

UNIVERSITÀ
DELLA CALABRIA

FACOLTA' DI INGEGNERIA INFORMATICA

Laurea Magistrale - A.A. 2015/2016

**PROGETTO DIDATTICO
TEORIA DELL'INFORMAZIONE**

Modello di riservazione passiva di banda durante il fenomeno
dell'hand-over basato su predizione del path effettuato con Reti Neurali

Matteo Calabrese 174967

Antonio Caliò 175056

Valerio Russo 175057

ABSTRACT

Proporre un modello di **riservazione passiva di banda** sulle celle wireless presenti su un territorio, basato su **predizione del path** dell'utente attraverso l'impiego di **reti neurali**.

OBIETTIVI:

- Fornire una stima delle possibili celle attraversate quanto più accurata possibile.
- Evitare la riservazione di celle che non saranno attraversate
- Evitare cadute di connessione durante il fenomeno dell'hand-over

ANALISI DEI REQUISITI e STRUTTURA BASE

Funzioni e rappresentazione:

L'applicazione, deve **predire**, sulla base dei **dati statistici** forniti dal software di simulazione. Si suppone che sulla mappa siano presenti delle celle, che rispecchiano la copertura wireless delle torri di comunicazione, rappresentate con forma esagonale.

Il sistema sarà dunque il grado di **prenotare passivamente le risorse sulle torri** adiacenti per far sì che il fenomeno dell'hand-over non comporti cadute di connessione.

Tecnologie implementative:

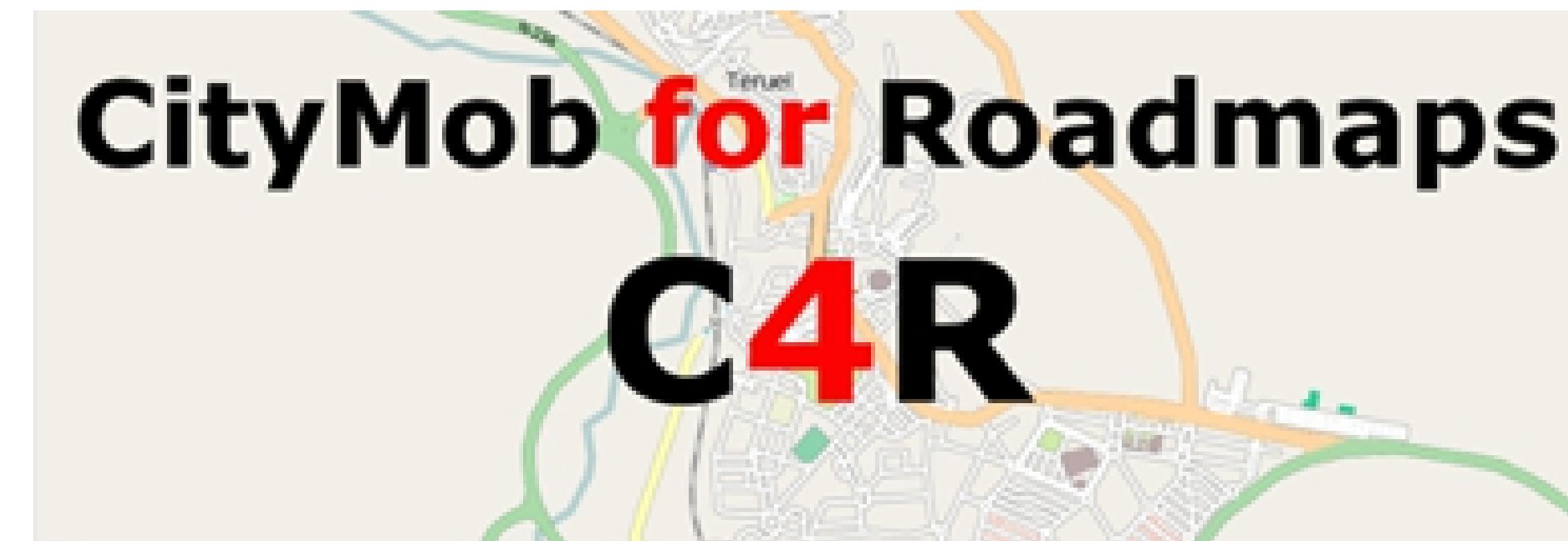
ENCOG: framework di lavoro con cui modellare le reti neurali

JAVA: linguaggio di programmazione

DATI STATISTICI E DI MOBILITÀ

I dati su cui si basa l'applicazione, sono stati ottenuti dal software: **C4R**.
Citymob per tabelle di marcia è un **generatore di mobilità per le reti veicolari**.

Ciò significa che è un programma che permette di *simulare* il traffico in differenti posizioni utilizzando mappe reali ottenute da open street map.



DATI STATISTICI E DI MOBILITÀ'

Le porzioni di mappe scelte, comprendono sempre un'area urbana della dimensione di circa $1,3 \text{ km}^2$

Per le simulazioni sono generate **due tracce**, generalmente da **2000 veicoli** e senza tenere conto del parametro che modella gli agglomerati urbani.

Il modello seguito è quello elaborato da Krauß con parametri di accelerazione, sigma e tau standard.

Il tempo di simulazione scelto è di 600s.

La visualizzazione della simulazione avviene attraverso il plug-in esterno **SUMO NetConvert**

DATI STATISTICI E DI MOBILITÀ'

Gli output della simulazione di nostro interesse sono dunque:

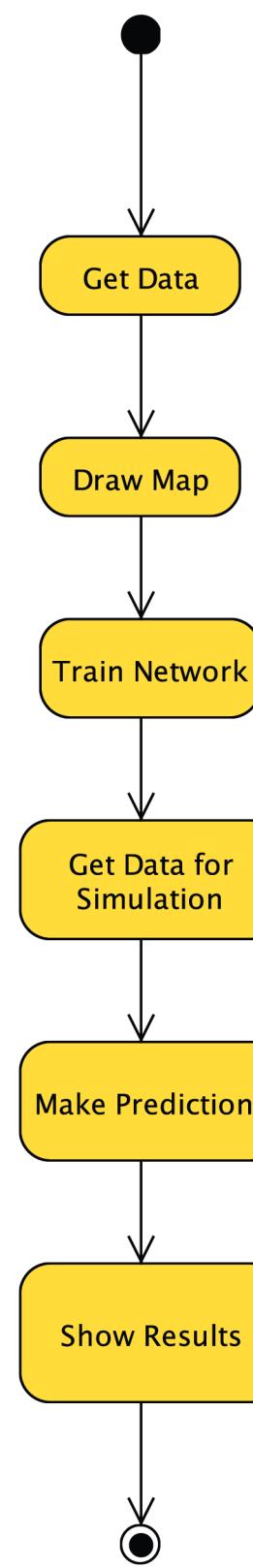
1. *Training.tcl* che contiene le tracce per l'addestramento della rete neurale
2. *Testing.tcl* che contiene le tracce per il test dell'applicazione
3. *Map.net.xml* che contiene i dati per ricostruire la mappa

```
1  #
2  # This file was parsed with Citymob for Roadmaps (C4R) version 1.0
3  #
4
5  #Node 0 = random_undefined_0_0
6  $node_(0) set X_ 816.96
7  $node_(0) set Y_ 812.65
8  $node_(0) set Z_ 0.0
9
10 #Path $node_(0) = random_undefined_0_0
11 $ns_ at 0.0 "$node_(0) setdest 817.7620909838289 811.9119281514239 1.0900000000000123"
12 $ns_ at 1.0 "$node_(0) setdest 819.1970243952659 810.5915243856227 1.9500000000000381"
13 $ns_ at 2.0 "$node_(0) setdest 821.581221448115 808.3976227439839 3.239999999999996"
14 $ns_ at 3.0 "$node_(0) setdest 824.9514753067721 805.296366719692 4.579999999999999"
15 $ns_ at 4.0 "$node_(0) setdest 828.8662679984873 801.6940343945319 5.320000000000009"
16 $ns_ at 5.0 "$node_(0) setdest 833.7303243316146 797.2182041935341 6.609999999999675"
17 $ns_ at 6.0 "$node_(0) setdest 839.4332648129666 791.9704456371448 7.750000000000024"
18 $ns_ at 7.0 "$node_(0) setdest 845.9971653411808 785.9304448212749 8.919999999999968"
19 $ns_ at 8.0 "$node_(0) setdest 853.0908874366947 779.402910307263 9.639999999999974"
20 $ns_ at 9.0 "$node_(0) setdest 860.824810592696 772.2862725746629 10.510000000000044"
```

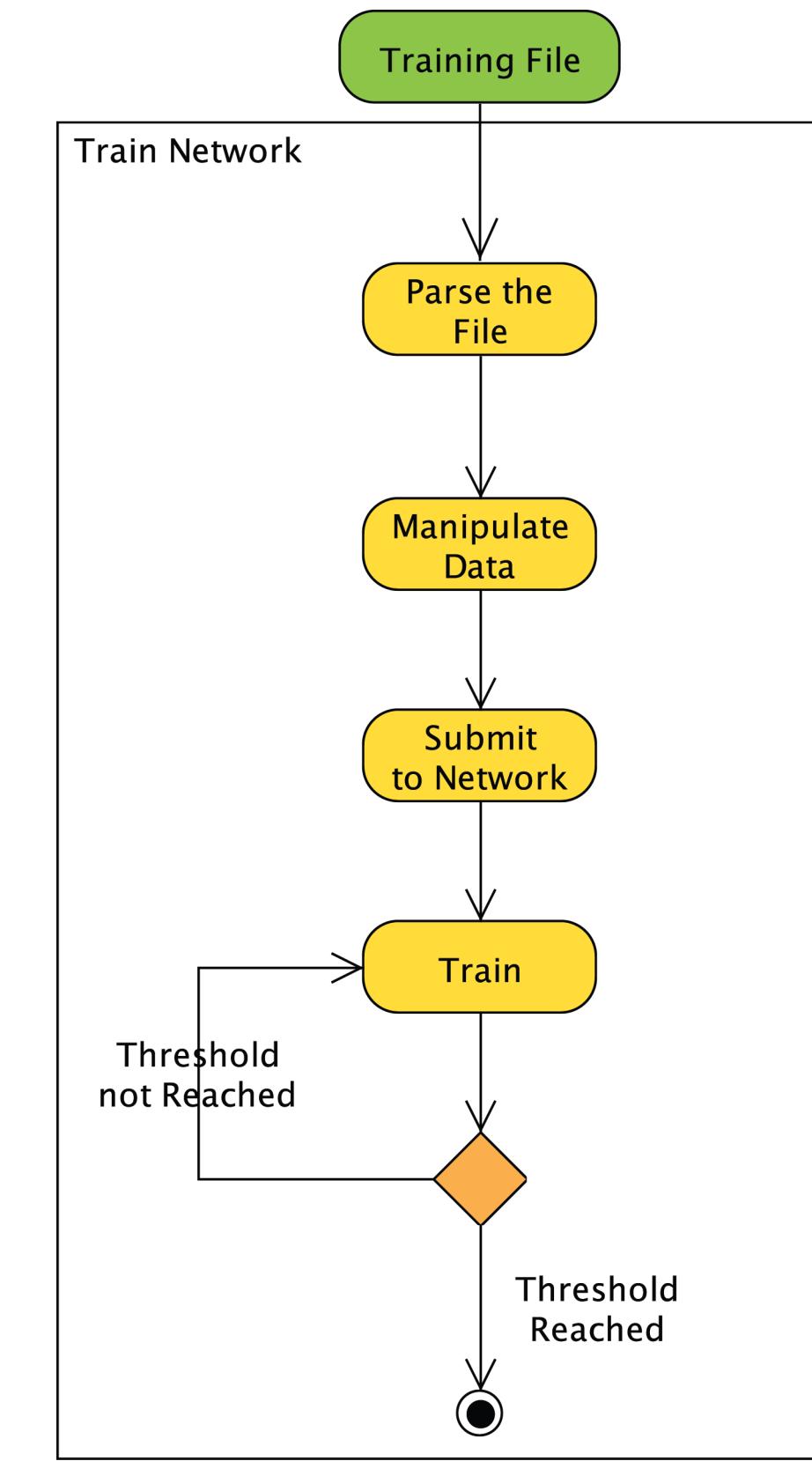
Le prime 3 righe rappresentano la posizione iniziale nello spazio, mentre le successive righe sono le posizioni del veicolo nello spazio ai vari step temporali.

FASE PROGETTUALE

Analizziamo ora, la progettazione dell'applicazione, definendo dettagli e funzionalità attraverso i diagrammi:



**diagramma
workflow in
generale**



**diagramma
workflow di addestramento
delle reti neurali**

FASE PROGETTUALE

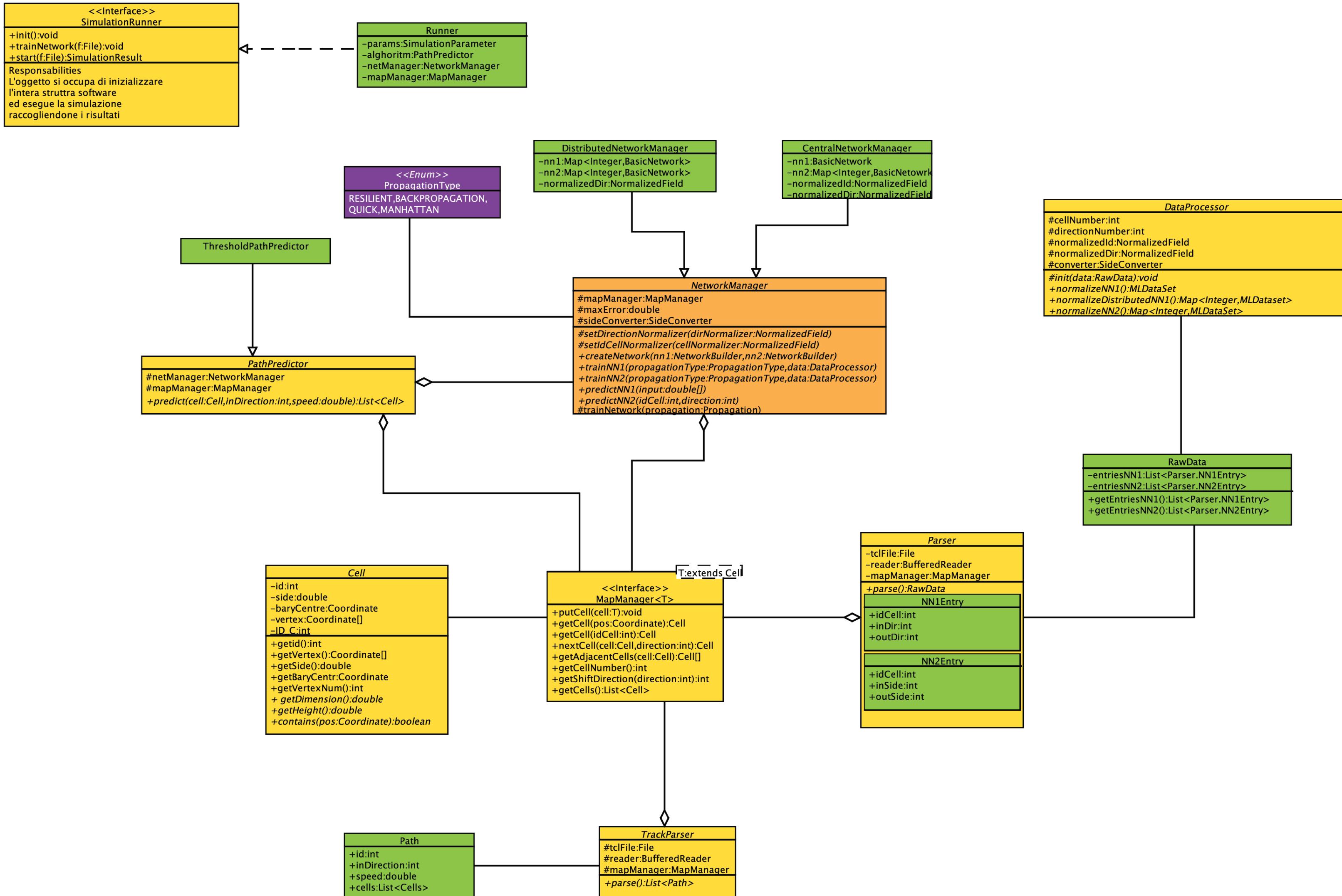


diagramma delle classi

GIALLO = interfaccia / classe astratta

VERDE = classe
concreta

VIOLA = enum

ARANCIONE =
principale classe
astratta della parte
logica

FASE PROGETTUALE

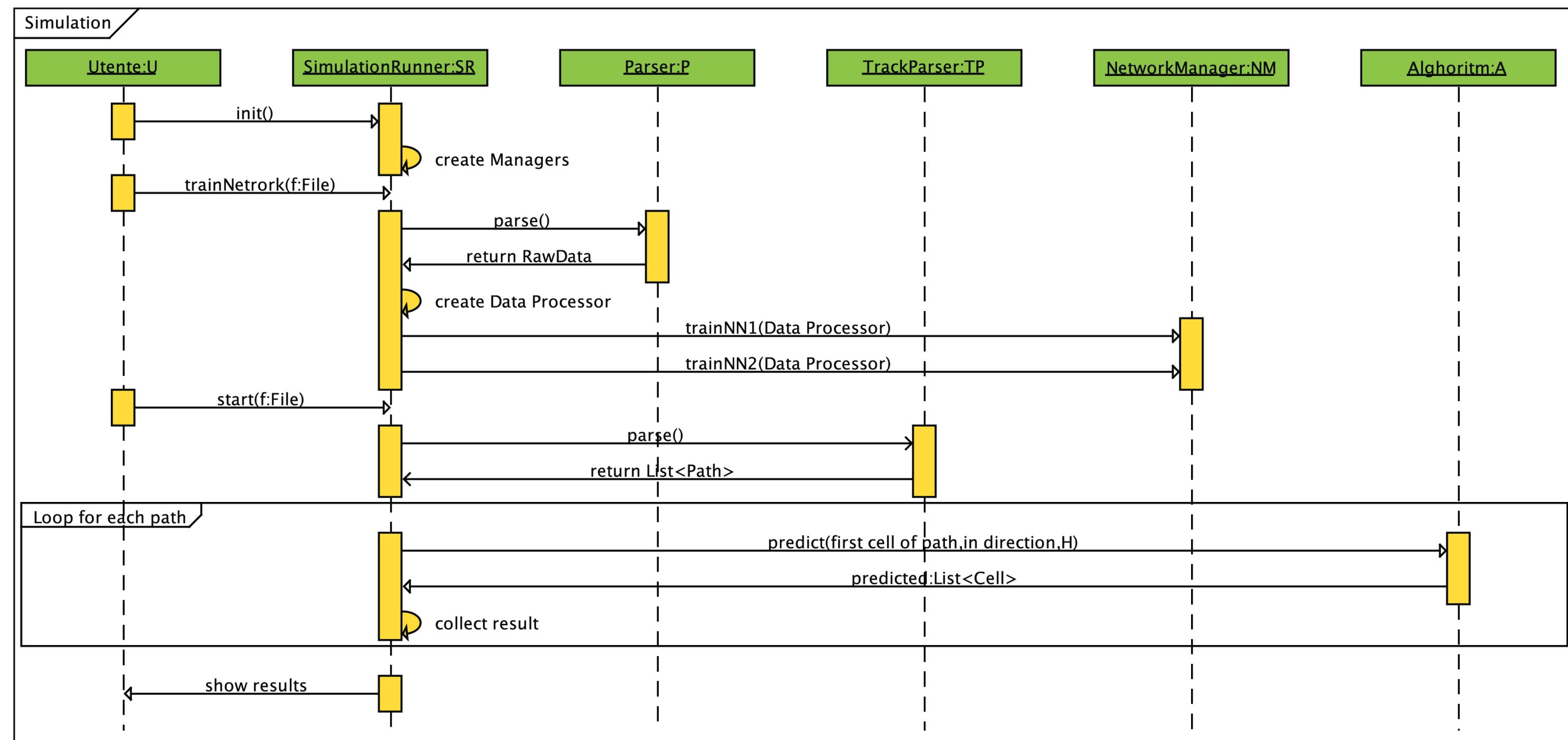


diagramma di sequenza
mostra le interazioni che sussistono tra le classi precedentemente elencate ed indica come il flusso dati è orchestrato.

STRUTTURA DELLE RETI NEURALI

Nell'ambito di questo progetto sono state utilizzate **due tipi di rete neurale**, Sono dunque implementate due soluzioni che chiameremo:

- **NN1 (neural network 1)**
- **NN2 (neural network 2)**

La differenza tra le due, oltre che per lo scopo, sta nel tipo di input. Infatti la rete neurale 1, ha in input 1 o 2 parametri a seconda dell'approccio utilizzato che può essere di tipo:

- **DISTRIBUITO (1 neurone in input)**
- **CENTRALIZZATO (2 neuroni in input)**

STRUTTURA DELLE RETI NEURALI

DIFFERENZE:

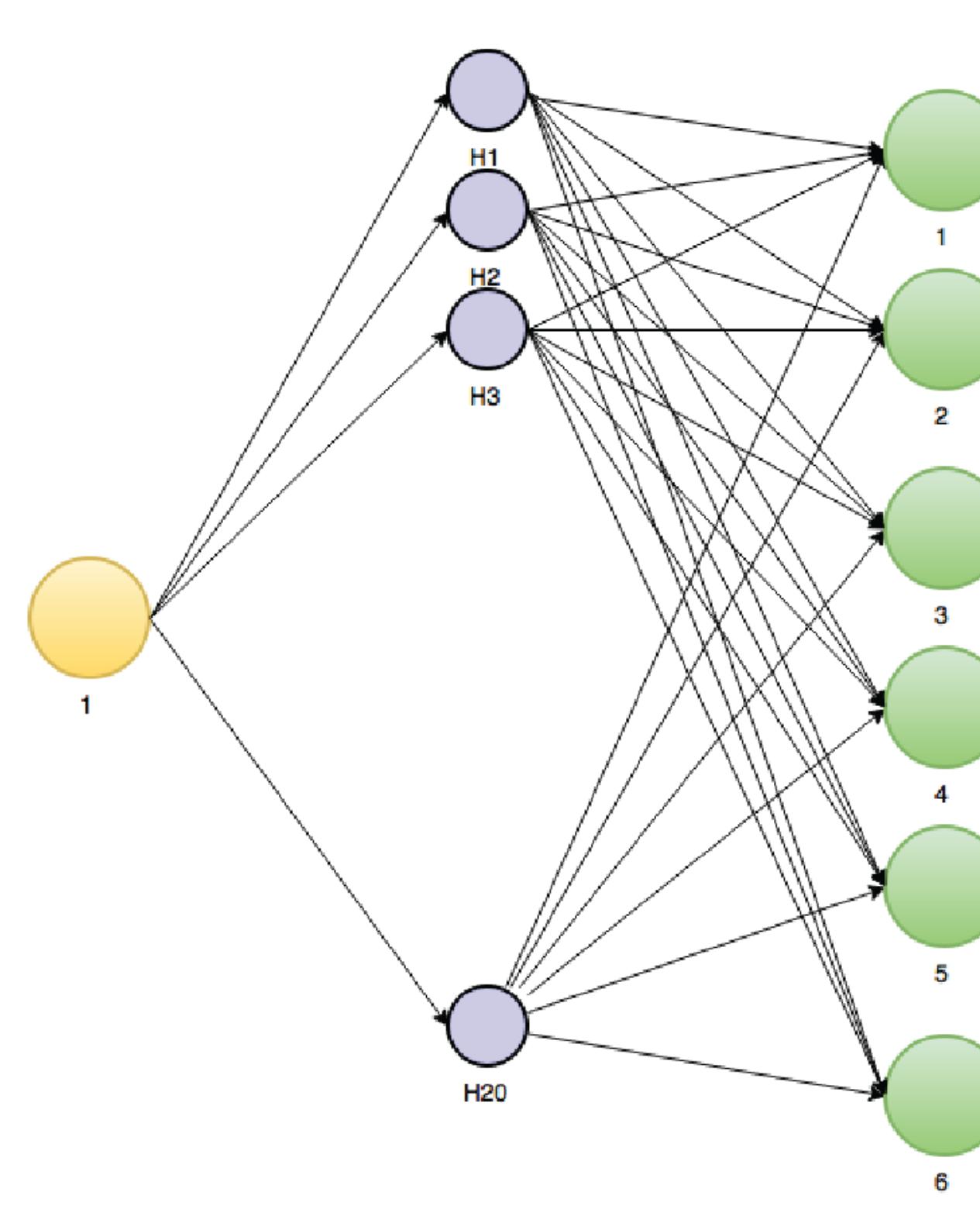
- **NN1**: caratterizza le celle dove nascono gli utenti
- **NN2**: ha 6 neuroni in input e descrive le celle in cui gli utenti transitano

Entrambe le reti hanno 6 neuroni di output che nient'altro sono che il numero di lati della cella esagonale.

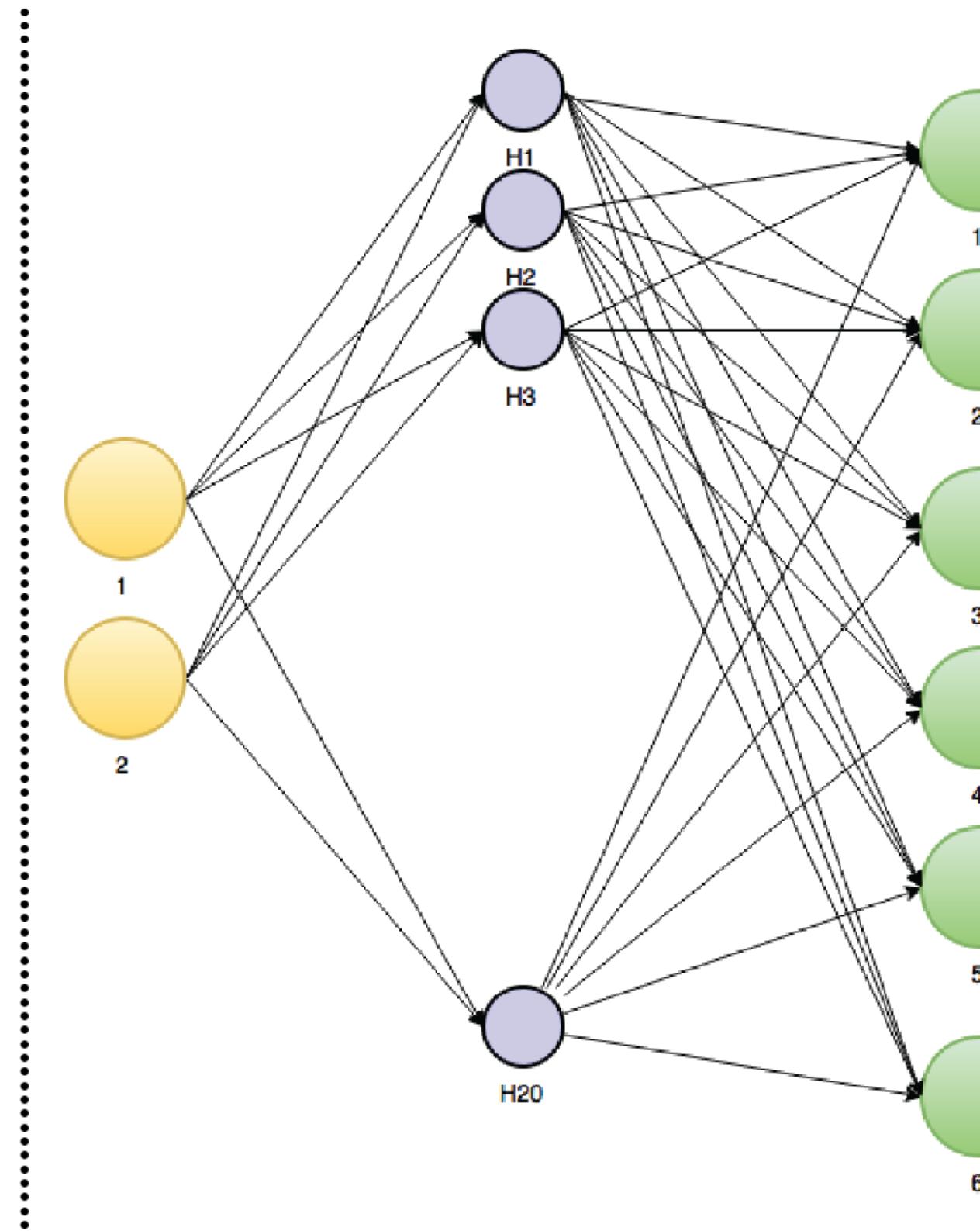
L'approccio centralizzato: prevede la presenza di un'unica rete neurale che gestisce l'intero sistema, vale a dire che esiste una sola rete per tutte le celle presenti. Prende in input 2 parametri: l'id della cella e la direzione.

L'approccio distribuito: invece prevede la presenza di una rete neurale per ogni cella del sistema. Prende in input solo la direzione.

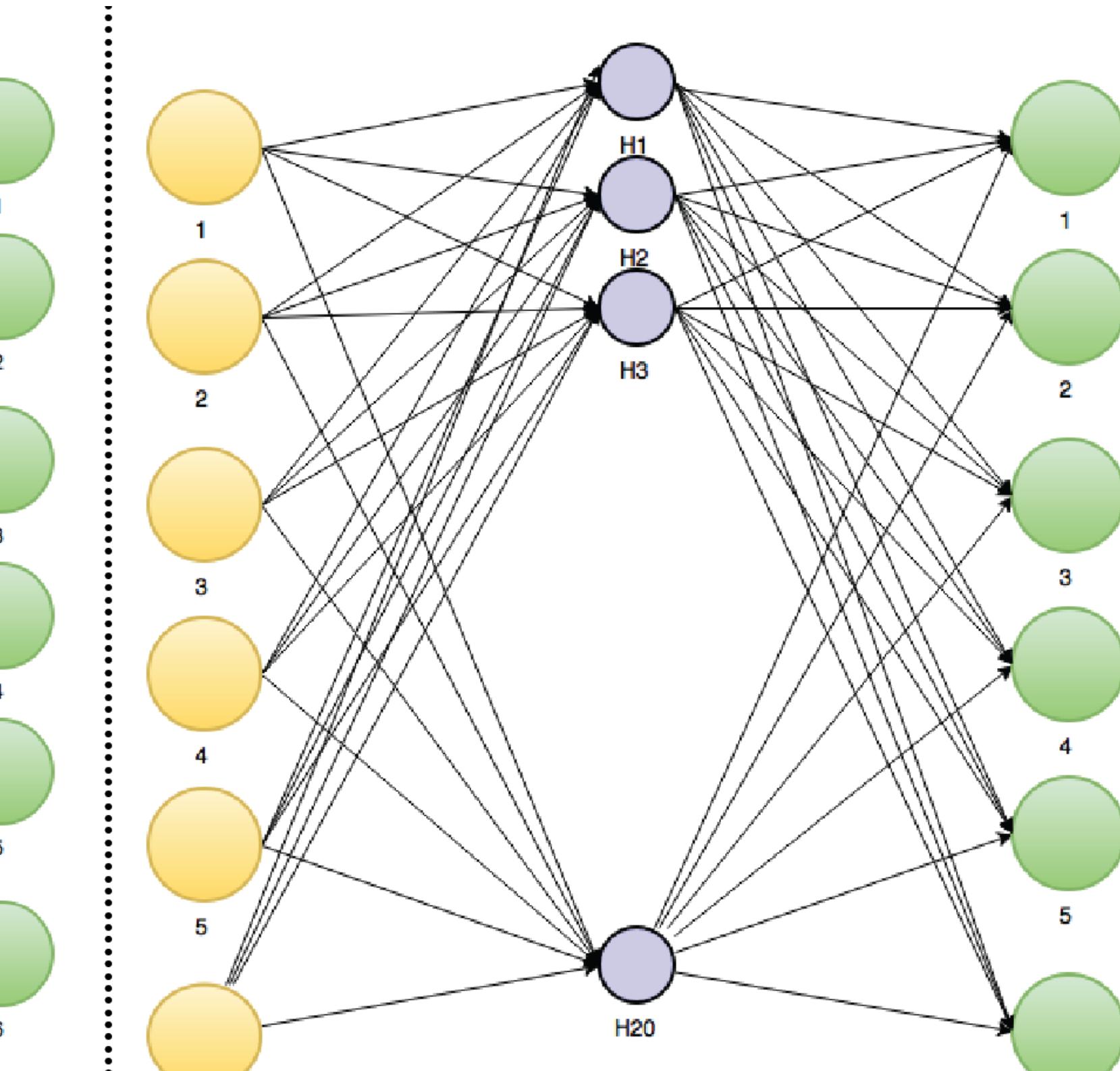
STRUTTURA DELLE RETI NEURALI



NN1 con 1 input (direzione)



NN1 con 2 input (dir, id_cellula)



NN2

PROPRIETA' DELLE RETI NEURALI

Sono stati previsti 4 differenti *algoritmi di addestramento/apprendimento* supervisionati:

1. **RESILIENT PROPAGATION**
2. **BACKPROPAGATION**
3. **MANHATTAN PROPAGATION**
4. **QUICK PROPAGATION**

Lo strato più interno, detto *hiddenlayer* contiene **20 neuroni**. Tale numero è frutto di un trade-off tra efficienza e complessità.

La funzione di attivazione tra i neuroni è la *Sigmoide* che è continua e derivabile e garantisce sempre un minimo ed un massimo locale.

DATI DI ADDESTRAMENTO

Per lanciare gli algoritmi di propagazione, è necessario fornire dei dati ben strutturati ed organizzati sulla base del layout della rete.

Le classi di `network.data` e nella fattispecie:

- **Parser**
- **TrackParser**
- **DataProcessor**
- **BasicData Processor**
- **NN2MLDataPair**

mettono a disposizione tutte le funzionalità necessarie per leggere il file con le tracce, estrapolare i dati per addestrare le due reti neurali ed infine memorizzarli nella struttura dati richiesta (matrice di double).

DATI DI ADDESTRAMENTO

Distinguiamo tre tipologie di strutture:

- 1. Struttura per NN1 DISTRIBUITA:** per ogni traccia, si considera la direzione di avanzamento dell'utente: [direzione]
- 2. Struttura per NN1 CENTRALIZZATA:** per ogni dato della traccia, si risale alla cella in cui ricade e si individua la direzione di avanzamento dell'utente: [i-d_cell,direzione]
- 3. Struttura per NN2:** essendo celle di transito, si risale al lato di ingresso dell'utente nella cella, e si definisce l'array. Si hanno dunque 6 valori nel caso di celle esagonali: [lato1,lato2,lato3,lato4,lato5,lato6]

I dati sono poi oggetto di un lavoro di normalizzazione in un dato range [-1,1].

L'ALGORITMO DI PREDIZIONE

```
Init) Determine the number H of probably visited cells;  
      Determine  $bs_o$ ,  $ap_o$  and  $d_o$  for user x; set  $h=0$ ;  
Step 0) Create an input pattern  $X_0^{NN1} = (bs_o, ap_o, d_o)$ ;  
Step 1) Activate NN1 with  $X_0^{NN1}$  and store the output  $Y_i$ ;  
Step 2) While ( $h < H$ )  
    Step 3) Evaluate from  $Y_i$  the probabilities of each near cell for the  $i$ -th step of the prediction;  $h++$ ;  
    Step 4) For each probability value that is higher than a fixed threshold  $T$ , do steps 5-10;  
        Step 5) Take the direction  $d_{ij}$  associated with  $Y_i[j]$ ;  
        Step 6) Ask the network to obtain the identifiers  $bs_i$ ,  $ap_i$  of the adjacent cell on direction  $d_{ij}$ ;  
        Step 7) Make the passive reservation on  $bs_i$ ,  $ap_i$ ;  
        Step 8) Create an input pattern  $X_i^{NN2} = (d_{i-p}, d_{i-l+p}, \dots, d_{i-p}, d_i)$  starting from the history of user's direction;  
        Step 9) Activate NN2 with  $X_i^{NN2}$  and store the output  $Y_{i+1}$ ;  
        Step 10) Evaluate from  $Y_{i+1}$  the probabilities of handing-out on the  $m$  different directions;  
End;  
End.
```

Le descrizioni sopra elencate non sono altro che le fondamenta su cui si basa il cuore del progetto, ossia **l'algoritmo di predizione** che è definito dallo pseudocodice adiacente.

Una piccola variante si ha quando si determina un input per la rete NN1 a seconda che si utilizzi un approccio centralizzato o distribuito.

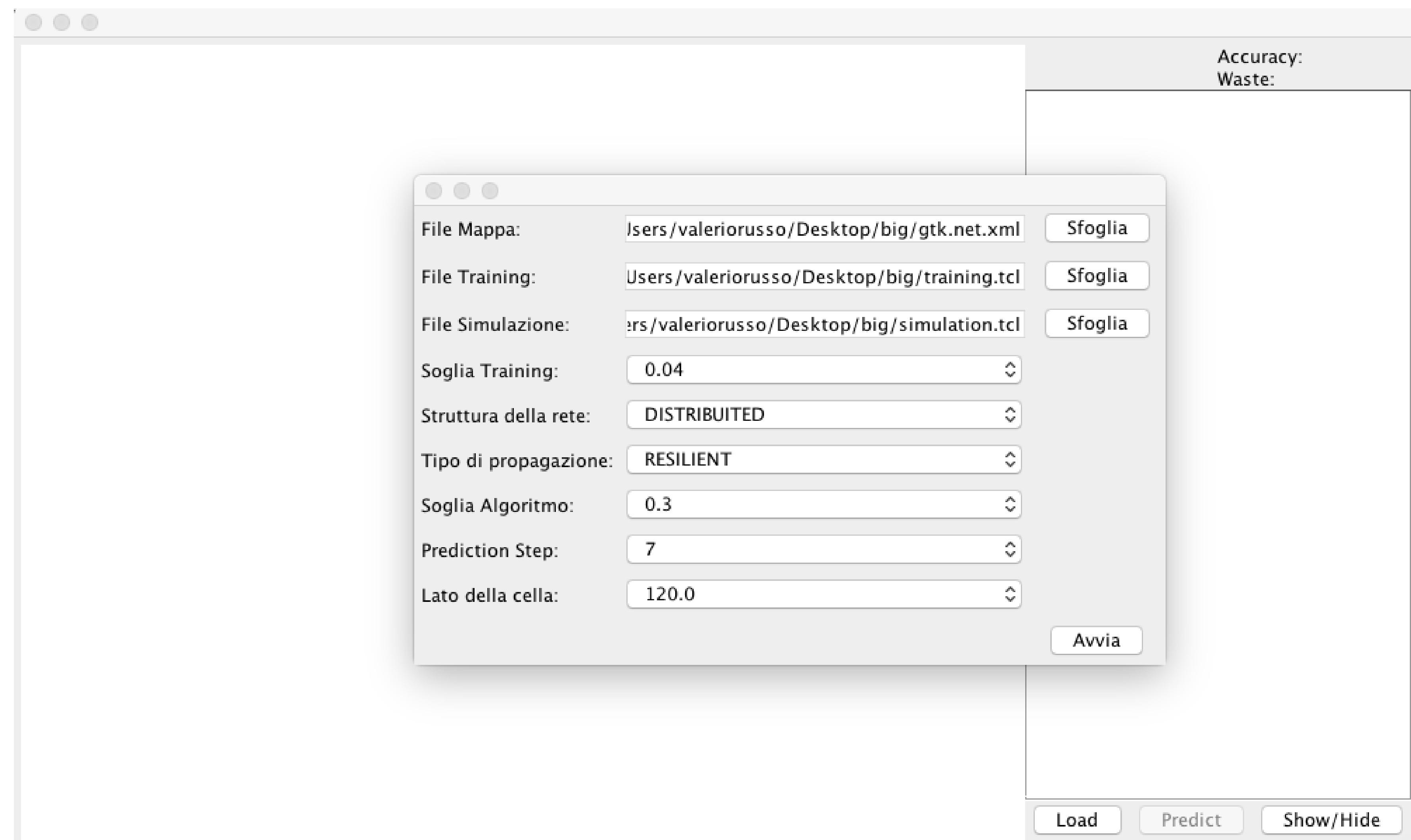
GUI

Per rendere più fruibile l'applicazione e dunque permettere agli utenti di *inserire una propria mappa e dei propri dati*, si è pensato di implementare l'interfaccia grafica del progetto.

Dopo averla lanciata, appare la finestra in cui è possibile caricare i dati e settare tutti i parametri richiesti dalla classe *runner* che fisicamente provvede all'esecuzione della parte logica.

In preset c'è un insieme di parametri ottenuti sulla base della rilevanza dei risultati , scartando dunque quelli che poco significativi.

GUI



GUI

- **Soglia Training:** il valore massimo di errore che l'addestramento della rete neurale può tollerare
- **Struttura della rete:** può essere Distribuita o Centralizzata. La differenza risiede solo nella struttura della rete neurale NN1.
- **Tipo di propagazione:** E' la tipologia di algoritmo utilizzato nella fase di addestramento della rete neurale.
- **Soglia algoritmo:** è il valore che indica la percentuale da soddisfare per la propria predizione.
- **Prediction Step:** rappresenta il valore H dell'algoritmo. E' l'upperbound delle chiamate ricorsive.
- **Lato della cella:** banalmente indica la dimensione del lato della cella

GUI & FASE DI TEST

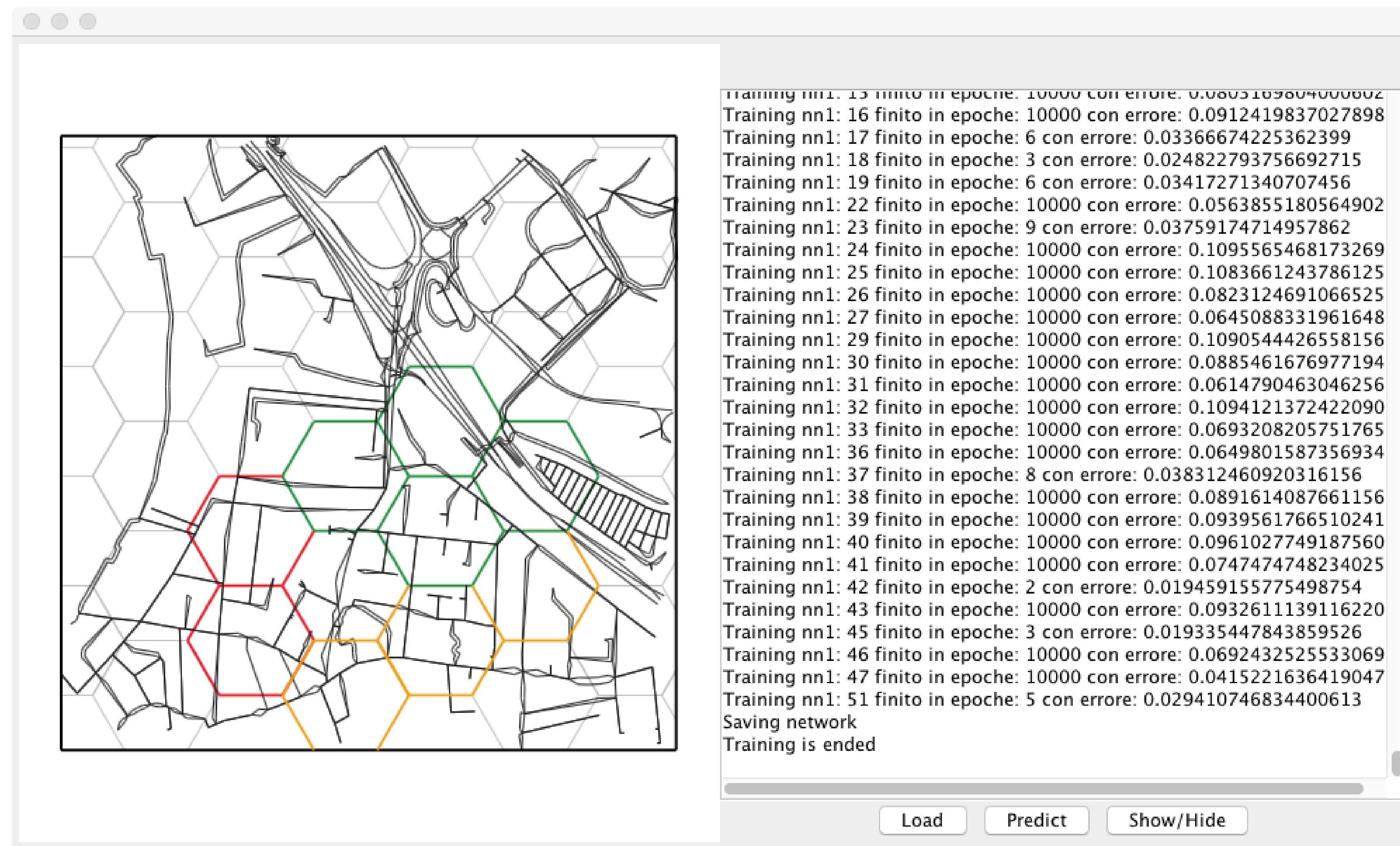
OTTIMALITÀ DELL'APPLICAZIONE

Per ottenere una stima sulla bontà della soluzione implementata, si è pensato di introdurre una classe *SimulationResult* che in abbinamento alle classi *PathMatcher* e *SimulationParameter* fornisce una stima dell'accuratezza e dello spreco. Effettuando un foreach sulle tracce ottenute dal *PathMatcher* si calcolano i parametri di:

- **Accuratezza** effettuando un rapporto tra il numero di celle correttamente predette ed il numero di celle realmente attraversate.
- **Spreco** definito come il rapporto tra il numero di celle sprecate ed il numero di celle predette.

Entrambi i valori sono normalizzati con il numero di tracce derivanti dalla classe *PathMatcher*.

GUI & FASE DI TEST



FASE DI TEST

Dopo aver concluso la fase progettuale, l'algoritmo è stato testato con alcuni dataset creati per l'occorrenza.

Sono stati eseguiti i test su una porzione di mappa della **città di Londra** caratterizzata appunto da **2000 tracce** e generata secondo i parametri indicati precedentemente.

Il file di **testing** è stato però ridotto a **250 tracce** per comodità computazionale.

FASE DI TEST

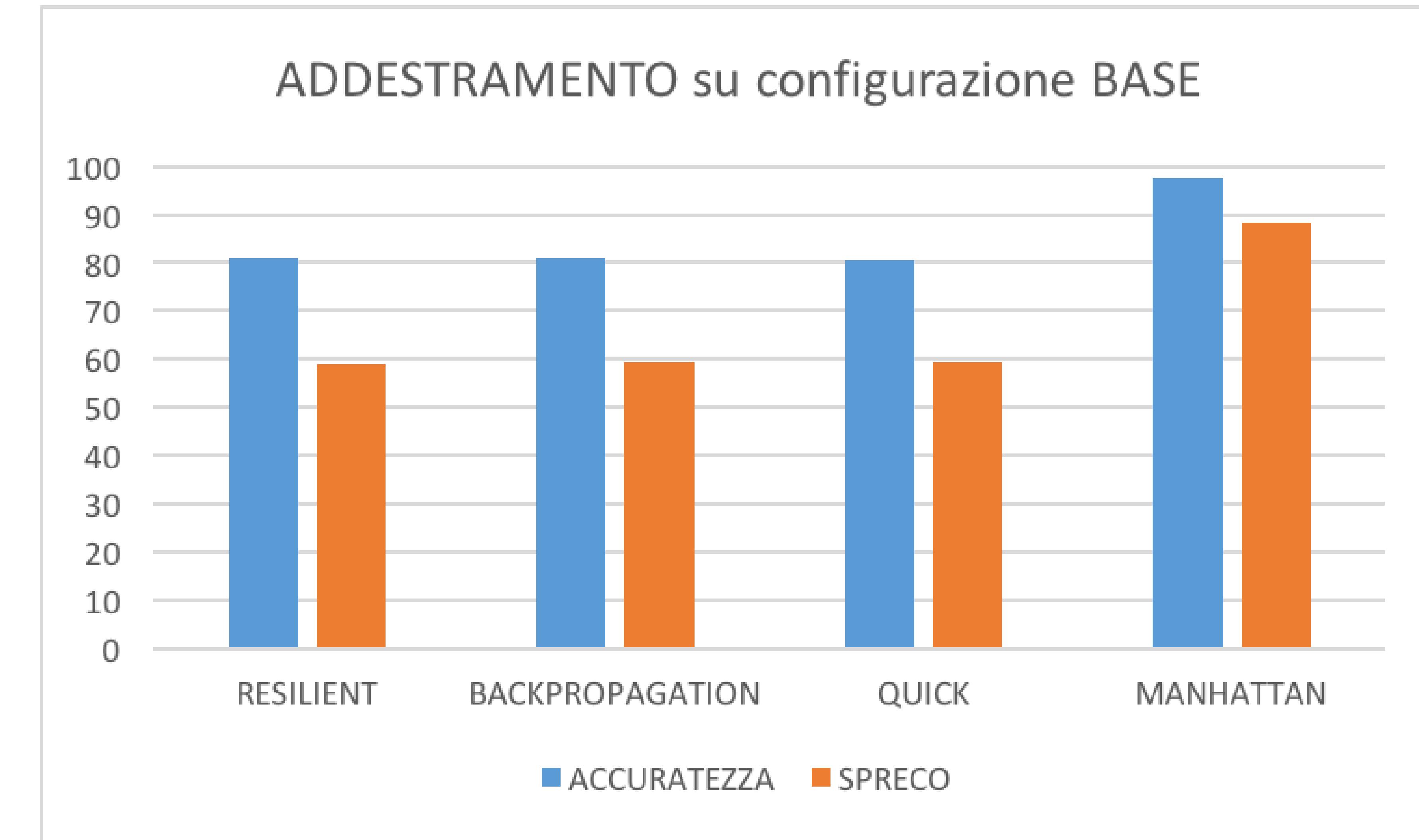
Per fornire una visualizzazione analitica del lavoro effettuato, si è proceduto a rapportare i risultati prodotti attraverso dei grafici.

Nello specifico si è proceduto dapprima a capire quale fosse la configurazione di riferimento, ed in seguito si è provveduto a verificare quale fosse il miglior algoritmo di addestramento.

La configurazione di base scelta è:

- ***Struttura della rete: DISTRIBUITA***
- ***Soglia Training: 0,04***
- ***Soglia dell'algoritmo: 0,30***
- ***Prediction Step: 7***
- ***Lato della cella: 120***

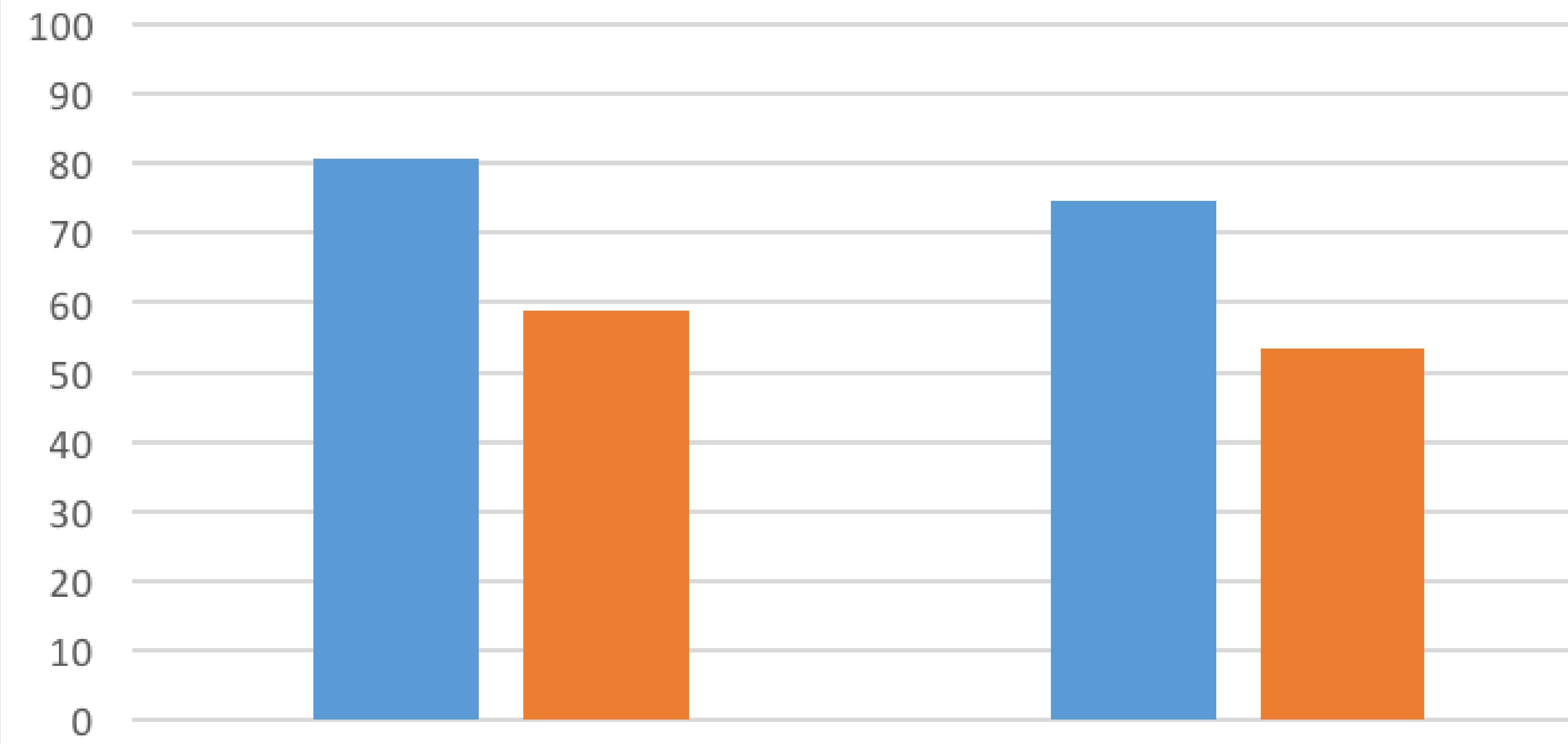
FASE DI TEST - Addestramento



Balza subito agli occhi, che l'algoritmo Manhattan arriva ad un picco di accuratezza pari al 97,58%. Ha però di contro l'alta percentuale di spreco pari al 88,44%.

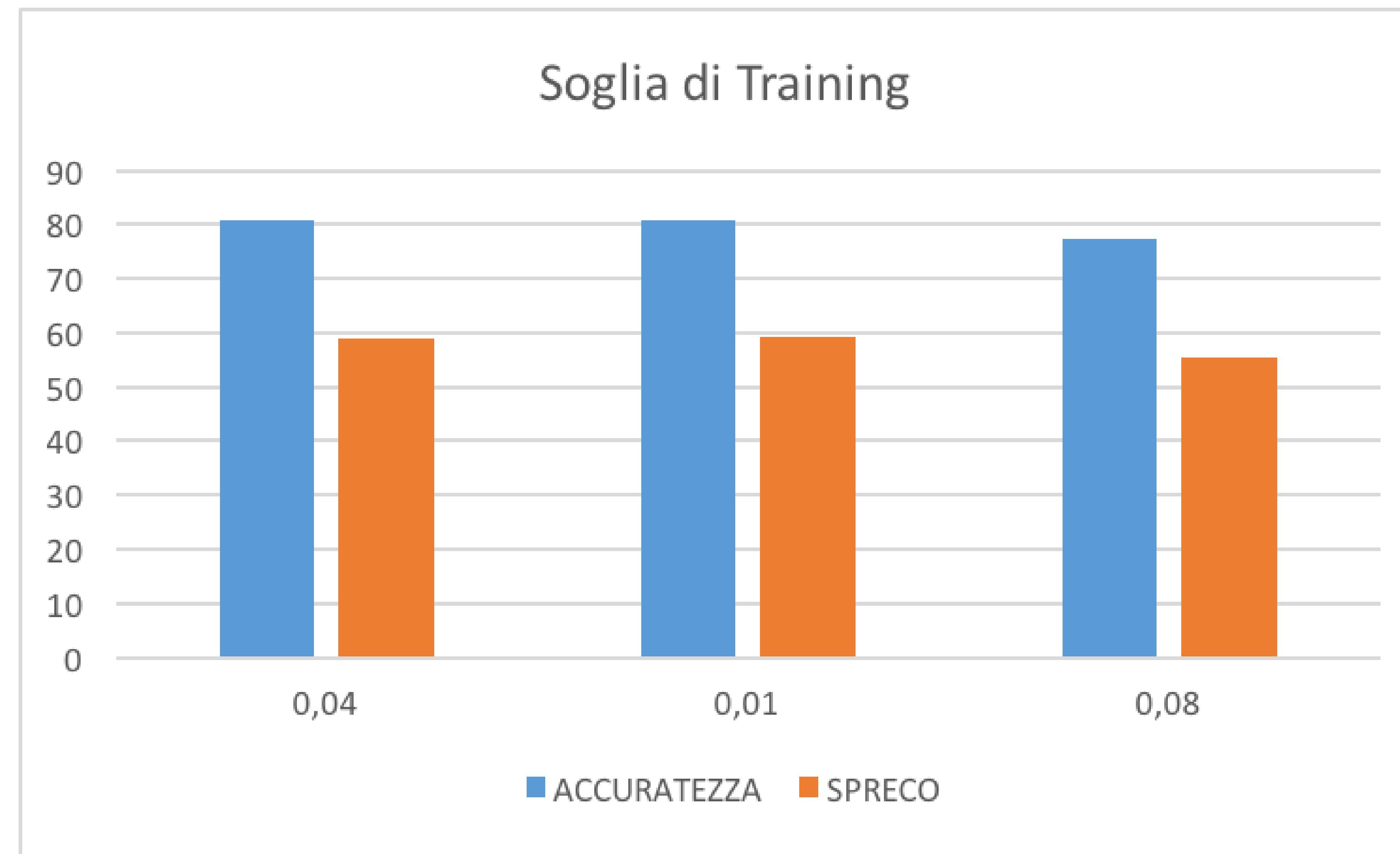
FASE DI TEST - Approccio

DISTRIBUITO VS CENTRALIZZATO

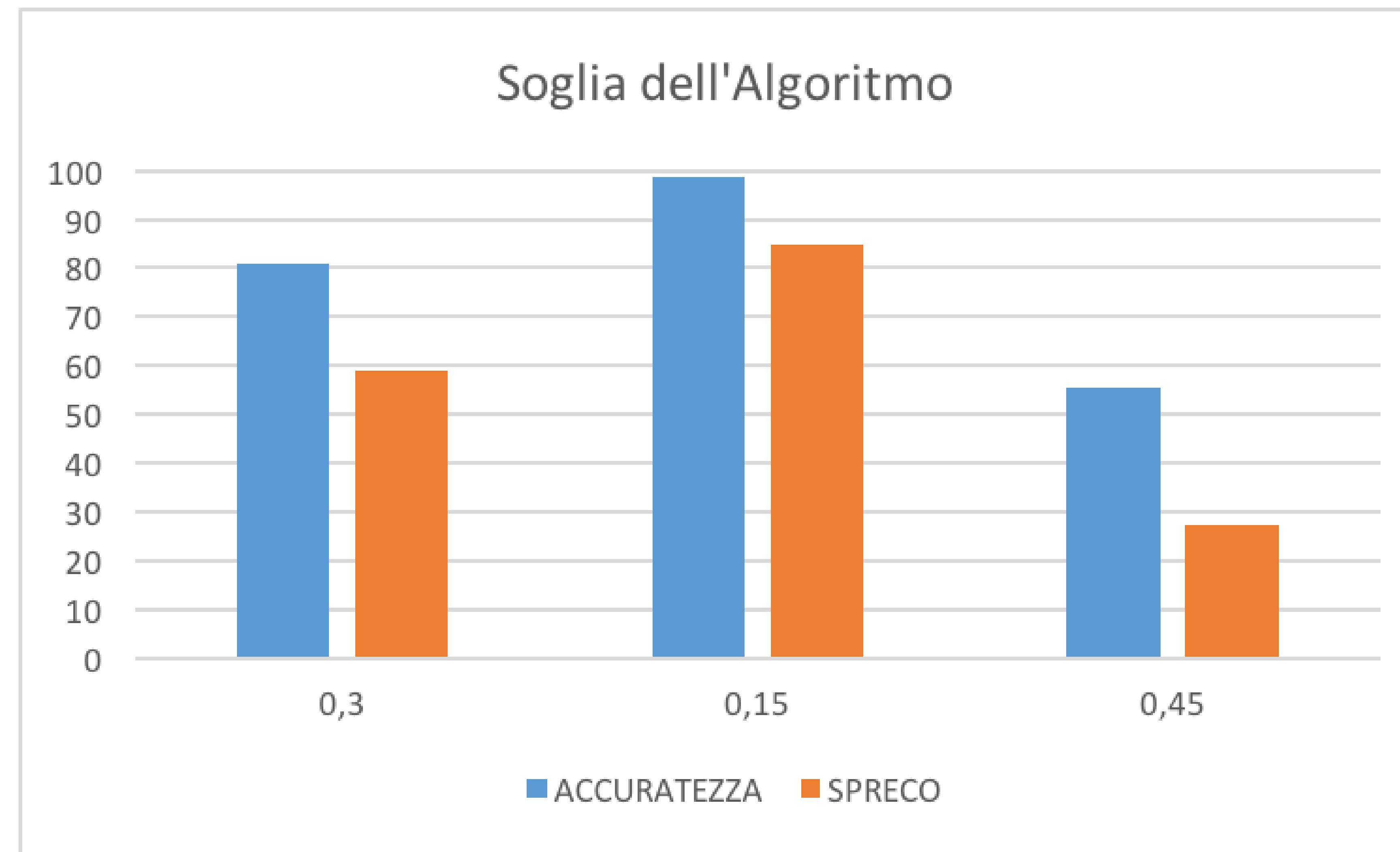


■ ACCURATEZZA ■ SPRECO

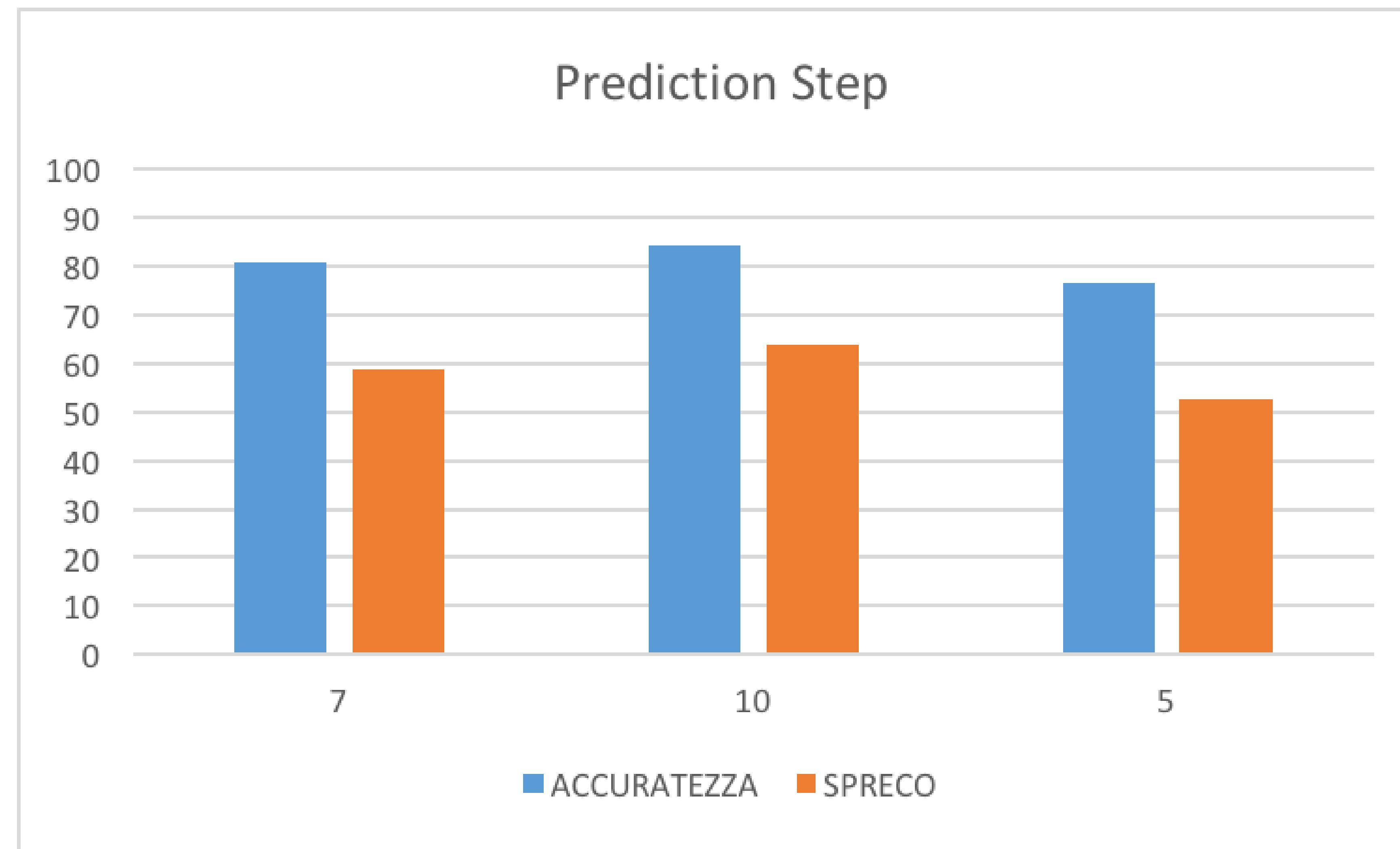
FASE DI TEST - Soglia di Training



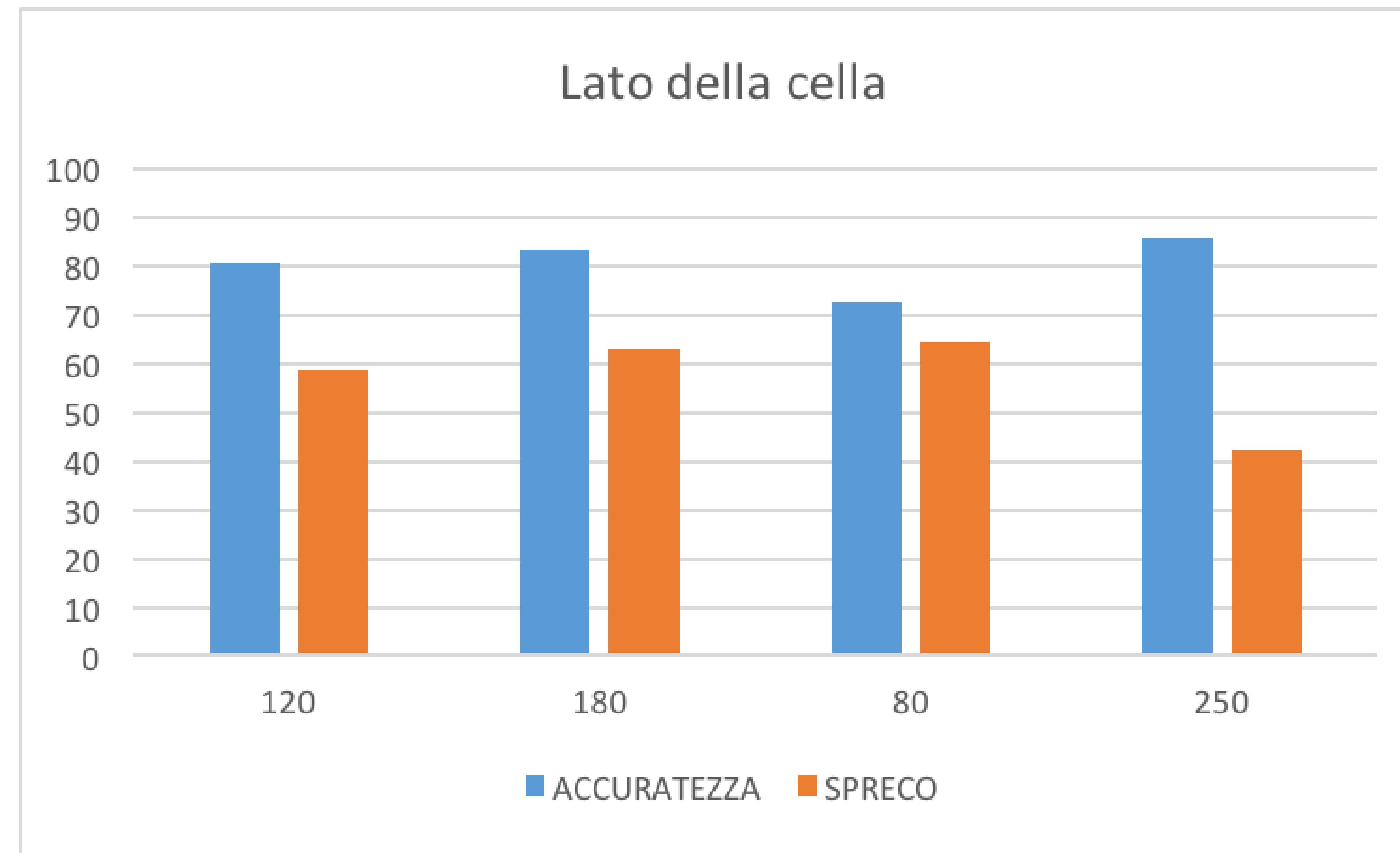
FASE DI TEST - Soglia dell'Algoritmo



FASE DI TEST - Prediction Step



FASE DI TEST - Lato della Cella



FASE DI TEST - Lato della Cella

Settando un lato di 180 e di 80 si osserva un comportamento tipico, ossia nel primo caso una **lieve crescita** dell'accuratezza e dello spreco.

Nel *secondo caso* invece si ha un **decremento** della precisione ed una crescita del numero di celle non utili.

Un discorso differente va invece fatto nel caso in cui il **lato della cella è fissato a 250**.

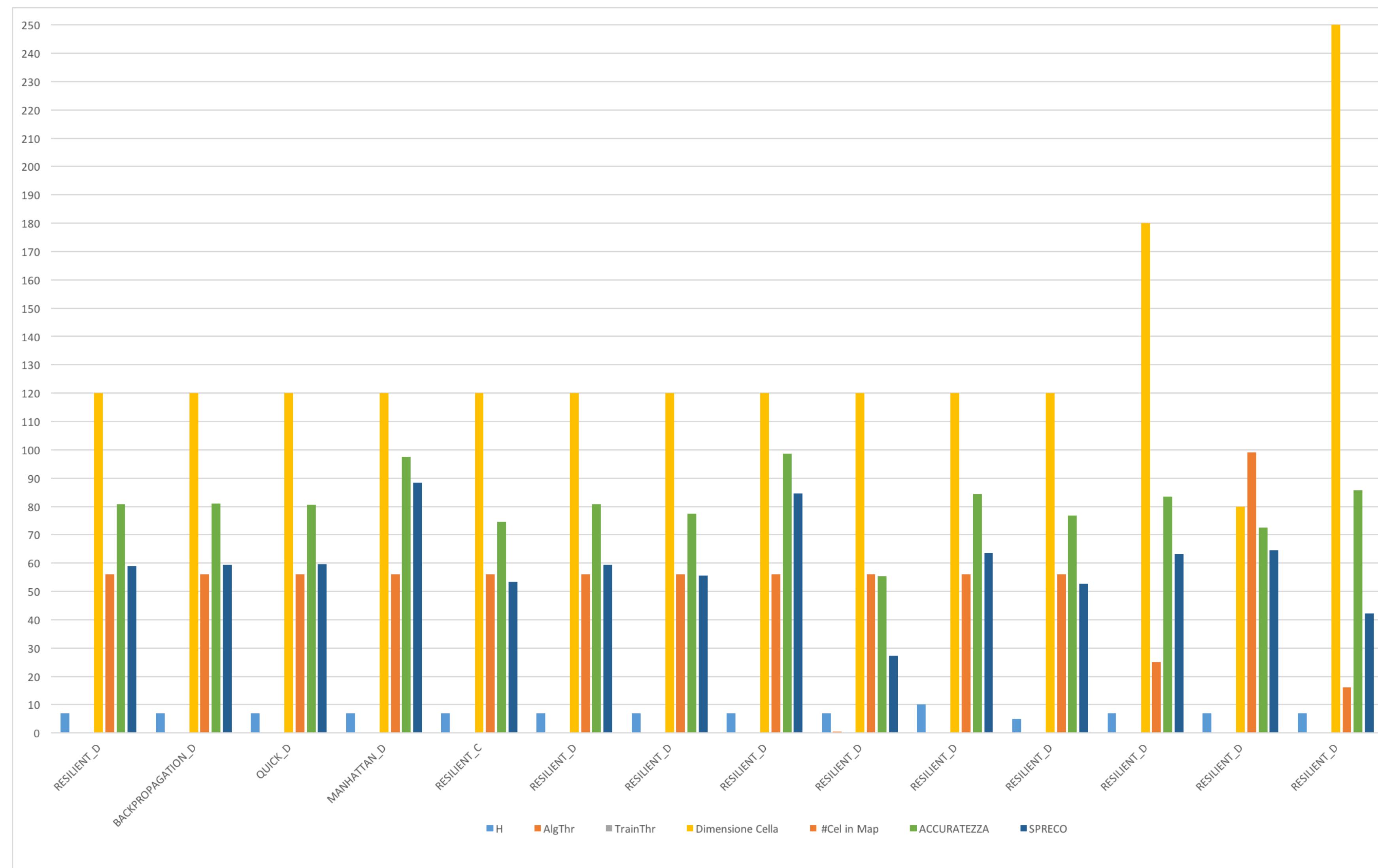
Dal grafico si evince un *piccolo incremento* dell'accuratezza ed una **notevole discesa dello spreco prossima al 40%** contro una media del 60%.

Il motivo risiede sempre nella *numerosità di tracce* adoperate per l'addestramento. E' ovvio quindi che una cella che occupa una maggiore porzione di mappa dispone di più dati per poter effettuare la propria previsione.

FASE DI TEST - Riepilogo

Prop.Type	Approccio	H	AlgThr	TrainThr	Dimensione Cella	#Cel in Map	ACCURATEZZA	SPRECO
RESILIENT_D	DISTRIBUITO	7	0,3	0,04	120	56	80,81	58,9
BACKPROPAGATION_D	DISTRIBUITO	7	0,3	0,04	120	56	81,06	59,26
QUICK_D	DISTRIBUITO	7	0,3	0,04	120	56	80,61	59,51
MANHATTAN_D	DISTRIBUITO	7	0,3	0,04	120	56	97,58	88,44
RESILIENT_C	CENTRALIZZATO	7	0,3	0,04	120	56	74,61	53,3
RESILIENT_D	DISTRIBUITO	7	0,3	0,01	120	56	80,89	59,35
RESILIENT_D	DISTRIBUITO	7	0,3	0,08	120	56	77,35	55,49
RESILIENT_D	DISTRIBUITO	7	0,15	0,04	120	56	98,52	84,67
RESILIENT_D	DISTRIBUITO	7	0,45	0,04	120	56	55,29	27,15
RESILIENT_D	DISTRIBUITO	10	0,3	0,04	120	56	84,24	63,69
RESILIENT_D	DISTRIBUITO	5	0,3	0,04	120	56	76,68	52,73
RESILIENT_D	DISTRIBUITO	7	0,3	0,04	180	25	83,52	63,2
RESILIENT_D	DISTRIBUITO	7	0,3	0,04	80	99	72,6	64,46
RESILIENT_D	DISTRIBUITO	7	0,3	0,04	250	16	85,76	42,14

FASE DI TEST - Riepilogo



CONCLUSIONI

Pur non avendo alcuna nozione passata di metodi di apprendimento e reti neurali, è stato interessante essere riusciti ,con buona approssimazione, a determinare la predizione ed a vedere dunque un buon funzionamento.

Questa applicazione, seppure distante da uno scenario reale, ci ha consentito di conoscere ed approfondire i meccanismi alla base di tanti sistemi che ignoravamo.

Uno sviluppo futuro potrebbe essere *migliorare l'algoritmo di predizione*, eliminando il *valore H* che limita staticamente il percorso e sostituendolo con un *parametro dinamico* che potrebbe scaturire dall'*analisi della mobilità degli utenti effettuata tramite un processo di analisi di BigData*.

FINE