riconoscimento di dati percepiti dai nostri sensi, con l’uso delle reti neurali; ed anche per esprimere i limiti della nostra applicazione, che sarebbe assolutamente inadatta per immagini di grosse dimensioni, senza un opportuno modulo di compressione (o - meglio – di “decodifica”).

3.7 Listato di tutto il codice

```
////////////////////////////////////////////////////////////////////////////
// vettore.h

#ifndef VETTORE_H
#define VETTORE_H

typedef double scalare;
class vettore
{
private:
    int         lun;
    scalare*    elem;
    scalare     casuale( const scalare& max );
public:
    inline      vettore();
                vettore( int lu );
                vettore( const vettore& vet );
    void        ricostruisci( int lu );
    inline      ~vettore();
    scalare&    operator[]( int i ) const;
    scalare&    operator[]( int i );
    vettore&    operator=( const vettore& vet );
    vettore     operator-( const vettore& vet );
    vettore     operator+( const vettore& vet );
    scalare     operator*( const vettore& vet );
    vettore     operator*( const scalare& sca ) const;
    vettore&    operator+=( const vettore& vet );
    void        valoriCasuali( const scalare& max );
    void        valoriZero();
    inline int  lunghezza() const;
    int         intero() const;
    void        salvaSuFile( ofstream& out );
    void        caricaDaFile( ifstream& in );
    void        stampa();
};

inline vettore::vettore()
{   lun = 0; elem = 0; }

inline vettore::~vettore()
{   delete[] elem;  }

inline int vettore::lunghezza() const
{   return lun; }


#endif
```

```
////////////////////////////////////////////////////////////////////////////
// vettore.cpp

#include <fstream.h>
#include <stdlib.h>
#include <math.h>
#include "vettore.h"

scalare vettore::casuale( const scalare& max )
{   return max * ( 2 * scalare ( rand() ) / scalare ( RAND_MAX ) - 1);
}

vettore::vettore( int lu )
{   lun = lu;
    elem = new scalare[lun];
}

void vettore::ricostruisci( int lu )
{   if ( elem != 0 )
    {   lun = 0;
        delete[] elem;
    }
    lun = lu;
    elem = new scalare[lun];
}

vettore::vettore( const vettore& vet )
{   lun = vet.lun;
    elem = new scalare[lun];
    for ( int i = 0; i < lun; i++ )
        elem[i] = vet.elem[i];
}

scalare& vettore::operator[]( int i ) const
{   if ( i < 0  ||  i >= lun )
    {   cerr << "indice fuori intervallo." << endl;
        exit(1);
    }
    else
        return elem[i];
}

scalare& vettore::operator[]( int i )
{   if ( i < 0 || i >= lun )
    {   cerr << "indice fuori intervallo." << endl;
        exit(1);
    }
    else
        return elem[i];
}

vettore& vettore::operator=( const vettore& vet )
{   if ( this == &vet )
        return *this;
    else
        if ( lun != vet.lun )
        {   cerr << "vettori di diverse dimensioni." << endl;
            exit(2);
        }
        else
            for ( int i = 0; i < lun; i++ )
                elem[i] = vet.elem[i];
    return *this;
```

```
}

vettore vettore::operator-( const vettore& vet )
{   vettore result( lun );
    if ( lun != vet.lun )
    {   cerr << "vettori di diverse dimensioni." << endl;
        exit(2);
    }
    else
    {   for ( int i = 0; i < lun; i++ )
            result.elem[i] = elem[i] - vet.elem[i];
        return result;
    }
}

vettore vettore::operator+( const vettore& vet )
{   vettore result( lun );
    if ( lun != vet.lun )
    {   cerr << "vettori di diverse dimensioni." << endl;
        exit(2);
    }
    else
    {   for ( int i = 0; i < lun; i++ )
            result.elem[i] = elem[i] + vet.elem[i];
        return result;
    }
}

scalare vettore::operator*( const vettore& vet )
{   if ( lun != vet.lun )
    {   cerr << "vettori di diverse dimensioni." << endl;
        exit(2);
    }
    else
    {   scalare result = 0;
        for ( int i = 0; i < lun; i++ )
            result += elem[i] * vet.elem[i];
        return result;
    }
}

vettore  vettore::operator*( const scalare& sca ) const
{   vettore result( *this );
    for ( int i = 0; i < lun; i++ )
        result.elem[i] = elem[i] * sca;
    return result;
}

vettore& vettore::operator+=( const vettore& vet )
{   if (lun != vet.lun)
    {   cerr << "vettori di diverse dimensioni." << endl;
        exit(2);
    }
    else
    {   for ( int i = 0; i < lun; i++ )
                elem[i] += vet.elem[i];
        return *this;
    }
}

void vettore::valoriCasuali( const scalare& max )
{   for ( int i = 0; i < lun; i++ )
        elem[i] = casuale( max );
```

```
}

void vettore::valoriZero()
{   for ( int i = 0; i < lun; i++ )
        elem[i] = 0;
}

int vettore::intero() const
{   int result = 0;
    for (int i = lun - 1; i >= 0; i-- )
        result += int( ldexp( int( elem[i] ), i ) );// elem[i]*2^i
    return result;
}

void vettore::salvaSuFile( ofstream& out )
{   if ( !out )
    {   cerr << "errore di apertura file" << endl;
        exit(7);
    }
    for ( int i = 0; i < lun; i++ )
        out << elem[i] << ' ';
}

void vettore::caricaDaFile( ifstream& in )
{   if ( !in )
    {   cerr << "errore di apertura file" << endl;
        exit(7);
    }
    for ( int i = 0; i < lun; i++ )
        in >> elem[i];
}

void vettore::stampa()
{   cout << "[";
    for ( int i = 0; i < lun; i++ )
        cout << elem[i] << ',';
    cout << "]";
}
```

```
////////////////////////////////////////////////////////////////////////////
//neurone.h

#ifndef NEURONE_H
#define NEURONE_H

#include "vettore.h"

class neurone
{
private:
    scalare         statoAttiv;
    vettore*        pesi;
public:
    inline          neurone();
                    neurone( int in );
    void            ricostruisci( int in );
    int             quantiIngressi() const;
                    neurone( const neurone& neu );
                    neurone( const vettore& pesiIniz, int in );
    void            ricostruisci( const vettore& pesiIniz, int in );
    inline          ~neurone();
    scalare         attivazione ( const vettore& ingr );
    scalare         uscita();
    inline scalare  leggiStatoAttiv();
    scalare         leggiPeso( int i );
    vettore         leggiPesi();
    void            assegnaPesi( const vettore& p );
    void            stampa();
    neurone&        operator+=( const neurone& neur );
    neurone&        operator=( const neurone& neur );
    void            pesiCasuali( const scalare& max );
    void            pesiZero();
    void            salvaSuFile( ofstream& out );
    void            caricaDaFile( ifstream& in );

};

inline neurone::neurone()
{   statoAttiv = 0; pesi = 0;   }

inline neurone::~neurone()
{   delete pesi;}

inline scalare neurone::leggiStatoAttiv()
{   return statoAttiv;  }


#endif
```

```
////////////////////////////////////////////////////////////////////////////
//neurone.cpp

#include <fstream.h>
#include <stdlib.h>
#include <math.h>
#include "neurone.h"

neurone::neurone( int in )
{   statoAttiv = 0;
    pesi = new vettore( in );
}

void neurone::ricostruisci( int in )
{   if ( pesi == 0 )
        pesi = new vettore( in );
    else
    {   cerr << "tentativo di ricostruire un neurone non vuoto." << endl;
        exit(3);
    }
}

int neurone::quantiIngressi() const
{   return (*pesi).lunghezza();
}

neurone::neurone( const neurone& neu )
{   statoAttiv = neu.statoAttiv;
    pesi = new vettore( neu.quantiIngressi() );
    (*pesi) = *(neu.pesi);
}

neurone::neurone( const vettore& pesiIniz, int in )
{   statoAttiv = 0;
    pesi = new vettore( in );
    (*pesi) = pesiIniz;
}

void neurone::ricostruisci( const vettore& pesiIniz, int in )
{   if ( pesi == 0 )
    {   pesi = new vettore( in );
        (*pesi) = pesiIniz;
    }
    else
    {   cerr << "tentativo di ricostruire un neurone non vuoto." << endl;
        exit(3);
    }
}

scalare neurone::attivazione( const vettore& ingr )
{   statoAttiv = (*pesi) * ingr ;           // prodotto scalare
    return statoAttiv;
}

scalare neurone::uscita()
{   return 1 / ( 1 + exp( - statoAttiv ) );
}

scalare neurone::leggiPeso( int i )
{   return (*pesi)[i];
}

vettore neurone::leggiPesi()
```

```
{   return (*pesi);
}

void neurone::assegnaPesi( const vettore& p )
{    (*pesi) = p ;
}

void neurone::stampa()
{   cout << "<" << statoAttiv << ">";
    (*pesi).stampa();
}

neurone& neurone::operator+=( const neurone& neur )
{   (*pesi) += *(neur.pesi);
    return *this;
}

neurone& neurone::operator=( const neurone& neur )
{   if ( this == &neur )
        return *this;
    else
        (*pesi) = *(neur.pesi);
    return *this;
}

void neurone::pesiCasuali( const scalare& max )
{   (*pesi).valoriCasuali( max );
}

void neurone::pesiZero()
{   (*pesi).valoriZero();
}

void neurone::salvaSuFile( ofstream& out )
{   (*pesi).salvaSuFile( out );
}

void neurone::caricaDaFile( ifstream& in )
{   (*pesi).caricaDaFile( in );
}
```

```
////////////////////////////////////////////////////////////////////////////
//strato.h

#ifndef STRATO_H
#define STRATO_H

#include "neurone.h"

class strato
{
private:
    int         neu;
    neurone*    neuro;
public:
    inline      strato();
                strato( int in, int ne );
    void        ricostruisci( int in, int ne );
                strato( const strato& str );
                strato( const vettore pesi[], int in, int ne );
    inline      ~strato();
    vettore     leggiPesi( int n );
    void        assegnaPesi( int n, const vettore& p );
    scalare     leggiPeso( int n, int j);
    void        attivazione( const vettore& ingr );
    vettore     uscita();
    strato&     operator+=( const strato& str );
    strato&     operator=( const strato& str );
    int         quantiNeuroni();
    inline int  quantiIngressi();
    void        pesiCasuali();
    void        pesiZero();
    void        salvaSuFile( ofstream& out );
    void        caricaDaFile( ifstream& in );
    void        stampa();
};


inline strato::strato()
{   neu = 0; neuro = 0; }

inline strato::~strato()
{   delete[] neuro; }

inline int strato::quantiNeuroni()
{   return neu; }

#endif
```

```
////////////////////////////////////////////////////////////////////////////
//strato.cpp

#include <fstream.h>
#include <stdlib.h>
#include <math.h>
#include "strato.h"

strato::strato( int in, int ne )
{   neu = ne;
    neuro = new neurone[neu];
    for ( int n = 0; n < neu; n++ )
        neuro[n].ricostruisci( in );
}

void strato::ricostruisci( int in, int ne )
{   if ( neuro == 0 )
    {   neu = ne;
        neuro = new neurone[neu];
        for ( int n = 0; n < neu; n++ )
            neuro[n].ricostruisci( in );
    }
    else
    {   cerr << "tentativo di ricostruire uno strato non vuoto." << endl;
        exit(4);
    }
}

strato::strato( const strato& str )
{   neu = str.neu;
    neuro = new neurone[neu];
    for ( int n = 0; n < neu; n++ )
        neuro[n].ricostruisci( quantiIngressi() );
    (*neuro) = *(str.neuro);
}

strato::strato( const vettore pesi[], int in, int ne )
{   neu = ne;
    neuro = new neurone[neu];
    for ( int n = 0; n < neu; n++ )
        neuro[n].ricostruisci( pesi[n], in );
}


vettore  strato::leggiPesi( int n )
{   return neuro[n].leggiPesi();
}


void strato::assegnaPesi( int n, const vettore& p )
{   neuro[n].assegnaPesi( p );
}

scalare  strato::leggiPeso( int n, int j)
{   return neuro[n].leggiPeso( j );
}

void strato::attivazione( const vettore& ingr )
{   for ( int n = 0; n < neu; n++ )
        neuro[n].attivazione( ingr );
}

vettore strato::uscita()
```

```
{   vettore result( neu );
    for ( int n = 0; n < neu; n++ )
        result[n] = neuro[n].uscita();
    return result;
}

strato& strato::operator+=( const strato& str )
{   for ( int n = 0; n < neu; n++ )
        neuro[n] += str.neuro[n];
    return *this;
}

strato& strato::operator=( const strato& str )
{   if ( this == &str )
        return *this;
    else
        for ( int n = 0; n < neu; n++ )
            neuro[n] = str.neuro[n];
    return *this;
}

int strato::quantiIngressi()
{   return neuro[0].quantiIngressi();
}

void strato::pesiCasuali()
{   scalare max = 1 / sqrt( scalare ( quantiIngressi() ) );
    for ( int n = 0; n < neu; n++ )
        neuro[n].pesiCasuali( max );
}


void strato::pesiZero()
{   for ( int n = 0; n < neu; n++ )
        neuro[n].pesiZero();
}

void strato::salvaSuFile( ofstream& out )
{   for ( int n = 0; n < neu; n++ )
        neuro[n].salvaSuFile( out );
}

void strato::caricaDaFile( ifstream& in )
{   for ( int n = 0; n < neu; n++ )
        neuro[n].caricaDaFile( in );
}

void strato::stampa()
{   for ( int n = 0; n < neu; n++ )
    {   cout << '{';
        neuro[n].stampa();
        cout << "(neurone " << n << ")" << '}';
    }
}
```

```
////////////////////////////////////////////////////////////////////////////
//rete.h

#ifndef RETE_H
#define RETE_H

#include "strato.h"

class rete
{
private:
    int         str;
    strato*     strat;
    double      errore;
public:
    inline      rete();
                rete( int st );
                rete( int st, const int ne[], int in );

                rete( const rete& ret );
    inline      ~rete();
    void        attivazione( const vettore& ingr );
    vettore     uscita();
    bool        osserva(vettore& risuVett);
    void        apprendiEsempio( rete& deltaRete,
                                 const vettore& ingrDes,
                                 const vettore& uscDes,
                                 const double eta );
    void        aggiornaErrore( const vettore& uscDes );
    bool        backpropagation( const vettore  ingrDes[],
                                 const vettore  uscDes[],
                                 const int numTrainSet,
                                 const double epslon,
                                 const double etaMax,
                                 const int maxTent );
    rete&       operator+=( const rete& delta );
    rete&       operator=( const rete& delta );
    void        pesiZero();
    void        pesiCasuali();
    int         quantiIngressi();
    void        salvaSuFile( ofstream& out );
    void        caricaDaFile( ifstream& in );
    void        stampa();
};

inline rete::rete()
{   str = 0; strat = 0; errore = 0; }

inline rete::~rete()
{   delete[] strat; }

#endif
```

```
////////////////////////////////////////////////////////////////////////////
//rete.cpp

#include <stdlib.h>
#include <fstream.h>
#include <math.h>
#include "rete.h"

rete::rete( int st )
{   str = st;
    errore = 0;
    strat= new strato[str];
}

rete::rete( int st, const int ne[], int in )
{   str = st;
    errore = 0;
    strat = new strato[str];
    strat[0].ricostruisci( in, ne[0] );
    for ( int s = 1; s < str; s++ )
        strat[s].ricostruisci( ne[s-1], ne[s] );
}

rete::rete( const rete& ret )
{   str = ret.str;
    errore = ret.errore;
    strat = new strato[ret.str];
    for ( int s = 0; s < str; s++ )
        strat[s].ricostruisci(  ret.strat[s].quantiIngressi(),
                                ret.strat[s].quantiNeuroni() );
}

void rete::attivazione( const vettore& ingr )
{   strat[0].attivazione( ingr );
    for ( int s = 1; s < str; s++ )
        strat[s].attivazione( strat[s-1].uscita() );
}

vettore rete::uscita()
{   return strat[str-1].uscita();
}

bool rete::osserva( vettore& risuVett )
{   bool conosco = true;
    risuVett = strat[str-1].uscita();
    for ( int i = 0; i < risuVett.lunghezza(); i++ )
        if ( risuVett[i] >= 0.9 ) risuVett[i] = 1;
        else
            if ( risuVett[i] <= 0.1 ) risuVett[i] = 0;
            else
            {   risuVett[i] = ceil ( risuVett[i] - 0.5 );
                conosco = false;
            }
    return conosco;
}

int rete::quantiIngressi()
{   return strat[0].quantiIngressi();
}
```

```
void rete::apprendiEsempio( rete& deltaRete,    const vettore& ingrDes,
                            const vettore& uscDes,  const double eta )
{   vettore uscVett = uscita();
    int numNeu = strat[str-1].quantiNeuroni();
    vettore deltaSucc( numNeu );
    for ( int n = 0; n < numNeu; n++ )
    {   deltaSucc[n] =  eta * ( uscDes[n] - uscVett[n] ) *
                        uscVett[n] * ( 1 - uscVett[n] );
        deltaRete.strat[str-1].assegnaPesi( n, strat[str-2].uscita() *
                                               deltaSucc[n] );
    }
    vettore delta;
    vettore uscPesi;
    int numNeuSucc;

    for ( int s = str - 2; s > 0; s-- )
    {   numNeuSucc = numNeu;
        numNeu = strat[s].quantiNeuroni();
        uscVett.ricostruisci( numNeu );
        uscVett = strat[s].uscita();
        delta.ricostruisci( numNeu );
        uscPesi.ricostruisci( numNeuSucc );
        for ( int n = 0; n < numNeu; n++ )
        {   for ( int j = 0; j < numNeuSucc; j++ )
                uscPesi[j] = strat[s+1].leggiPeso( j, n );
            delta[n] = ( deltaSucc * uscPesi ) *
                       ( eta * uscVett[n] * ( 1 - uscVett[n] ) );
            deltaRete.strat[s].assegnaPesi( n, strat[s-1].uscita() *
                                               delta[n] );
        }
        deltaSucc.ricostruisci( delta.lunghezza());
        deltaSucc = delta;
    }

    numNeuSucc = numNeu;
    numNeu = strat[0].quantiNeuroni();
    uscVett.ricostruisci( numNeu );
    uscVett = strat[0].uscita();
    delta.ricostruisci( numNeu );
    uscPesi.ricostruisci( numNeuSucc );
    for ( n = 0; n < numNeu; n++ )
    {   for ( int j = 0; j < numNeuSucc; j++ )
            uscPesi[j] = strat[1].leggiPeso( j, n );
        delta[n] = ( deltaSucc * uscPesi ) *
                   ( eta * uscVett[n] * ( 1 - uscVett[n] ) );
        deltaRete.strat[0].assegnaPesi( n, ingrDes * delta[n] );
    }
}

void rete::aggiornaErrore( const vettore& uscDes )
{   vettore usc = strat[str-1].uscita();
    for ( int u = 0; u < usc.lunghezza(); u++ )
        errore += pow( ( uscDes[u] - usc[u] ), 2 ) / 2;
}

void permuta( int perm[], int numTrainSet )
{   int temp;
    int r;
    for ( int i = 0; i < numTrainSet; i++ )
        perm[i] = i;
    for ( i = 0; i < numTrainSet; i++ )
    {   r = rand() % numTrainSet;
        temp = perm[i];
```

```
        perm[i] = perm[r];
        perm[r] = temp;
    }
}

bool rete::backpropagation( const vettore ingrDes[], const vettore uscDes[],
                            const int numTrainSet,   const double epslon,
                            const double etaMax,     const int maxTent )
{   rete deltaRete = ( *this );
    int tentativi = 0;
    pesiCasuali();
    int* perm = new int[numTrainSet]; // permutazione casuale indici
    int p;
    double eta = etaMax;
    do
    {   errore = 0;
        permuta( perm, numTrainSet );
        for (int t = 0; t < numTrainSet; t++ )
        {   p = perm[t];
            attivazione( ingrDes[p] );
            deltaRete.pesiZero();
            apprendiEsempio( deltaRete, ingrDes[p], uscDes[p], eta );
            ( *this ) += deltaRete;
            aggiornaErrore( uscDes[p] );
        }
        eta = etaMax * ( 1 - double ( tentativi ) / double ( maxTent ) );
        tentativi++;
        cout << "N. " << tentativi << "\tE = " << errore
             << "\teta = " << eta << endl;
        cerr << "N. " << tentativi << "\tE = " << errore
             << "\teta = " << eta << endl;

    }
    while ( ( errore > epslon ) && ( tentativi < maxTent ) );
    delete[] perm;

    if ( tentativi == maxTent ) return false;
    return true;
}

rete& rete::operator+=( const rete& delta )
{   for ( int s = 0; s < str; s++ )
        strat[s] += delta.strat[s];
    return *this;
}

rete& rete::operator=( const rete& delta )
{   if ( this == &delta )
        return *this;
    else
    for ( int s = 0; s < str; s++ )
        strat[s] = delta.strat[s];
    return *this;
}

void rete::pesiCasuali()
{   for ( int s = 0; s < str; s++ )
        strat[s].pesiCasuali();
}

void rete::pesiZero()
{   for ( int s = 0; s < str; s++ )
        strat[s].pesiZero();
```

```
}

void rete::salvaSuFile( ofstream& out )
{   for ( int s = 0; s < str; s++ )
        strat[s].salvaSuFile( out );
}

void rete::caricaDaFile( ifstream& in )
{   for ( int s = 0; s < str; s++ )
        strat[s].caricaDaFile( in );
}

void rete::stampa()
{   for ( int s = 0; s < str; s++ )
    {   cout << '|' << s;
        strat[s].stampa();
        cout << '|';
    }
}
```

```
////////////////////////////////////////////////////////////////////////////
//inout.h

#ifndef INOUT_H
#define INOUT_H

#include "vettore.h"

typedef char* stringa;
typedef int* vettint;
typedef struct
            {   int     pixelImmY;
                int     pixelImmX;
                int     preamboloPbm;
                int     numStrati;
                int     numIngrRete;
                vettint numNeuStr;
                double  etaMax;
                double  epslon;
                int     maxLunStr;
                int     numTrainSet;
                int     maxTent;
            } parametri;

enum tipo_modo { aiuto, apprendimento, esecuzione };

int     caricaTrainSet( const stringa& nomefile,vettore ingrDes[],
                        vettore uscDes[],           stringa risp[],
                        const tipo_modo modo,       const parametri& par );

boolcaricaParametri( const stringa& nomefile,   parametri& par);

voidcaricaIngr( const stringa& nomefile,vettore& ingr,
                    const parametri& par );

stringa risposta( const vettore& usc, const stringa risp[],
                  const parametri& par );

voidaggiungiEsempio( const stringa& nuovaImm,
                         const stringa& fileInputImm,
                         const stringa& fileTrainingSet,
                         const stringa& fileParametri,
                         const parametri& par );

#endif
```

```
////////////////////////////////////////////////////////////////////////////
//inout.cpp

#include <fstream.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "inout.h"

void caricaUscDes( const stringa& risposta, vettore& uscDes,
                   stringa& risp,           int intero)
{
    risp = new char[ strlen( risposta ) ];
    strcpy( risp, risposta );
    int bit = 0;
    while ( intero != 0 )
    {   uscDes[bit] =  intero % 2 ;
        intero = intero / 2;
        bit++;
    }
}

int caricaTrainSet( const stringa& nomefile,    vettore ingrDes[],
                    vettore uscDes[],           stringa risp[],
                    const tipo_modo modo,       const parametri& par )
{
    stringa nomeImm = new char[ par.maxLunStr ];
    stringa risposta = new char[ par.maxLunStr ];
    char c;
    ifstream in( nomefile );
    if ( !in )
    {   cerr << "errore di apertura file." << endl;
        exit(7);
    }
    do in >> c; while ( c != '-' );
    do in >> c; while ( c != '/' );
    for ( int i = 0; i < par.numTrainSet; i++)
    {   in >> nomeImm; in >> risposta;
        if ( modo == apprendimento )
            caricaIngr( nomeImm,ingrDes[i], par );
        caricaUscDes( risposta, uscDes[i], risp[i], i );
    }
    in.close();
    delete[] nomeImm;
    delete[] risposta;
    return i;
}

bool caricaParametri( const stringa& nomefile,  parametri& par )
{
    char leggi;
    double parametro;
    ifstream in( nomefile );
    if ( !in )
    {   cerr << "errore di apertura file." << endl;
        exit(7);
    }
    bool uscita = false;
    int cont=0;
    while ( !uscita )
    {   do  in >> leggi; while ( ( leggi != '=' ) && ( leggi != '.' ) );
        if ( leggi == '=' )
        {   in >> parametro;
```

```
            switch( cont )
            {   case 0:     par.pixelImmY =     int( parametro ); break;
                case 1:     par.pixelImmX =     int( parametro ); break;
                case 2:     par.preamboloPbm =  int( parametro ); break;
                case 3:     par.numStrati =     int( parametro ); break;
                case 4:     par.numIngrRete =   int( parametro ); break;
                case 5:     par.epslon =        double( parametro ); break;
                case 6:     par.etaMax =        double( parametro ); break;
                case 7:     par.maxTent =       int( parametro ); break;
                case 8:     par.numTrainSet =   int( parametro ); break;
                case 9:     par.maxLunStr =     int (parametro ); break;
                case 10:    par.numNeuStr =     new int[par.numStrati];
                            par.numNeuStr[0] =  int( parametro );
                            for ( int s = 1; s < par.numStrati; s++ )
                            {   in >> parametro;
                                par.numNeuStr[s]= int( parametro );
                            }
                            break;
            }
        }
        else if ( leggi == '.' ) uscita=true;
        cont++;
    }
    in.close();
    return uscita;
}

void caricaIngr( const stringa& nomefile,   vettore& ingr,
                 const parametri& par )
{   char leggiCar;
    int  leggiInt;
    ifstream in( nomefile );
    if ( !in )
    {   cerr << "errore di apertura file." << endl;
        exit(7);
    }
    for ( int i = 0; i < par.preamboloPbm; i++) in >> leggiCar;
    in >> leggiInt;
    if ( leggiInt != par.pixelImmX )
    {   cerr << "errore nella larghezza dell\'immagine" << endl;
        exit(8);
    }
    in >> leggiInt;
    if ( leggiInt != par.pixelImmY )
    {   cerr << "errore nella altezza dell\'immagine" << endl;
        exit(8);
    }
    for ( i = 0; i < par.numIngrRete; i++ )
    {   in >> leggiInt;
        ingr[i] = scalare ( leggiInt );
    }
    in.close();
}

stringa risposta( const vettore& usc, const stringa risp[],
                  const parametri& par )
{
    int numRisp = usc.intero();
    if ( numRisp >=0 && numRisp < par.numTrainSet )
        return risp[ usc.intero() ];
    return "non so cosa";
}
```

```
void aggiungiEsempio( const stringa& nuovaImm,
                      const stringa& fileInputImm,
                      const stringa& fileTrainingSet,
                      const stringa& fileParametri,
                      const parametri& par )
{
    stringa nomeImm = new char[ par.maxLunStr ];
    stringa risposta = new char[ par.maxLunStr ];
    strcpy ( nomeImm, nuovaImm );

    ifstream in( fileInputImm );
        ofstream out( strcat( nomeImm,".pbm") );
            char c;
            while ( in.get(c) )
                out.put(c);
        out.close();
    in.close();

    ifstream ing( fileTrainingSet );
        ofstream usc( "trainingset.tmp" );
            do
            {   ing.get(c); usc.put(c); }
            while ( c != '-' );
            do
            {   ing.get(c); usc.put(c); }
            while ( c != '/' );
            for ( int i = 0; i < par.numTrainSet; i++)
            {   ing >> nomeImm; ing >> risposta;
                usc << endl << nomeImm << '\t' << risposta;
            }
            strcpy( nomeImm, nuovaImm );
            usc << endl << strcat( nomeImm, ".pbm" ) << '\t' << nuovaImm;
            while ( ing.get(c) )
                usc.put(c);
        usc.close();
    ing.close();

    delete[] nomeImm;
    delete[] risposta;

    _unlink( fileTrainingSet );
    rename( "trainingset.tmp", fileTrainingSet );

    double parametro;
    ifstream ingr( fileParametri );
        ofstream usci( "parametri.tmp" );
            int cont = 0;
            while ( ingr.get(c) )
            {   usci.put(c);
                if (c == '=')
                {   cont++;
                    ingr >> parametro;
                    if ( cont == 9 ) parametro++;
                    usci << ' ' << parametro;
                }
            }
        usci.close();
    ingr.close();

    _unlink(fileParametri);
    rename( "parametri.tmp", fileParametri );
}
```

```
////////////////////////////////////////////////////////////////////////////
//main.cpp

#include <fstream.h>
#include <stdlib.h>
#include "rete.h"
#include "inout.h"

stringa const fileTrainingSet   = "trainingset.cfg";
stringa const fileParametri     = "parametri.cfg";
stringa const fileInputImm      = "input.pbm";
stringa const fileRete          = "rete.dat";

void main( int argc, char* argv[] )
{
 tipo_modo modo = aiuto;
 if (argc == 2)
     if ( argv[1][0] == 'a' ) modo = apprendimento;
     else if ( argv[1][0] == 'e' ) modo = esecuzione;

 if ( modo == aiuto )
 {cout <<
  "-------------------------------------------------------------------------\n"
  "FORME, by M.Cimino & G. D\'Alessandro - apr 2001 \n"
  "-------------------------------------------------------------------------\n"
  "La rete impara a riconoscere delle forme, e risponde anche in  presenza\n"
  "di rumore. I parametri sono caricati dal file \'" << fileParametri << "\'.\n"
  "OPZIONI:\'forme a\', modalita\' di apprendimento,  le immagini pulite e \n"
  "le risposte relative sono caricate dal file \'" << fileTrainingSet << "\';\n"
  "        \'forme e\', modalita\' di esecuzione, la rete osserva l\'immagine\n"
  "\'" << fileInputImm << "\' con del rumore aggiunto e da\' una risposta.\n"
  "-------------------------------------------------------------------------\n";
 }
 else
 {parametri par;

  if ( !caricaParametri( fileParametri, par ) )
  { cerr << "errore sul file \'" << fileParametri << "\'" << endl;
    exit(7);
  }

  int uscRete = par.numNeuStr[ par.numStrati - 1 ];
  rete vista( par.numStrati, par.numNeuStr, par.numIngrRete );

  vettore* ingrDes = new vettore[ par.numTrainSet ];

  if ( modo == apprendimento )
    for ( int i = 0; i < par.numTrainSet; i++ )
        ingrDes[i].ricostruisci( par.numIngrRete );

  vettore* uscDes = new vettore[ par.numTrainSet ];
  for ( int  i = 0; i < par.numTrainSet; i++ )
  { uscDes[i].ricostruisci( uscRete );
    uscDes[i].valoriZero();
  }

  stringa* risp = new stringa[ par.numTrainSet ];
    for ( i = 0; i < par.numTrainSet; i++ )
        risp[i] = 0;

  if ( par.numTrainSet !=
       caricaTrainSet( fileTrainingSet, ingrDes, uscDes, risp, modo, par ) )
```

```
  { cerr << "errore nel file \'" << fileTrainingSet << "\'" << endl;
    exit(6);
  }

  if ( modo == apprendimento )
  { if ( !vista.backpropagation( ingrDes,    uscDes, par.numTrainSet,
                                 par.epslon, par.etaMax, par.maxTent ) )
    {   cerr <<
        "la rete non apprende bene: variare i parametri"
        << endl;
        exit(8);
    }

    ofstream out( fileRete );
    vista.salvaSuFile( out );
    out.close();
  }
  else
  {
    ifstream in( fileRete );
    vista.caricaDaFile( in );
    in.close();

    vettore ingr( par.numIngrRete );
    caricaIngr( fileInputImm, ingr, par );
    vista.attivazione( ingr );
    vettore risuVett ( vista.uscita().lunghezza() );

    if ( vista.osserva( risuVett ) )
        cout <<
        " L\'immagine somiglia ad un(a) " <<
        risposta( risuVett, risp, par )
        << endl;
    else
    {   cout <<
        " L\'immagine sembra vagamente un(a) " <<
        risposta( risuVett, risp, par )
        << endl;
        char c;
        cout << " E\' una nuova forma (s/n)? ";
        cin >> c;
        if ( c == 's' )
        {   cout << " Come si chiama ? ";
            stringa nuovaImm = new char[ par.maxLunStr ];
            cin >> nuovaImm;
            aggiungiEsempio( nuovaImm,  fileInputImm,   fileTrainingSet,
                             fileParametri, par );
            cout <<
            " L\'ho inserita nel mio insieme di apprendimento."
            << endl;
            cout <<
            " E\' necessario riaddestrare la rete."
            << endl;
        }
    }
  }

  delete[] risp;
  if ( modo == apprendimento ) delete[] ingrDes;
  delete[] uscDes;
  delete[] par.numNeuStr;
 }
}
```

```
////////////////////////////////////////////////////////////////////////////
// parametri.cfg

PARAMETRI DI INIZIALIZZAZIONE:
--------------------------------------------------------------------------
altezza delle immagini in pixel = 13
larghezza delle immagini in pixel = 13
lunghezza del preambolo del file pbm (caratteri) = 24
numero degli strati della rete  = 3
numero degli ingressi della rete (13x13) = 169
errore minimo prefissato (epslon) = 0.004
rapidita' di discesa iniziale (eta massimo ) = 1
max tentativi di insegnamento del training set = 1000
numero di esempi del Training Set = 12
lunghezza max delle stringhe di risposta = 64
numero di neuroni per ogni strato = 256 128 4
--------------------------------------------------------------------------
.
nota: l'ultimo strato ha logbin ( esempi trainset )


////////////////////////////////////////////////////////////////////////////
// trainingset.cfg

/* NOMI DELLE IMMAGINI DEL TRAINIG SET ED USCITE DESIDERATE RELATIVE
------------------------------------------------------------------------ */
casa.pbm        casa
croce.pbm       croce
cuore.pbm       cuore
donna.pbm       donna
freccia.pbm     freccia
luna.pbm        luna
nota.pbm        nota
ombrello.pbm    ombrello
rombo.pbm       rombo
smile.pbm       smile
stella.pbm      stella
uomo.pbm        uomo
/* ------------------------------------------------------------------------ */
```

```
////////////////////////////////////////////////////////////////////////////
// files della directory di progetto


 Il volume nell'unità D è DATI
 Numero di serie del volume: 1E4A-1F34
 Directory di D:\Informatica\c++\forme

.              <DIR>        10/05/01  15.04 .
..             <DIR>        10/05/01  15.04 ..
OMBRELLO PBM            380  11/05/01  14.43 ombrello.pbm
CROCE    PBM            380  11/05/01  11.07 croce.pbm
STELLA   PBM            380  11/05/01  14.02 stella.pbm
FORME    DSP          4.950  06/05/01   1.01 forme.dsp
FORME    DSW            535  05/05/01  23.11 forme.dsw
FORME    NCB        132.096  13/05/01  22.29 forme.ncb
FORME    OPT         50.688  13/05/01  22.29 forme.opt
FORME    PLG            608  12/05/01  23.59 forme.plg
INOUT    CPP          4.967  12/05/01  16.34 inout.cpp
INOUT    H            1.157  12/05/01  16.03 inout.h
INPUT    PBM            380  12/05/01  16.44 input.pbm
MAIN     CPP          3.728  12/05/01  16.15 main.cpp
NEURONE  CPP          2.096  10/05/01  21.44 neurone.cpp
NEURONE  H            1.265  08/05/01  23.05 neurone.h
PARAME~1 CFG            743  12/05/01  12.31 parametri.cfg
RETE     DAT        804.643  11/05/01  19.26 rete.dat
RETE     CPP          5.374  12/05/01  23.59 rete.cpp
FILEUSC  TXT              0  14/05/01  18.16 fileusc.txt
RETE     H            1.354  12/05/01  23.21 rete.h
STRATO   CPP          2.518  12/05/01   1.38 strato.cpp
STRATO   H            1.161  12/05/01   1.39 strato.h
TRAINI~1 CFG            430  11/05/01  14.49 trainingset.cfg
VETTORE  CPP          3.414  12/05/01   0.22 vettore.cpp
VETTORE  H            1.295  12/05/01   0.18 vettore.h
DEBUG          <DIR>        10/05/01  15.04 Debug
LUNA     PBM            380  11/05/01  11.20 luna.pbm
FRECCIA  PBM            380  11/05/01  11.22 freccia.pbm
SMILE    PBM            380  11/05/01  11.24 smile.pbm
UOMO     PBM            380  11/05/01  14.38 uomo.pbm
CASA     PBM            380  11/05/01  11.38 casa.pbm
DONNA    PBM            380  11/05/01  14.37 donna.pbm
ROMBO    PBM            380  11/05/01  14.23 rombo.pbm
NOTA     PBM            380  11/05/01  14.37 nota.pbm
CUORE    PBM            380  11/05/01  11.06 cuore.pbm
        33 file      1.027.962 byte
         3 dir        32.636.928 byte disponibili
```

4. TEST DI VERIFICA

4.1 Fase di apprendimento

Con un Training Set di 12 immagini 13 x 13 , abbiamo ottenuto buone performances - in termini di velocita' di apprendimento - con due strati nascosti di 256 e 128 neuroni rispettivamente; troppi strati provocano oscillazioni dell' errore; troppo pochi neuroni impediscono all'errore di scendere velocemente oltre una certa soglia.

Il parametro eta diminuisce linearmente, per limitare le oscillazioni, a partire da un massimo configurabile dall'utente, arrivando a zero sul massimo numero di epoche presentabili, ma e' importante che sia abbastanza alto nella fase iniziale per una rapida convergenza.

Dopo sole 240 epoche, raggiunte in qualche minuto su un processore a 200 Mhz, l'errore globale e' sceso gia' a 0.01; in tali condizioni, la rete riconosce bene le forme alterate fino ad un 5% dei pixel.

In un tempo complessivo di un quarto d'ora circa, si puo' arrivare a 480 epoche, con un errore di 0.004, che porta al comportamento della rete documentato nel prossimo paragrafo.

Mostriamo nella pagina seguente un grafico sulla fase di apprendimento, costruito con Matlab sulle tabulazioni mostrate successivamente, dell'errore globale e di eta, ottenute redirigendo l'output della fase di apprendimento su un file.

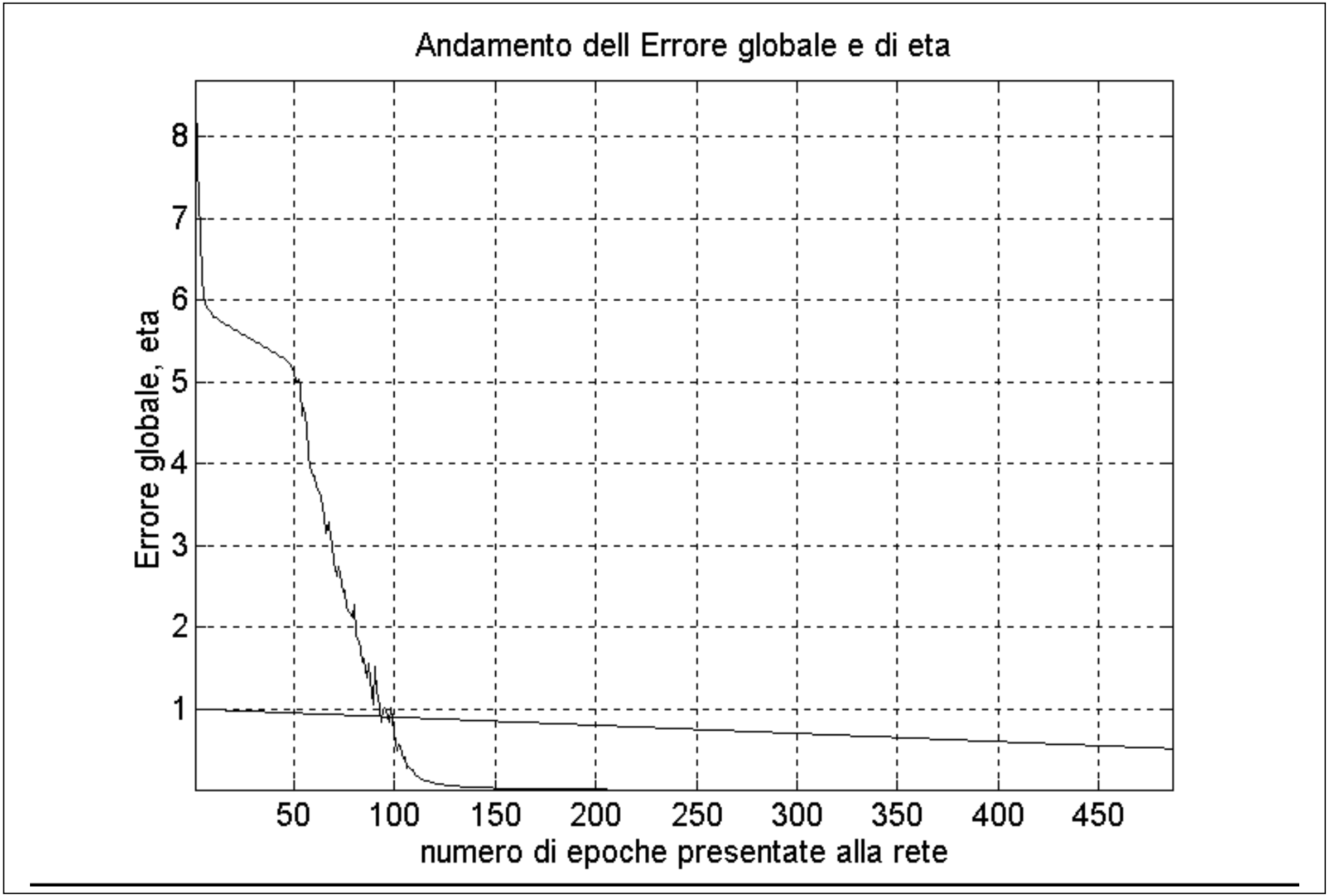
A scopo sperimentale, mostriamo infine qui in basso una parte dei pesi memorizzati nel file "rete.dat".

```
////////////////////////////////////////////////////////////////////////////
// rete.dat
```

" -0.0767306 0.00978236 -0.047184 0.0474985 0.0856974 0.0393245 -0.0233417 0.103338 0.0879669 0.0752323 -0.0128442 0.0552221 0.0323848 0.0020823 -0.0749123 -0.145784 -0.0614084 -0.0797533 -0.0510397 -0.0961578 0.117254 -0.0642816 -0.154329 -0.147371 -0.0830167 -0.0187877 0.00487122 -0.0602149 0.087175 0.0287537 -0.115359 -0.0539008 -0.0150355 -0.0255658 -0.127602 0.00326716 0.0434936 -0.0246116 0.00305889 -0.101636 0.175311 0.0360661 -0.0374366 0.0666879 -0.000217325 -0.17996 -0.0105224 -0.0738458 -0.0129699 -0.0434839 0.0888806 -0.0182966 0.12154 0.121727 -0.00319906 0.0271095 -0.0460784 0.040752 0.

< …continua… >

-0.275145 -0.414546 0.109545 -0.339836 -0.382233 -0.259316 0.461987 -0.334251 -0.310467 0.0346937 0.936745 -0.4822 -0.298711 -0.450158 -0.549466 1.93563 0.438058 -0.457561 -0.455836 1.13201 -0.499072 -0.448039 -0.157941 -0.307582 -0.150381 0.0458865 1.24897 -0.303396 -0.154982 -0.533387 -0.583592 -0.286262 -0.20406 -0.693549 -0.470015 -0.400392 -0.312973 -0.0261946 -0.302569 -0.115807 -0.327762 -0.382398 1.17366 -0.242562 -0.223615 -0.245805 -0.437966 -0.346465 -0.281667 -0.590992 -0.405723 -0.329451 -0.190307 0.563161 -0.313509 -0.341731 -0.340781 -0.520482 -0.536133 -0.996691 -0.0713138 -0.375001 -0.123897 -0.242366 0.296633 -0.359105 -0.469797 0.586558 -0.548143 0.770457 -0.449135 -0.325467 -0.25971 -0.233002 -0.340479 -0.299402 0.0445204 -0.509107 0.775412 -0.272661 -0.347003 0.75659 -0.258497 0.390855 -0.372289 -0.303936 -3.74081 -0.358987 -0.269937 -0.511559 0.029809 0.417708 -0.372005 -0.244016 0.178045 -0.315657 0.511842 1.4728 -0.202089 -0.321354 -0.29641 -0.422172 0.0699076 -0.229146 -0.450732 0.641093 "


Andamento dell Errore globale e di eta
Errore globale, eta
numero di epoche presentate alla rete
1
2
3
4
5
6
7
8
50
100
150
200
250
300
350
400
450

// output rediretto durante l'apprendimento

```
 1   E = 8.68881    eta = 1
 2   E = 7.67713    eta = 0.999
 3   E = 6.80999    eta = 0.998
 4   E = 6.28385    eta = 0.997
 5   E = 6.08775    eta = 0.996
 6   E = 5.9511     eta = 0.995
 7   E = 5.89303    eta = 0.994
 8   E = 5.87541    eta = 0.993
 9   E = 5.84911    eta = 0.992
 10  E = 5.7976     eta = 0.991
 11  E = 5.78709    eta = 0.99
 12  E = 5.76724    eta = 0.989
 13  E = 5.75479    eta = 0.988
 14  E = 5.73257    eta = 0.987
 15  E = 5.70903    eta = 0.986
 16  E = 5.70592    eta = 0.985
 17  E = 5.69238    eta = 0.984
 18  E = 5.67434    eta = 0.983
 19  E = 5.65657    eta = 0.982
 20  E = 5.64412    eta = 0.981
 21  E = 5.6352     eta = 0.98
 22  E = 5.6119     eta = 0.979
 23  E = 5.60456    eta = 0.978
 24  E = 5.58683    eta = 0.977
 25  E = 5.57729    eta = 0.976
 26  E = 5.56149    eta = 0.975
 27  E = 5.54592    eta = 0.974
 28  E = 5.53415    eta = 0.973
 29  E = 5.52181    eta = 0.972
 30  E = 5.508      eta = 0.971
 31  E = 5.49152    eta = 0.97
 32  E = 5.47678    eta = 0.969
 33  E = 5.46129    eta = 0.968
 34  E = 5.44779    eta = 0.967
 35  E = 5.42995    eta = 0.966
 36  E = 5.41801    eta = 0.965
 37  E = 5.40462    eta = 0.964
 38  E = 5.38738    eta = 0.963
 39  E = 5.3743     eta = 0.962
 40  E = 5.36527    eta = 0.961
 41  E = 5.34919    eta = 0.96
 42  E = 5.33086    eta = 0.959
 43  E = 5.31413    eta = 0.958
 44  E = 5.30464    eta = 0.957
 45  E = 5.28528    eta = 0.956
 46  E = 5.26105    eta = 0.955
 47  E = 5.24429    eta = 0.954
 48  E = 5.21673    eta = 0.953
 49  E = 5.17508    eta = 0.952
 50  E = 5.13054    eta = 0.951
 51  E = 4.99352    eta = 0.95
 52  E = 5.03413    eta = 0.949
 53  E = 4.93368    eta = 0.948
 54  E = 4.58177    eta = 0.947
 55  E = 4.68615    eta = 0.946
 56  E = 4.47353    eta = 0.945
 57  E = 4.11062    eta = 0.944
 58  E = 3.97548    eta = 0.943
 59  E = 3.85872    eta = 0.942
 60  E = 3.84717    eta = 0.941
 61  E = 3.72406    eta = 0.94
 62  E = 3.69517    eta = 0.939
 63  E = 3.62065    eta = 0.938
 64  E = 3.46354    eta = 0.937
 65  E = 3.36675    eta = 0.936
 66  E = 3.15227    eta = 0.935
 67  E = 3.28195    eta = 0.934
 68  E = 3.09317    eta = 0.933
 69  E = 3.03028    eta = 0.932
 70  E = 2.80136    eta = 0.931
 71  E = 2.63458    eta = 0.93
 72  E = 2.73405    eta = 0.929
 73  E = 2.63816    eta = 0.928
 74  E = 2.4355     eta = 0.927
 75  E = 2.44914    eta = 0.926
 76  E = 2.25086    eta = 0.925
 77  E = 2.20163    eta = 0.924
 78  E = 2.16934    eta = 0.923
 79  E = 2.12274    eta = 0.922
 80  E = 2.27824    eta = 0.921
 81  E = 1.90632    eta = 0.92
 82  E = 1.83806    eta = 0.919
 83  E = 1.73058    eta = 0.918
 84  E = 1.57721    eta = 0.917
 85  E = 1.60877    eta = 0.916
 86  E = 1.39314    eta = 0.915
 87  E = 1.5642     eta = 0.914
 88  E = 1.34121    eta = 0.913
 89  E = 1.05683    eta = 0.912
 90  E = 1.51253    eta = 0.911
 91  E = 1.19591    eta = 0.91
 92  E = 1.17294    eta = 0.909
 93  E = 0.848799   eta = 0.908
 94  E = 1.00396    eta = 0.907
 95  E = 1.01794    eta = 0.906
 96  E = 0.989354   eta = 0.905
 97  E = 0.842278   eta = 0.904
 98  E = 1.02246    eta = 0.903
 99  E = 0.894117   eta = 0.902
 100 E = 0.702605   eta = 0.901
 101 E = 0.48134    eta = 0.9
 102 E = 0.573436   eta = 0.899
 103 E = 0.550448   eta = 0.898
 104 E = 0.393073   eta = 0.897
 105 E = 0.409364   eta = 0.896
 106 E = 0.285206   eta = 0.895
 107 E = 0.287325   eta = 0.894
 108 E = 0.26608    eta = 0.893
 109 E = 0.26063    eta = 0.892
 110 E = 0.19646    eta = 0.891
 111 E = 0.175215   eta = 0.89
 112 E = 0.164691   eta = 0.889
 113 E = 0.135116   eta = 0.888
 114 E = 0.14164    eta = 0.887
 115 E = 0.126261   eta = 0.886
 116 E = 0.111792   eta = 0.885
 117 E = 0.113289   eta = 0.884
 118 E = 0.100292   eta = 0.883
 119 E = 0.0978912  eta = 0.882
 120 E = 0.0906022  eta = 0.881
 121 E = 0.0849258  eta = 0.88
 122 E = 0.0808564  eta = 0.879
 123 E = 0.076851   eta = 0.878
 124 E = 0.0737832  eta = 0.877
 125 E = 0.0709976  eta = 0.876
 126 E = 0.067566   eta = 0.875
 127 E = 0.0648071  eta = 0.874
 128 E = 0.0624459  eta = 0.873
 129 E = 0.0604303  eta = 0.872
 130 E = 0.0579592  eta = 0.871
 131 E = 0.055636   eta = 0.87
 132 E = 0.0540941  eta = 0.869
 133 E = 0.0518272  eta = 0.868
 134 E = 0.0505168  eta = 0.867
 135 E = 0.0487336  eta = 0.866
 136 E = 0.0471117  eta = 0.865
 137 E = 0.0456197  eta = 0.864
 138 E = 0.0442945  eta = 0.863
 139 E = 0.0430714  eta = 0.862
 140 E = 0.0418426  eta = 0.861
 141 E = 0.040638   eta = 0.86
 142 E = 0.0395886  eta = 0.859
 143 E = 0.0384636  eta = 0.858
 144 E = 0.037497   eta = 0.857
 145 E = 0.0362459  eta = 0.856
 146 E = 0.0356715  eta = 0.855
 147 E = 0.0344732  eta = 0.854
 148 E = 0.0339684  eta = 0.853
 149 E = 0.0329872  eta = 0.852
 150 E = 0.0323573  eta = 0.851
 151 E = 0.0316599  eta = 0.85
 152 E = 0.0308804  eta = 0.849
 153 E = 0.0301083  eta = 0.848
 154 E = 0.02957    eta = 0.847
 155 E = 0.0289338  eta = 0.846
 156 E = 0.0283504  eta = 0.845
 157 E = 0.027784   eta = 0.844
 158 E = 0.0270809  eta = 0.843
 159 E = 0.0267011  eta = 0.842
 160 E = 0.0261069  eta = 0.841
 161 E = 0.025642   eta = 0.84
 162 E = 0.0251688  eta = 0.839
 163 E = 0.0246961  eta = 0.838
 164 E = 0.0242645  eta = 0.837
 165 E = 0.0238165  eta = 0.836
 166 E = 0.0233966  eta = 0.835
 167 E = 0.0229776  eta = 0.834
 168 E = 0.0226111  eta = 0.833
 169 E = 0.0222115  eta = 0.832
 170 E = 0.021866   eta = 0.831
 171 E = 0.0214951  eta = 0.83
 172 E = 0.0211506  eta = 0.829
 173 E = 0.0207599  eta = 0.828
 174 E = 0.0204909  eta = 0.827
 175 E = 0.0201767  eta = 0.826
 176 E = 0.0198624  eta = 0.825
 177 E = 0.019569   eta = 0.824
 178 E = 0.0192746  eta = 0.823
 179 E = 0.0190039  eta = 0.822
 180 E = 0.0187123  eta = 0.821
 181 E = 0.0184611  eta = 0.82
 182 E = 0.0182016  eta = 0.819
 183 E = 0.0179443  eta = 0.818
 184 E = 0.0177133  eta = 0.817
 185 E = 0.0174454  eta = 0.816
 186 E = 0.0172132  eta = 0.815
 187 E = 0.0170149  eta = 0.814
 188 E = 0.0167705  eta = 0.813
 189 E = 0.0165775  eta = 0.812
 190 E = 0.0163662  eta = 0.811
 191 E = 0.016161   eta = 0.81
 192 E = 0.0159613  eta = 0.809
 193 E = 0.0157656  eta = 0.808
 194 E = 0.0155614  eta = 0.807
 195 E = 0.0153773  eta = 0.806
 196 E = 0.0152022  eta = 0.805
 197 E = 0.0150419  eta = 0.804
 198 E = 0.0148671  eta = 0.803
 199 E = 0.0146954  eta = 0.802
 200 E = 0.0145303  eta = 0.801
 201 E = 0.0143637  eta = 0.8
 202 E = 0.0141985  eta = 0.799
 203 E = 0.0140575  eta = 0.798
 204 E = 0.0138833  eta = 0.797
 205 E = 0.0137552  eta = 0.796
 206 E = 0.0136113  eta = 0.795
 207 E = 0.0134641  eta = 0.794
 208 E = 0.0133307  eta = 0.793
 209 E = 0.0131967  eta = 0.792
 210 E = 0.0130623  eta = 0.791
 211 E = 0.0129305  eta = 0.79
 212 E = 0.0128009  eta = 0.789
 213 E = 0.012676   eta = 0.788
 214 E = 0.0125572  eta = 0.787
 215 E = 0.0124378  eta = 0.786
 216 E = 0.0123185  eta = 0.785
 217 E = 0.0122035  eta = 0.784
 218 E = 0.0120916  eta = 0.783
 219 E = 0.0119778  eta = 0.782
 220 E = 0.0118721  eta = 0.781
 221 E = 0.0117647  eta = 0.78
 222 E = 0.0116575  eta = 0.779
 223 E = 0.011557   eta = 0.778
 224 E = 0.0114513  eta = 0.777
 225 E = 0.0113513  eta = 0.776
 226 E = 0.0112531  eta = 0.775
 227 E = 0.0111642  eta = 0.774
 228 E = 0.0110706  eta = 0.773
 229 E = 0.0109781  eta = 0.772
 230 E = 0.0108821  eta = 0.771
 231 E = 0.0107977  eta = 0.77
 232 E = 0.0107108  eta = 0.769
 233 E = 0.0106226  eta = 0.768
 234 E = 0.0105383  eta = 0.767
 235 E = 0.0104546  eta = 0.766
 236 E = 0.0103739  eta = 0.765
 237 E = 0.0102917  eta = 0.764
 238 E = 0.0102132  eta = 0.763
 239 E = 0.0101341  eta = 0.762
 240 E = 0.0100594  eta = 0.761
 241 E = 0.00998379 eta = 0.76
 242 E = 0.00990814 eta = 0.759
 243 E = 0.00983759 eta = 0.758
```

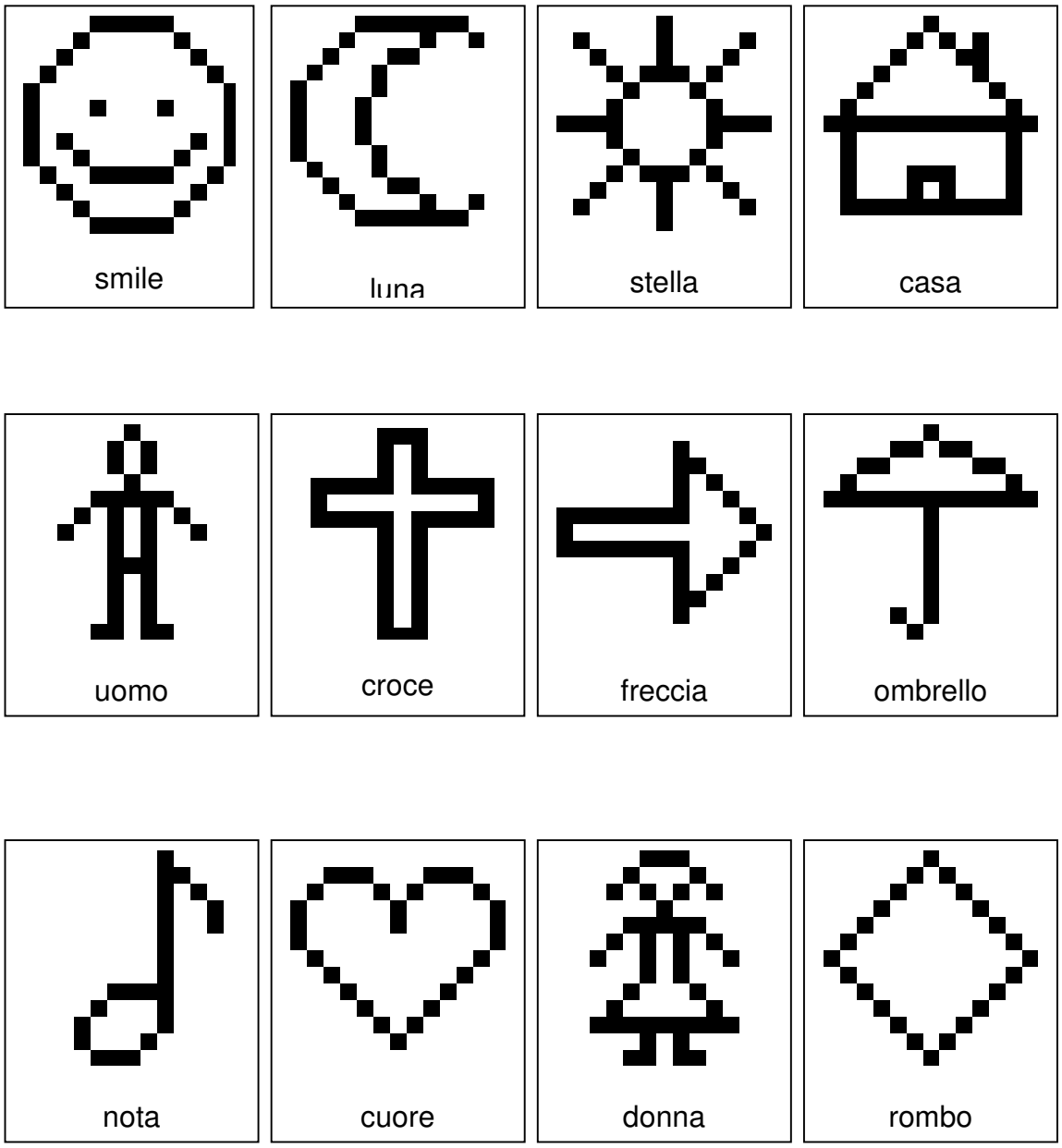
244 E = 0.00976348 eta = 0.757
245 E = 0.00968608 eta = 0.756
246 E = 0.00962294 eta = 0.755
247 E = 0.009554 eta = 0.754
248 E = 0.00948715 eta = 0.753
249 E = 0.0094164 eta = 0.752
250 E = 0.00935539 eta = 0.751
251 E = 0.00929074 eta = 0.75
252 E = 0.00922841 eta = 0.749
253 E = 0.00916546 eta = 0.748
254 E = 0.00910157 eta = 0.747
255 E = 0.0090433 eta = 0.746
256 E = 0.00898165 eta = 0.745
257 E = 0.00892398 eta = 0.744
258 E = 0.0088658 eta = 0.743
259 E = 0.00880854 eta = 0.742
260 E = 0.00875099 eta = 0.741
261 E = 0.00869657 eta = 0.74
262 E = 0.00864084 eta = 0.739
263 E = 0.00858652 eta = 0.738
264 E = 0.00853356 eta = 0.737
265 E = 0.00848076 eta = 0.736
266 E = 0.00842846 eta = 0.735
267 E = 0.00837674 eta = 0.734
268 E = 0.00832558 eta = 0.733
269 E = 0.00827649 eta = 0.732
270 E = 0.00822717 eta = 0.731
271 E = 0.00817875 eta = 0.73
272 E = 0.00813022 eta = 0.729
273 E = 0.00808384 eta = 0.728
274 E = 0.00803682 eta = 0.727
275 E = 0.00799128 eta = 0.726
276 E = 0.00794248 eta = 0.725
277 E = 0.00790052 eta = 0.724
278 E = 0.00785617 eta = 0.723
279 E = 0.00781176 eta = 0.722
280 E = 0.00776889 eta = 0.721
281 E = 0.00772551 eta = 0.72
282 E = 0.00768396 eta = 0.719
283 E = 0.00764191 eta = 0.718
284 E = 0.00760123 eta = 0.717
285 E = 0.00756053 eta = 0.716
286 E = 0.00752032 eta = 0.715
287 E = 0.007481 eta = 0.714
288 E = 0.00744156 eta = 0.713
289 E = 0.00740297 eta = 0.712
290 E = 0.00736454 eta = 0.711
291 E = 0.00732678 eta = 0.71
292 E = 0.00728953 eta = 0.709
293 E = 0.00725274 eta = 0.708
294 E = 0.00721566 eta = 0.707
295 E = 0.00718022 eta = 0.706
296 E = 0.00714466 eta = 0.705
297 E = 0.0071092 eta = 0.704
298 E = 0.0070745 eta = 0.703
299 E = 0.00703994 eta = 0.702
300 E = 0.00700566 eta = 0.701
301 E = 0.00697207 eta = 0.7
302 E = 0.00693924 eta = 0.699
303 E = 0.00690474 eta = 0.698
304 E = 0.00687386 eta = 0.697
305 E = 0.00684152 eta = 0.696
306 E = 0.00680976 eta = 0.695
307 E = 0.00677822 eta = 0.694
308 E = 0.00674708 eta = 0.693
309 E = 0.00671609 eta = 0.692
310 E = 0.00668574 eta = 0.691
311 E = 0.00665574 eta = 0.69
312 E = 0.00662555 eta = 0.689
313 E = 0.00659651 eta = 0.688
314 E = 0.00656717 eta = 0.687
315 E = 0.00653828 eta = 0.686
316 E = 0.0065095 eta = 0.685
317 E = 0.00648152 eta = 0.684
318 E = 0.00645311 eta = 0.683
319 E = 0.00642582 eta = 0.682
320 E = 0.00639769 eta = 0.681
321 E = 0.00637126 eta = 0.68
322 E = 0.00634435 eta = 0.679
323 E = 0.00631779 eta = 0.678
324 E = 0.00629132 eta = 0.677
325 E = 0.0062654 eta = 0.676
326 E = 0.00623902 eta = 0.675
327 E = 0.00621368 eta = 0.674
328 E = 0.00618902 eta = 0.673
329 E = 0.00616397 eta = 0.672
330 E = 0.00613937 eta = 0.671
331 E = 0.00611482 eta = 0.67
332 E = 0.00608964 eta = 0.669
333 E = 0.00606645 eta = 0.668
334 E = 0.00604231 eta = 0.667
335 E = 0.00601907 eta = 0.666
336 E = 0.00599578 eta = 0.665
337 E = 0.00597248 eta = 0.664
338 E = 0.005949 eta = 0.663
339 E = 0.00592689 eta = 0.662
340 E = 0.00590457 eta = 0.661
341 E = 0.00588211 eta = 0.66
342 E = 0.00586024 eta = 0.659
343 E = 0.00583831 eta = 0.658
344 E = 0.00581661 eta = 0.657
345 E = 0.0057948 eta = 0.656
346 E = 0.00577389 eta = 0.655
347 E = 0.0057527 eta = 0.654
348 E = 0.00573151 eta = 0.653
349 E = 0.00571146 eta = 0.652
350 E = 0.00569053 eta = 0.651
351 E = 0.0056706 eta = 0.65
352 E = 0.00565046 eta = 0.649
353 E = 0.00563063 eta = 0.648
354 E = 0.00561089 eta = 0.647
355 E = 0.00559092 eta = 0.646
356 E = 0.00557185 eta = 0.645
357 E = 0.00555279 eta = 0.644
358 E = 0.00553373 eta = 0.643
359 E = 0.0055149 eta = 0.642
360 E = 0.00549605 eta = 0.641
361 E = 0.00547747 eta = 0.64
362 E = 0.0054592 eta = 0.639
363 E = 0.0054409 eta = 0.638
364 E = 0.00542274 eta = 0.637
365 E = 0.00540493 eta = 0.636
366 E = 0.00538701 eta = 0.635
367 E = 0.00536959 eta = 0.634
368 E = 0.00535197 eta = 0.633
369 E = 0.00533481 eta = 0.632
370 E = 0.00531768 eta = 0.631
371 E = 0.00530066 eta = 0.63
372 E = 0.00528335 eta = 0.629
373 E = 0.00526706 eta = 0.628
374 E = 0.0052503 eta = 0.627
375 E = 0.00523391 eta = 0.626
376 E = 0.00521764 eta = 0.625
377 E = 0.00520129 eta = 0.624
378 E = 0.00518524 eta = 0.623
379 E = 0.00516942 eta = 0.622
380 E = 0.00515369 eta = 0.621
381 E = 0.00513764 eta = 0.62
382 E = 0.00512254 eta = 0.619
383 E = 0.0051071 eta = 0.618
384 E = 0.00509169 eta = 0.617
385 E = 0.00507661 eta = 0.616
386 E = 0.00506166 eta = 0.615
387 E = 0.00504675 eta = 0.614
388 E = 0.00503185 eta = 0.613
389 E = 0.0050172 eta = 0.612
390 E = 0.00500266 eta = 0.611
391 E = 0.00498825 eta = 0.61
392 E = 0.00497387 eta = 0.609
393 E = 0.00495967 eta = 0.608
394 E = 0.00494551 eta = 0.607
395 E = 0.0049316 eta = 0.606
396 E = 0.0049177 eta = 0.605
397 E = 0.0049039 eta = 0.604
398 E = 0.00489022 eta = 0.603
399 E = 0.00487661 eta = 0.602
400 E = 0.00486321 eta = 0.601
401 E = 0.00484979 eta = 0.6
402 E = 0.00483645 eta = 0.599
403 E = 0.00482337 eta = 0.598
404 E = 0.0048101 eta = 0.597
405 E = 0.00479739 eta = 0.596
406 E = 0.00478442 eta = 0.595
407 E = 0.0047718 eta = 0.594
408 E = 0.0047591 eta = 0.593
409 E = 0.0047463 eta = 0.592
410 E = 0.00473407 eta = 0.591
411 E = 0.00472165 eta = 0.59
412 E = 0.00470937 eta = 0.589
413 E = 0.00469717 eta = 0.588
414 E = 0.00468503 eta = 0.587
415 E = 0.00467302 eta = 0.586
416 E = 0.00466104 eta = 0.585
417 E = 0.00464928 eta = 0.584
418 E = 0.00463747 eta = 0.583
419 E = 0.00462583 eta = 0.582
420 E = 0.00461424 eta = 0.581
421 E = 0.00460269 eta = 0.58
422 E = 0.00459127 eta = 0.579
423 E = 0.00457992 eta = 0.578
424 E = 0.00456865 eta = 0.577
425 E = 0.00455751 eta = 0.576
426 E = 0.00454626 eta = 0.575
427 E = 0.00453542 eta = 0.574
428 E = 0.00452449 eta = 0.573
429 E = 0.00451362 eta = 0.572
430 E = 0.00450261 eta = 0.571
431 E = 0.00449208 eta = 0.57
432 E = 0.0044814 eta = 0.569
433 E = 0.00447096 eta = 0.568
434 E = 0.00446037 eta = 0.567
435 E = 0.00445005 eta = 0.566
436 E = 0.00443965 eta = 0.565
437 E = 0.00442932 eta = 0.564
438 E = 0.00441924 eta = 0.563
439 E = 0.00440908 eta = 0.562
440 E = 0.00439908 eta = 0.561
441 E = 0.00438907 eta = 0.56
442 E = 0.00437918 eta = 0.559
443 E = 0.0043692 eta = 0.558
444 E = 0.00435941 eta = 0.557
445 E = 0.00434981 eta = 0.556
446 E = 0.00434017 eta = 0.555
447 E = 0.0043306 eta = 0.554
448 E = 0.00432109 eta = 0.553
449 E = 0.00431158 eta = 0.552
450 E = 0.00430218 eta = 0.551
451 E = 0.00429294 eta = 0.55
452 E = 0.00428361 eta = 0.549
453 E = 0.00427444 eta = 0.548
454 E = 0.0042653 eta = 0.547
455 E = 0.00425622 eta = 0.546
456 E = 0.00424724 eta = 0.545
457 E = 0.00423825 eta = 0.544
458 E = 0.00422939 eta = 0.543
459 E = 0.00422048 eta = 0.542
460 E = 0.00421169 eta = 0.541
461 E = 0.004203 eta = 0.54
462 E = 0.00419424 eta = 0.539
463 E = 0.00418577 eta = 0.538
464 E = 0.00417721 eta = 0.537
465 E = 0.00416868 eta = 0.536
466 E = 0.00416026 eta = 0.535
467 E = 0.00415182 eta = 0.534
468 E = 0.00414355 eta = 0.533
469 E = 0.0041352 eta = 0.532
470 E = 0.00412703 eta = 0.531
471 E = 0.00411885 eta = 0.53
472 E = 0.00411076 eta = 0.529
473 E = 0.00410268 eta = 0.528
474 E = 0.00409466 eta = 0.527
475 E = 0.00408663 eta = 0.526
476 E = 0.00407884 eta = 0.525
477 E = 0.00407097 eta = 0.524
478 E = 0.00406313 eta = 0.523
479 E = 0.0040554 eta = 0.522
480 E = 0.00404765 eta = 0.521
481 E = 0.00404003 eta = 0.52
482 E = 0.00403239 eta = 0.519
483 E = 0.00402482 eta = 0.518
484 E = 0.00401731 eta = 0.517
485 E = 0.00400984 eta = 0.516
486 E = 0.00400238 eta = 0.515
487 E = 0.00399497 eta = 0.514

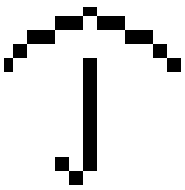
4.2 Fase di Esecuzione

Abbiamo introdotto sulle immagini vari tipi di rumore: aggiungendo pixel bianchi (es. cuore ed ombrello) o neri (es. nota ed ombrello), sovrapponendo due immagini diverse (es. nota e rombo), componendo un'immagine da due iniziali (esempio dell'uomo-donna), distorcendone la forma (esempio della donna e della casa); infine, creando una nuova forma da apprendere (diamante).

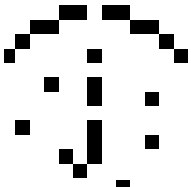
I risultati sono valutabili direttamente nelle pagine seguenti.

fig. Training Set

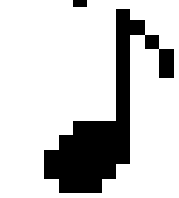




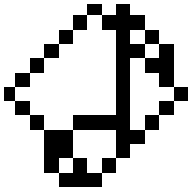
L'immagine somiglia ad un(a) ombrello



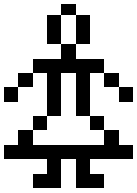
L'immagine somiglia ad un(a) ombrello



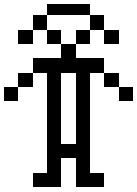
L'immagine somiglia ad un(a) nota



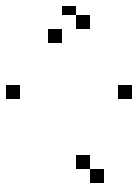
L'immagine sembra vagamente un(a) rombo
E' una nuova forma (s/n)?
n



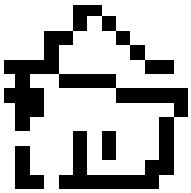
L'immagine sembra vagamente un(a) uomo
E' una nuova forma (s/n)?
n



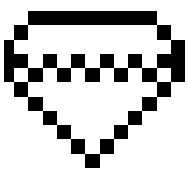
L'immagine sembra vagamente un(a) donna
E' una nuova forma (s/n)?
n



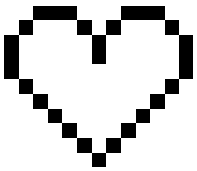
L'immagine sembra vagamente un(a) non so cosa
E' una nuova forma (s/n)?
n



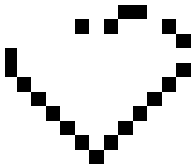
L'immagine sembra vagamente un(a) casa
E' una nuova forma (s/n)?
n



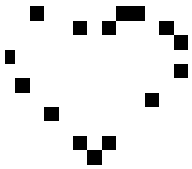
L'immagine sembra vagamente un(a) cuore
E' una nuova forma (s/n)?
s
Come si chiama ?
diamante
L'ho inserita nel mio insieme di apprendimento.
E' necessario riaddestrare la rete.



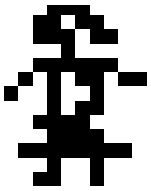
L'immagine somiglia ad un(a) cuore



L'immagine somiglia ad un(a) cuore



L'immagine sembra vagamente un(a) cuore
E' una nuova forma (s/n)?
n



L'immagine sembra vagamente un(a) donna
E' una nuova forma (s/n)?
n

5. BIBLIOGRAFIA

[Buttazzo, ????] Giorgio Buttazzo: *Introduzione alle Reti Neuronali*, ARTS Lab – SS S.Anna – Pisa.

[Appunti, 2001] Appunti del Corso di Ingegneria della Conoscenza e Sistemi Esperti, Pisa 2001.

[Dorbolò-Frosini-Lazzerini, 2000] Daniela Dorbolò, Graziano Frosini, Beatrice Lazzerini: *Programmazione a oggetti con riferimento al C++,* FrancoAngeli – Milano 2000.

[Domenici-Frosini, 1996] Andrea Domenici, Graziano Frosini: *Introduzione alla programmazione ed elementi di strutture dati con il linguaggio C++*, FrancoAngeli – Milano 1996.

[Stroustrup, 2000] Bjarne Stroustrup: *C++, linguaggio, libreria standard, principi di programmazione*, Addison-Wesley – Milano 2000.

[Domenici, 1998] Andrea Domenici: *Appunti per le lezioni di Ingegneria del Software*, Pisa 1998.

[UML v1.3, 1999] AA.VV.: *UML Notation Guide, 1999*