



Università degli Studi di Catania
Dipartimento di Ingegneria Elettrica Elettronica e Informatica
Corso di Laurea Magistrale in Ingegneria Informatica

CONFIGURABLE MULTI-LAYER PERCEPTRON FOR NEURON CRITICALITY ANALYSIS AND APPROXIMATE COMPUTING

Presentazione progetto finale

Studente:
Marco Pisasale (055000348)

Anno Accademico 2018/2019

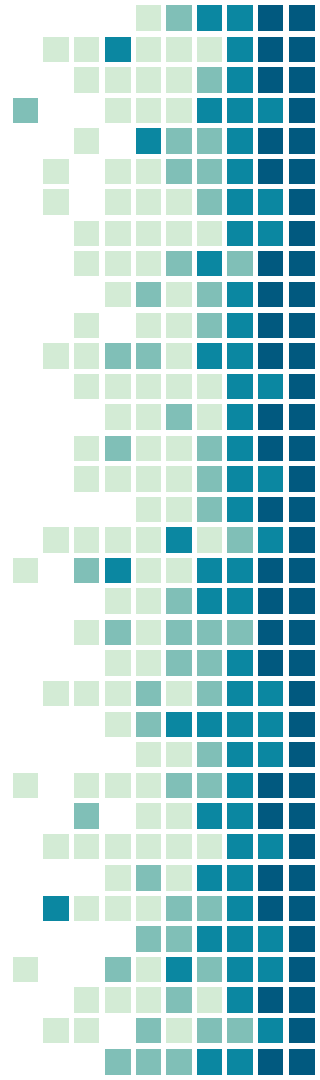
Introduzione

Chapter 0



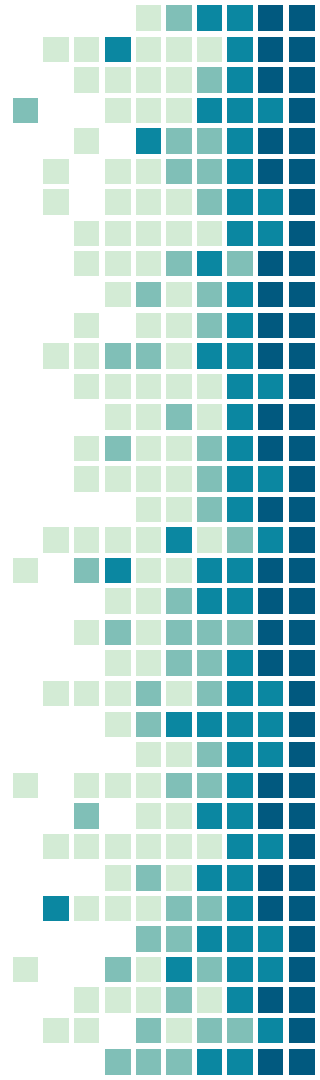
Obiettivi del progetto

- 1) Allenare una rete neurale nella sua configurazione originale (Aorig).
- 2) Determinare il **criticality factor** di ogni neurone.
- 3) Applicare una tecnica a scelta di **approximate computing** ai neuroni meno critici.
- 4) Valutare l'accuratezza della rete approssimata (Aapprox).
- 5) Allenare la rete approssimata (Aapprox2)
- 6) Confrontare Aorig, Aapprox, Aapprox2.




Repository GitHub




- Il progetto realizzato è disponibile al seguente link:
<https://github.com/marco-prg/IOT-Projectwork>
- La repository contiene tutto il codice sorgente prodotto, la documentazione completa, i checkpoint di tutti i modelli testati e screenshot e immagini, che evidenziano i progressi effettuati.




Repository GitHub



[Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)

 [marco-prg / IOT-Projectwork](#)

Unwatch 1

Star 0

Fork 0

<> Code

Issues 0

Pull requests 0

Projects 0

Wiki

Security

Insights

Settings

Project work of the Internet of Things based Smart Systems course at the University of Catania.

tinydnn [Manage topics](#)

3 commits

1 branch

0 releases

1 contributor

GPL-3.0

Branch: master


New pull request

Create new file








Upload files

Find File

Clone or download

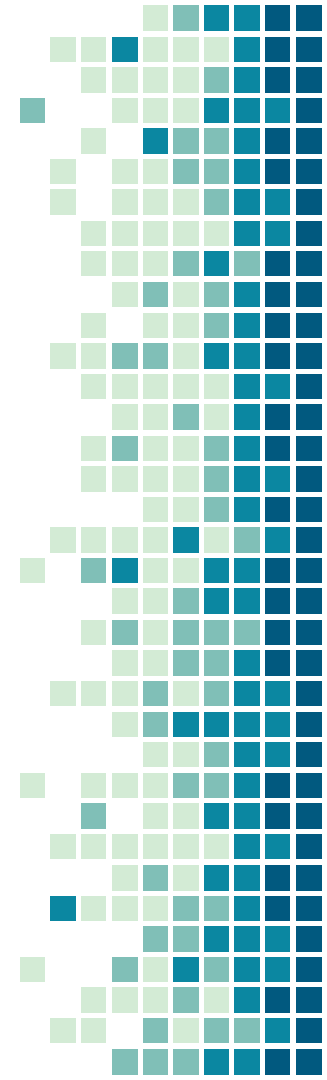
 marco-prg Updated README

Latest commit 34Fc085 8 minutes ago

 doc	Updated repository	19 minutes ago
 imgs	Updated repository	19 minutes ago
 models	Updated repository	19 minutes ago
 screenshots	Updated repository	19 minutes ago
 src	Updated repository	19 minutes ago
 LICENSE	Initial commit	2 hours ago
 README.md	Updated README	8 minutes ago

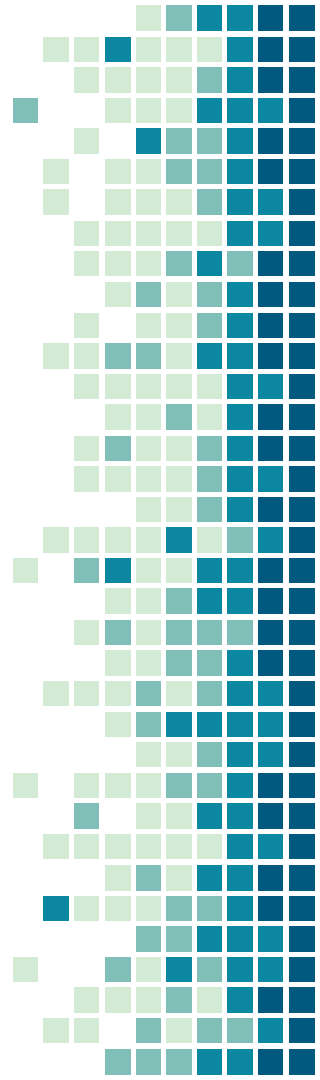
Contenuti repository GitHub

- **src:** codice sorgente prodotto.
- **src/data:** MNIST dataset e samples usati per NCA.
- **models:** save dei modelli di tutte le configurazioni testate.
- **doc:** documentazione del progetto.
- **screenshots:** screenshot del programma in esecuzione.
- **imgs:** immagini relative alla NCA, al modello e al dataset.

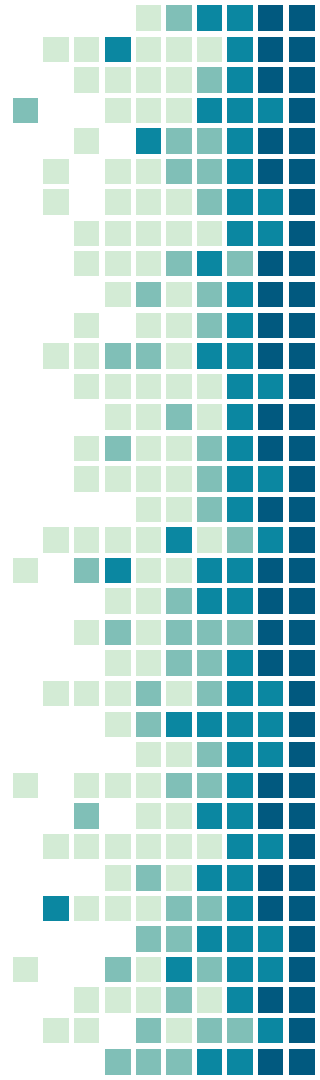


Ambito applicativo e dataset

- È stato affrontato un problema di **classificazione multiclasse**.
- Il dataset scelto è il **MNIST handwritten digit**, dataset di immagini contenente 60.000 training samples e 10.000 test samples, ampiamente utilizzato nell'ambito delle reti neurali.
- Le immagini rappresentano numeri scritti a mano, per un totale quindi di 10 classi (0-9), e hanno dimensione 32x32.

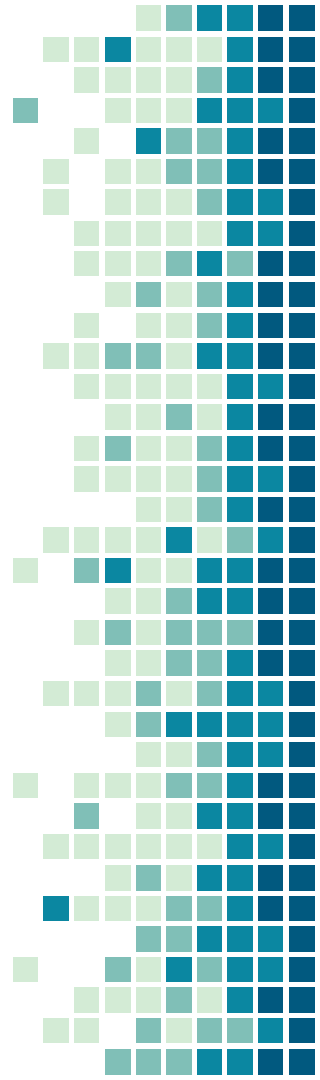


MNIST dataset: esempi



Rete neurale: architettura

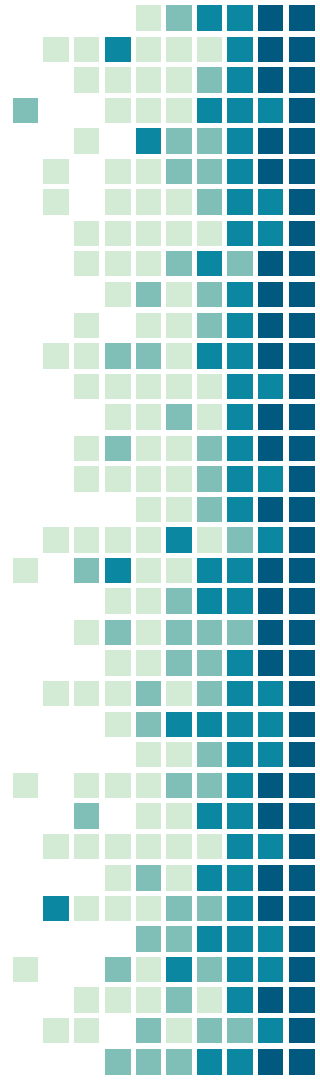
- La rete neurale utilizzata è **tinyDNN**.
- L'architettura della rete neurale utilizzata è di tipo **multi-layer perceptron**: solo fully connected layer (no convolutional layer).
- N. 3 fully connected layer (+ activation functions).
- Possibilità di specificare il numero di neuroni dei primi due layer (eccetto output layer).



Rete neurale: architettura

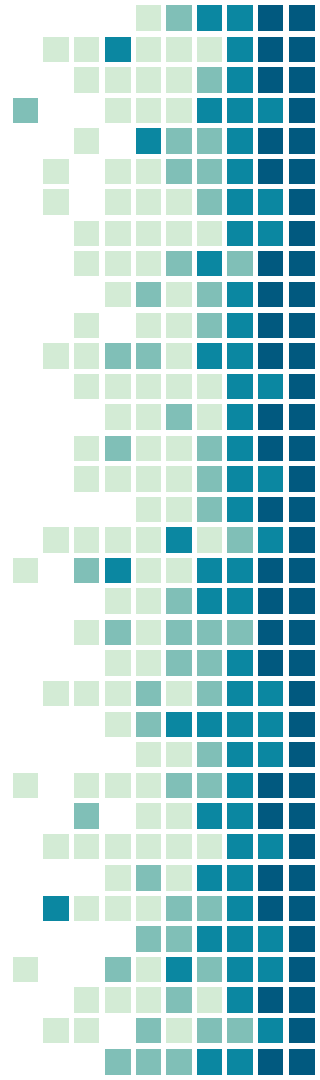
- Dato l'utilizzo esclusivo di layer fc, per ridurre il numero di pesi del primo hidden layer l'input size viene ridotta da 32x32 a 16x16 con un layer di average pooling, senza conseguenze per l'accuracy della rete.

```
nn <- ave_pool(32, 32, 1, 2) // S1, 1@32x32-in, 1@16x16-out
  <- fc(256, a, true, backend_type) // F2, 256-in, a-out
  <- tanh()
  <- fc(a, b, true, backend_type) // F3, a-in, b-out
  <- tanh()
  <- fc(b, 10, true, backend_type) // F4, b-in, 10-out
  <- tanh();
```



Rete neurale: architettura

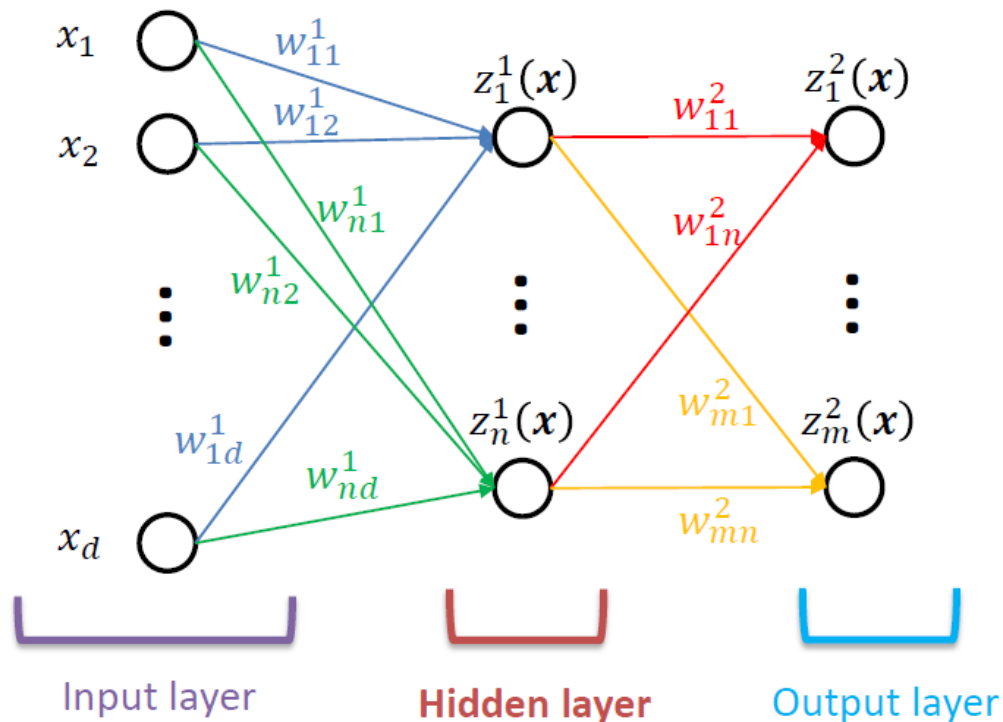
- L'input image ridotta (16x16) viene vettorizzata in un vettore di 256 elementi, che costituisce l'input layer.
- Ogni neurone del primo hidden layer avrà un peso per ciascuno degli elementi dell'input layer, per un totale di 256 pesi.
- L'**output layer**, coerentemente con il problema di classificazione e il dataset scelto, possiederà 10 neuroni, ognuno dei quali rappresenterà l'attivazione di una classe (0-9).
- I neuroni dell'output layer daranno uno score per ogni classe e la classe predetta sarà determinata dallo score maggiore.



Multi-layer Perceptron

- Input layer: size = 256.
- 2 hidden layer (numero di neuroni configurabile).
- Output layer: 10 neuroni (1 per classe, 0-9).
- Multiclass classification:

$$y = \operatorname{argmax}_c \mathbf{w}_c^T \mathbf{v} + b_c$$



Funzionalità sviluppate

```
*** Configurable Multilayer Perceptron for Neuron Criticality Analysis and Approximate Computing ***
```

```
Please digit one of the following command number:
```

- 1) Network construction and training using original configuration
- 2) Single file input test and Neuron Criticality Analysis (NCA)
- 3) Network weights approximation (based on last NCA ranking)
- 4) Complete test set evaluation
- 5) Training existent model with weights approximation (based on last NCA ranking)
- 6) Training new model from scratch with weights approximation (based on last NCA ranking)
- 7) Full comparison (Aorig, Aapprox, Aapprox2, Aapprox3)
- 8) Print network weights (most recent configuration)
- 9) Exit

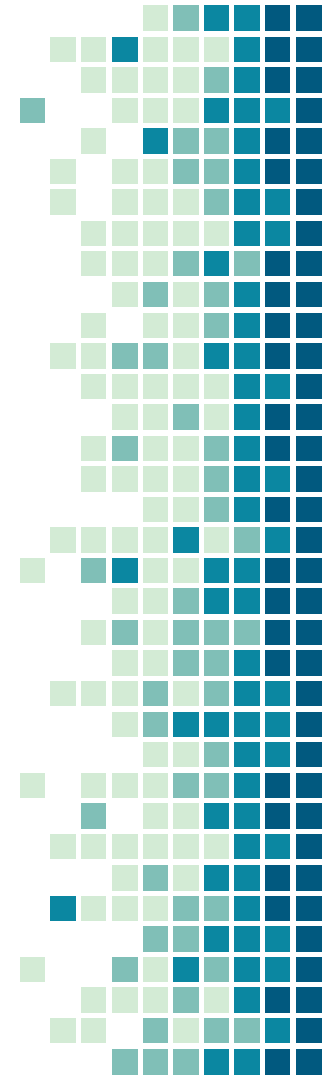
Aorig training

Chapter 1



Configurazione utilizzata

- Per il training della rete in tutte le configurazioni valutate nelle varie fasi del progetto sono stati utilizzati gli stessi valori per gli **hyperparameters**, di seguito riportati:
- Learning rate = 0.1
- Minibatch size = 64
- Number of epochs = 5
- Per quanto riguarda l'**architettura**, il numero di neuroni degli hidden layers (configurabile) è stato impostato a 10 per entrambi, per un totale di 30 neuroni (10-10-10).



Risultati ottenuti

- Le migliori performance si sono registrate all'epoca 5, con una test accuracy pari al 83,77%.

accuracy:83.77% (8377/10000)

*	0	1	2	3	4	5	6	7	8	9
0	929	0	14	6	0	51	18	4	6	18
1	0	1105	48	12	24	25	8	51	38	14
2	2	5	821	16	7	8	3	24	9	5
3	7	4	38	914	4	121	2	12	48	21
4	1	1	12	2	819	48	11	13	9	48
5	3	0	2	5	0	357	4	0	12	1
6	24	5	25	6	19	48	910	4	18	4
7	2	0	24	20	1	33	1	897	11	56
8	12	15	40	22	23	172	1	1	797	14
9	0	0	8	7	85	29	0	22	26	828

Confusion matrix



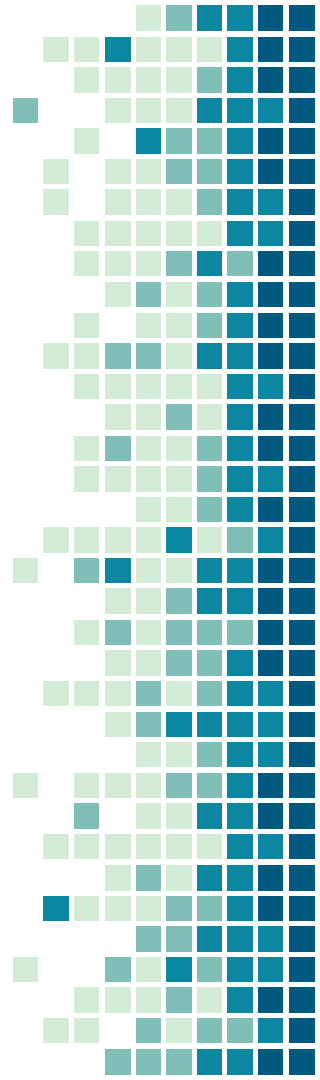
Neuron criticality analysis

Chapter 2



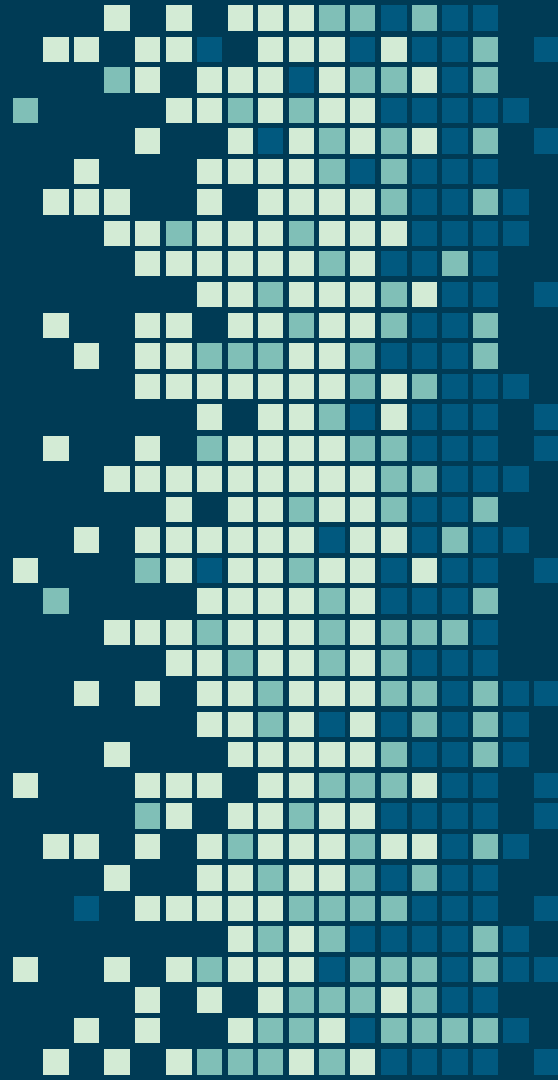
Neuron criticality analysis

- La NCA è una tecnica che permette di valutare il cosiddetto **criticality factor** per ogni neurone della rete neurale, sia per quelli dell'output layer che per quelli degli hidden layers.
- Il criticality factor permette di capire quali neuroni impattano maggiormente sull'uscita della rete.
- L'idea è quella di valutare il criticality factor di ogni neurone, determinare così un **ranking** e successivamente applicare una tecnica di **approximate computing** ai neuroni meno critici.
- Approssimare i neuroni meno critici comporta sicuramente una **quality degradation** minore.

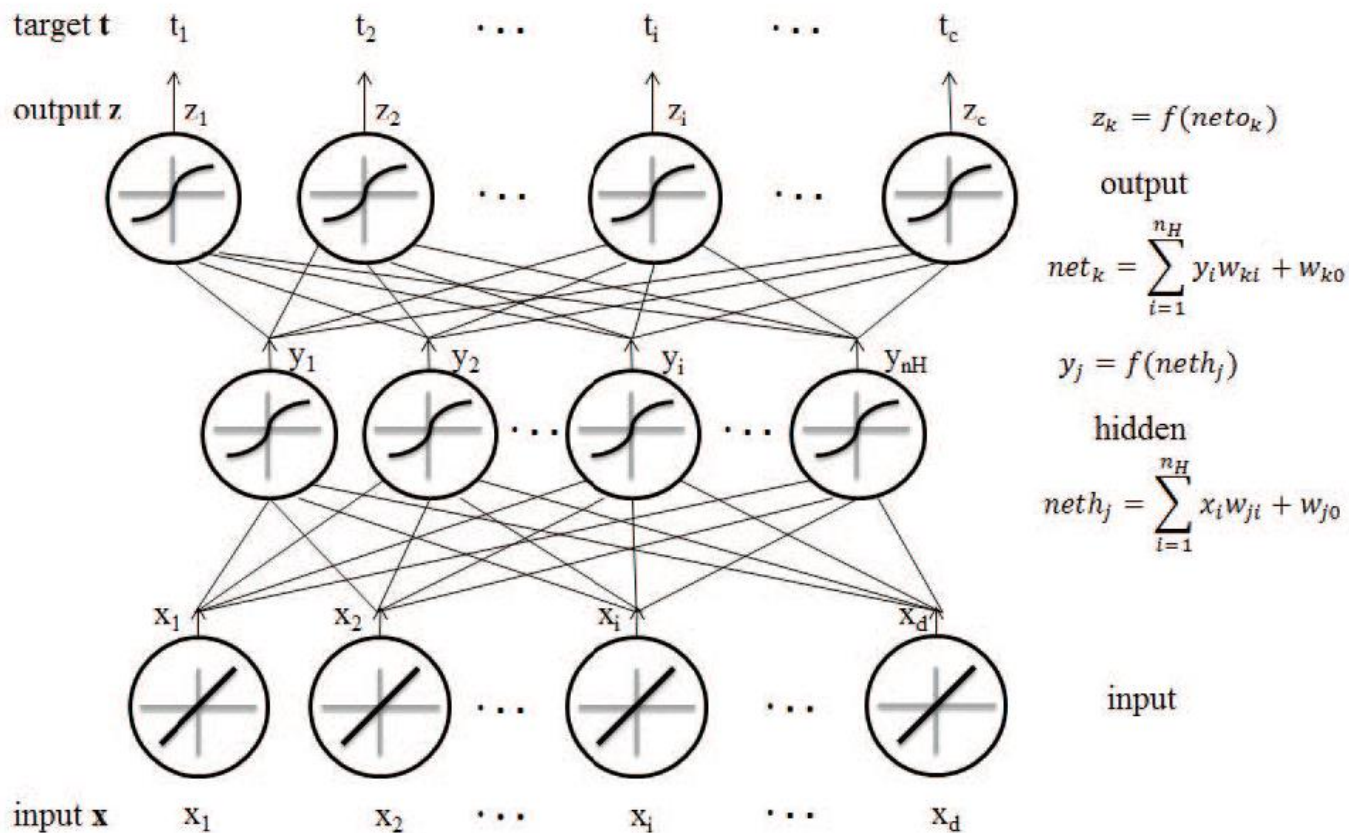


“ [...] we say a neuron is less “critical” if its accuracy requirements relaxation leads to less final quality degradation. A less “critical” neuron will have a higher priority to be approximated. Based on this analysis, we sort all the output and hidden neurons in ascending order, and finally get the criticality ranking vector $s = \{s_1, s_2, \dots, s_n\}$ for a given network.

*~ Neuron criticality analysis,
ApproxANN paper*



Configurazione iniziale



Formulazione analitica

- La **cost function** finale della rete è descritta come:

$$E = \frac{1}{2} \sum_{k=1}^c (t_k - z_k)^2 = \frac{1}{2} \|t - z\|^2$$

- La **neuron criticality** (nc_i) sarà rappresentata dalla derivata di E rispetto all'output del neurone prima dell'applicazione dell'activation function (net_i):

$$nc_i = \frac{\partial E}{\partial net_i}.$$

Ricordiamo:

$$z_k = f(net_o_k)$$

output

$$net_k = \sum_{i=1}^{n_H} y_i w_{ki} + w_{k0}$$

$$y_j = f(neth_j)$$

hidden

$$neth_j = \sum_{i=1}^{n_H} x_i w_{ji} + w_{j0}$$

Calcolo criticality factor

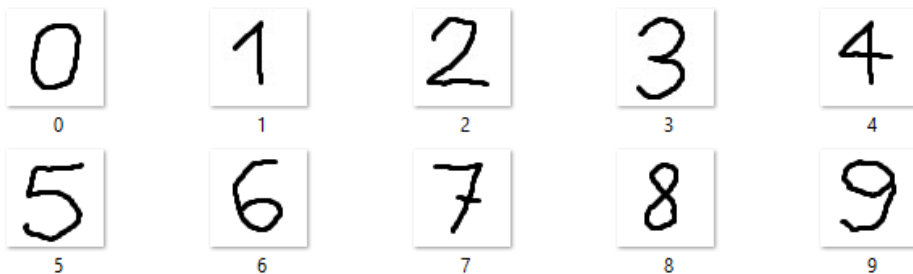
- Per i neuroni dell'output layer: $nco_k = \frac{\partial E}{\partial neto_k}$
$$= \frac{\partial \frac{1}{2} \sum_{k=1}^c (t_k - z_k)^2}{\partial z_k} \cdot \frac{\partial z_k}{\partial neto_k}$$
$$= -(t_k - z_k) \cdot f'(neto_k)$$

- Per i neuroni degli hidden layer: $nch_j = \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial neth_j}$
$$= -f'(neth_j) \cdot \sum_{k=1}^c nco_k w_{kj}$$

$$\begin{aligned} \frac{\partial E}{\partial y_j} &= \frac{\partial}{\partial y_j} \left[\frac{1}{2} \sum_{k=1}^c (t_k - z_k)^2 \right] \\ &= - \sum_{k=1}^c (t_k - z_k) \cdot \frac{\partial z_k}{\partial neto_k} \cdot \frac{\partial neto_k}{\partial y_j} \\ &= - \sum_{k=1}^c nco_k w_{kj} \end{aligned}$$

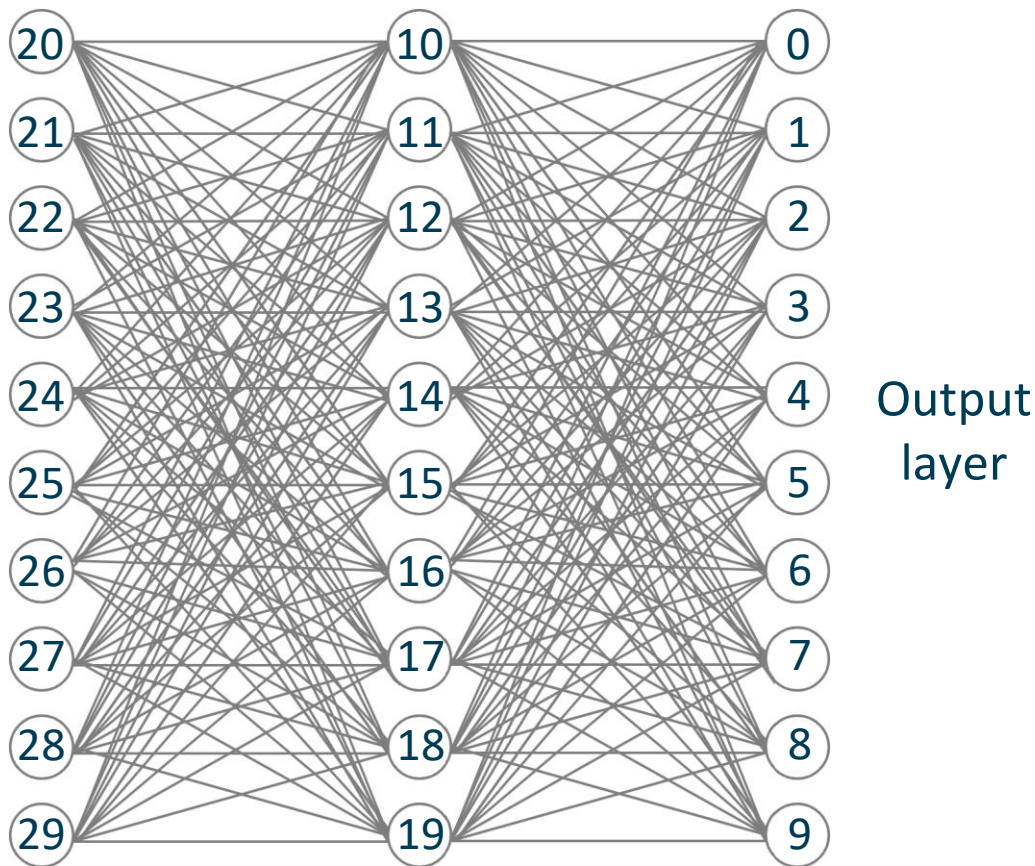
Neuron criticality analysis

- Si è scelto di valutare la criticality dei neuroni, applicando le formule precedentemente viste, basandosi su **singolo input**.
- In questo modo è stato possibile evidenziare le variazioni sul ranking dei neuroni al variare della target class.
- A tale scopo, sono stati generati 10 samples appartenenti alle 10 classi (0-9) del dataset, che verranno innanzitutto **testati**, per poi eseguire una NCA su ciascuno di essi:



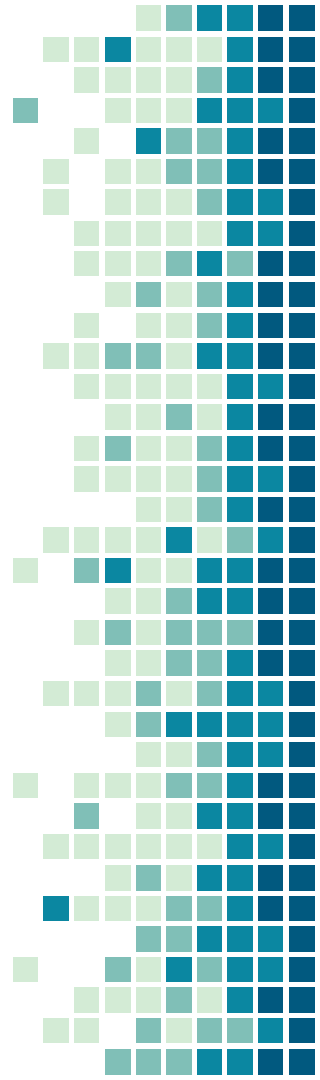
Configurazione: numerazione dei neuroni

Input (32x32) →
avg_pool (2x2)
→ (16x16)
vettorizzato →
Input layer a
256 entries



NCA: numerazione neuroni e ranking

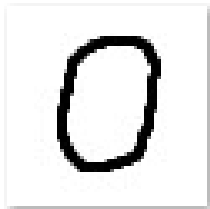
- In questo modo il numero di ciascun neurone dell'output layer corrisponde direttamente alla target class che rappresenta ai fini della classificazione, facilitando ulteriormente la comprensione dei risultati ottenuti.
- Il ranking relativo alla NCA viene presentato con il numero del neurone così assegnato e il corrispondente valore del criticality factor.
- Il ranking è ordinato in **ordine crescente di criticità**.
- Viene inoltre effettuato un rescaling (0-100) dei risultati, per facilitarne la comprensione.



NCA: target class 0

```
Neuron 0: -0.715782
Neuron 17: -0.265993
Neuron 14: -0.151234
Neuron 16: -0.148171
Neuron 10: -0.138503
Neuron 12: -0.132075
Neuron 9: -0.103511
Neuron 1: -0.0896535
Neuron 25: -0.0666972
Neuron 8: -0.0378488
Neuron 18: -0.026341
Neuron 28: -0.018794
Neuron 13: 0.0138321
Neuron 4: 0.033758
Neuron 24: 0.0472664
Neuron 29: 0.0703422
Neuron 23: 0.0922675
Neuron 26: 0.0926065
Neuron 5: 0.105781
Neuron 27: 0.10783
Neuron 2: 0.137302
Neuron 15: 0.16602
Neuron 22: 0.193723
Neuron 7: 0.217835
Neuron 6: 0.251545
Neuron 20: 0.284116
Neuron 3: 0.298748
Neuron 19: 0.365249
Neuron 11: 0.487036
Neuron 21: 0.492211
```

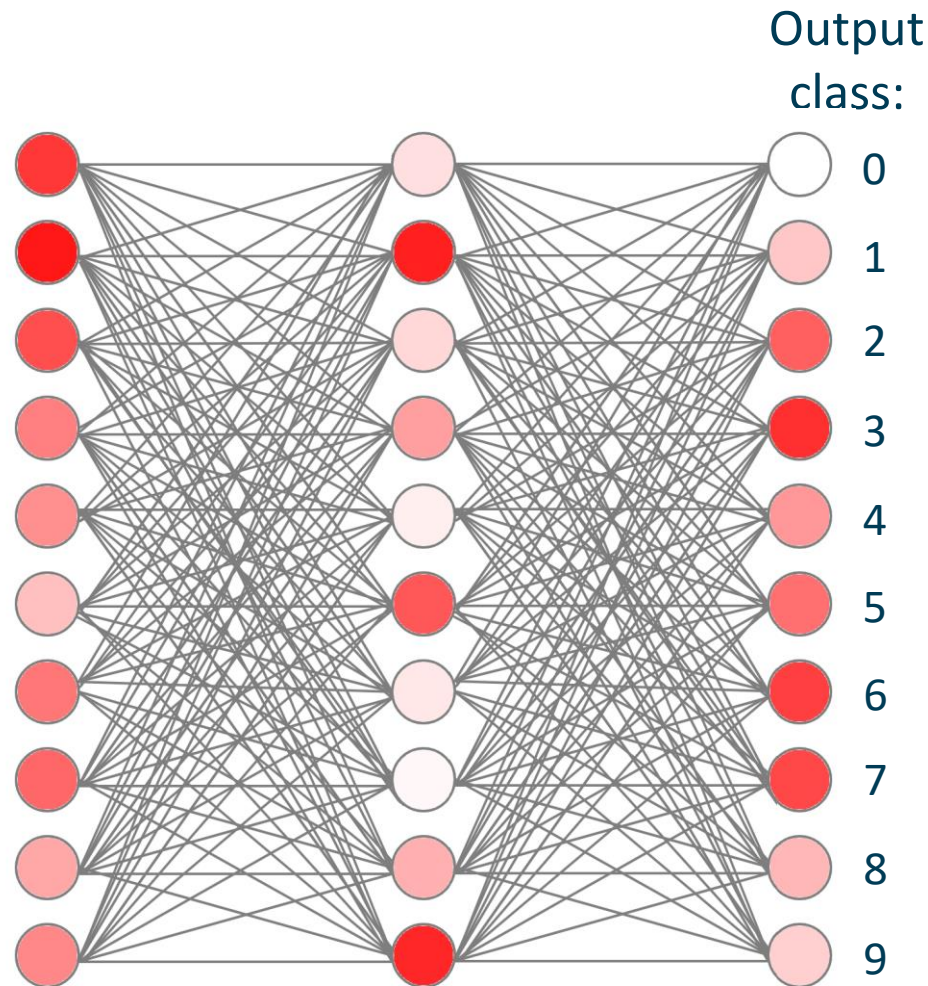
Test sample:



0

```
*** Prediction results ***
3: 71.0645
6: 66.9734
0: 65.0211
7: 64.3751
2: 58.753
5: 56.6879
4: 52.1123
8: 47.631
1: 44.3505
9: 43.4589
```

Predizione errata



NCA: target class 1

```
Neuron 1: -0.377386
Neuron 12: -0.247271
Neuron 14: -0.1714
Neuron 4: -0.15464
Neuron 10: -0.0923554
Neuron 0: -0.0771706
Neuron 17: -0.0771229
Neuron 23: -0.0546145
Neuron 25: -0.039508
Neuron 28: -0.0364336
Neuron 20: -0.0208223
Neuron 5: -0.00380213
Neuron 8: 0.0025645
Neuron 16: 0.0181323
Neuron 13: 0.0433729
Neuron 19: 0.0473312
Neuron 6: 0.0725197
Neuron 9: 0.0903343
Neuron 29: 0.0923066
Neuron 11: 0.0946805
Neuron 24: 0.100113
Neuron 21: 0.113558
Neuron 27: 0.122407
Neuron 15: 0.131807
Neuron 2: 0.137951
Neuron 3: 0.138916
Neuron 26: 0.195659
Neuron 7: 0.200929
Neuron 18: 0.207289
Neuron 22: 0.217665
```

Test sample:

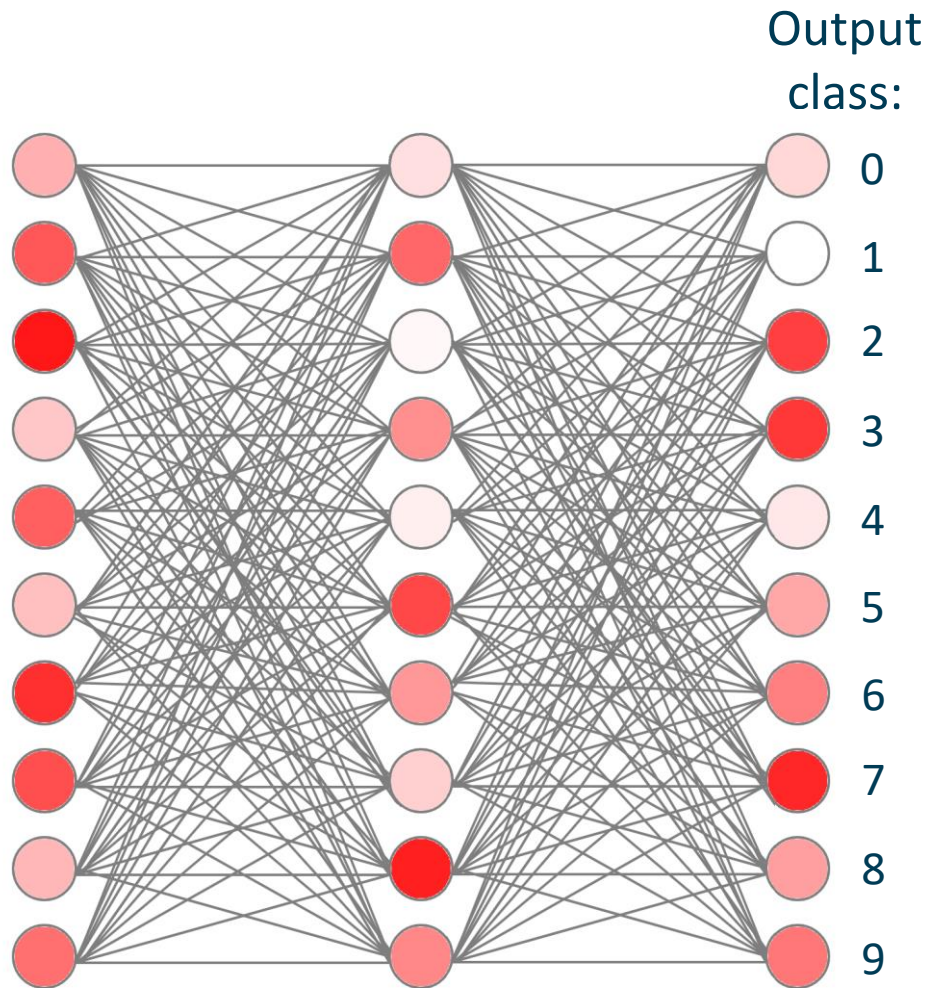


1

*** Prediction results ***

```
1: 81.1308
7: 63.1387
3: 58.8603
2: 58.7961
9: 55.6931
6: 54.5567
8: 50.1603
5: 49.7624
0: 45.1476
4: 40.0855
```

Predizione corretta



NCA: target class 2

```
Neuron 2: -0.719116
Neuron 11: -0.485235
Neuron 10: -0.251519
Neuron 27: -0.242619
Neuron 14: -0.228863
Neuron 22: -0.212346
Neuron 21: -0.211507
Neuron 4: -0.196748
Neuron 8: -0.183155
Neuron 7: -0.120645
Neuron 5: -0.117648
Neuron 18: -0.0933397
Neuron 12: -0.0156006
Neuron 0: -0.0129109
Neuron 17: 0.0115651
Neuron 23: 0.0347612
Neuron 20: 0.0409065
Neuron 26: 0.0498443
Neuron 15: 0.0613136
Neuron 28: 0.0626261
Neuron 9: 0.0727647
Neuron 13: 0.118424
Neuron 25: 0.134721
Neuron 29: 0.255968
Neuron 6: 0.28699
Neuron 1: 0.290501
Neuron 3: 0.347358
Neuron 19: 0.388119
Neuron 16: 0.399127
Neuron 24: 0.523087
```

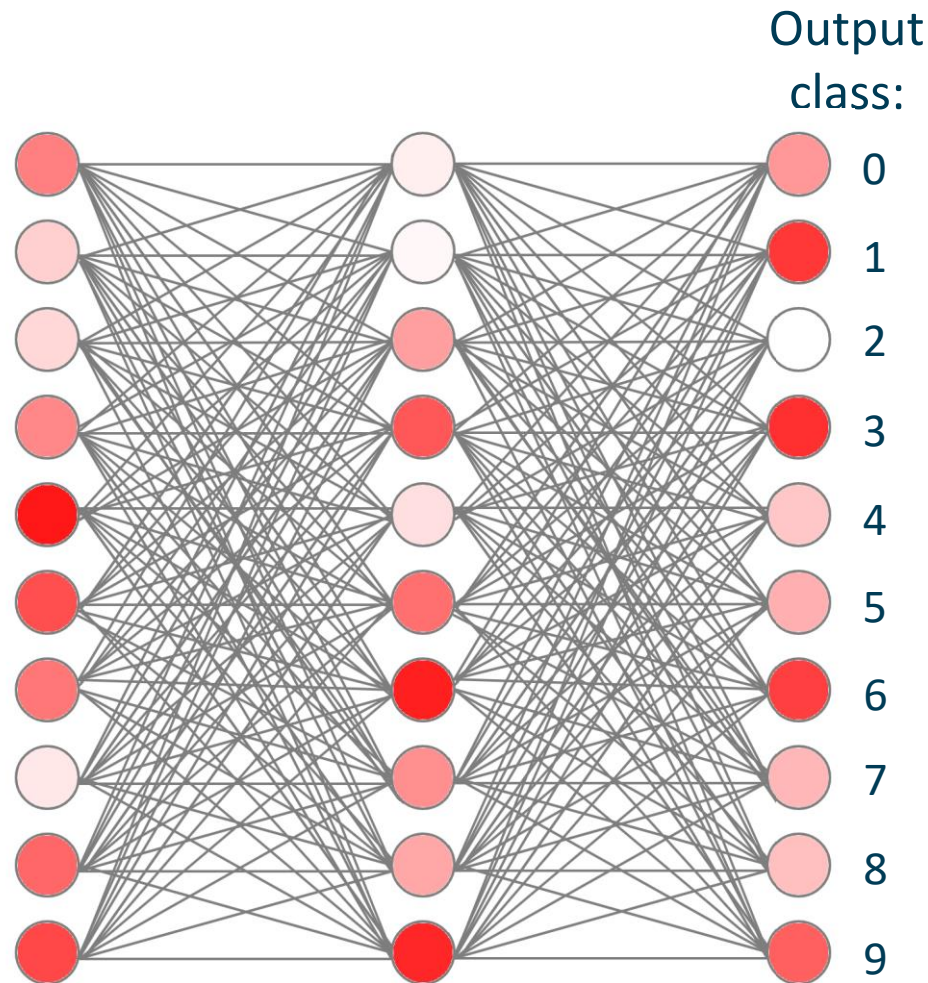
Test sample:



2

```
*** Prediction results ***
3: 76.4436
1: 70.2969
6: 69.9782
2: 64.8617
9: 54.5723
0: 49.1929
5: 42.5407
7: 42.3449
8: 38.124
4: 37.1615
```

Predizione errata



NCA: target class 3

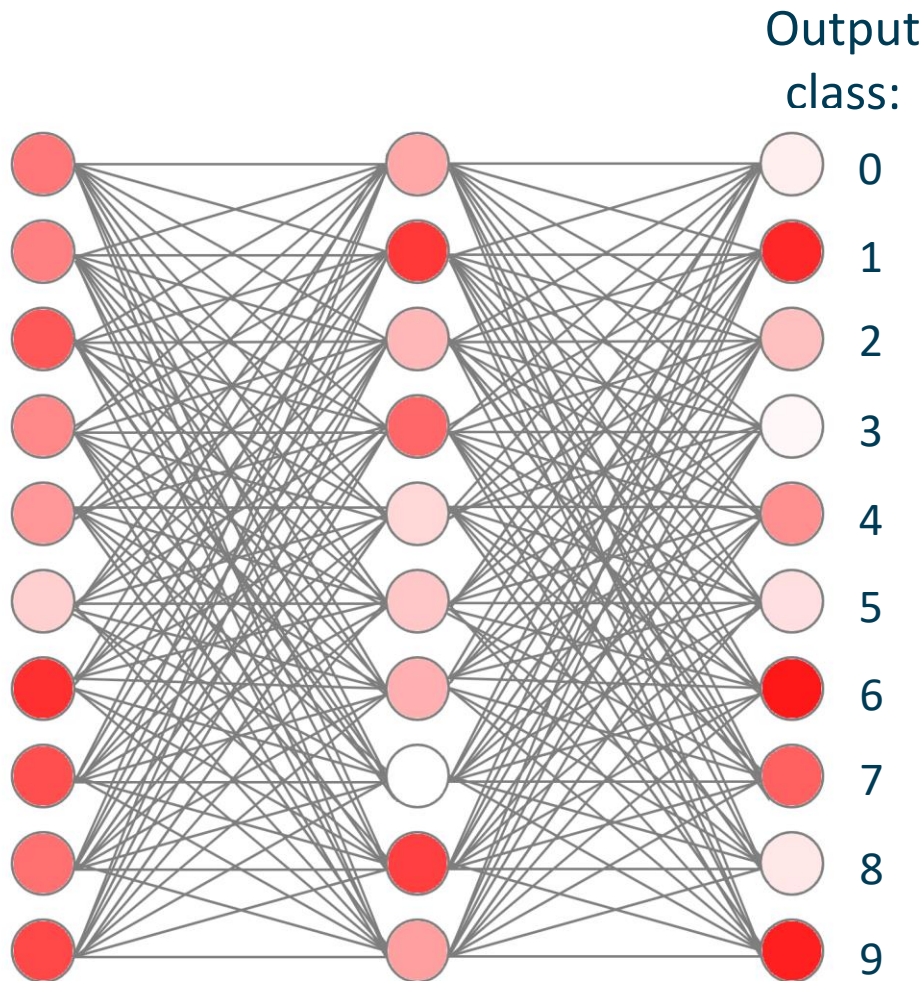
```
Neuron 17: -0.370634
Neuron 3: -0.359704
Neuron 0: -0.323274
Neuron 8: -0.251979
Neuron 5: -0.140262
Neuron 14: -0.133985
Neuron 25: -0.133737
Neuron 15: -0.12005
Neuron 2: -0.084331
Neuron 12: -0.0646769
Neuron 16: -0.0253029
Neuron 10: -0.00341158
Neuron 19: 0.000645789
Neuron 24: 0.00554787
Neuron 4: 0.0120169
Neuron 23: 0.0267845
Neuron 21: 0.0315101
Neuron 20: 0.0323004
Neuron 28: 0.0377106
Neuron 13: 0.0466207
Neuron 7: 0.0487398
Neuron 22: 0.0912011
Neuron 27: 0.0922821
Neuron 29: 0.10813
Neuron 18: 0.143906
Neuron 11: 0.151854
Neuron 26: 0.158589
Neuron 1: 0.218497
Neuron 9: 0.274672
Neuron 6: 0.361587
```

Test sample:



```
*** Prediction results ***
3: 82.0187
6: 78.5677
9: 68.8935
1: 64.4244
7: 53.0535
4: 50.7512
2: 44.691
5: 41.0501
8: 32.9918
0: 26.4532
```

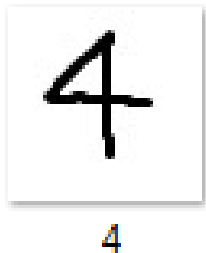
Predizione corretta



NCA: target class 4

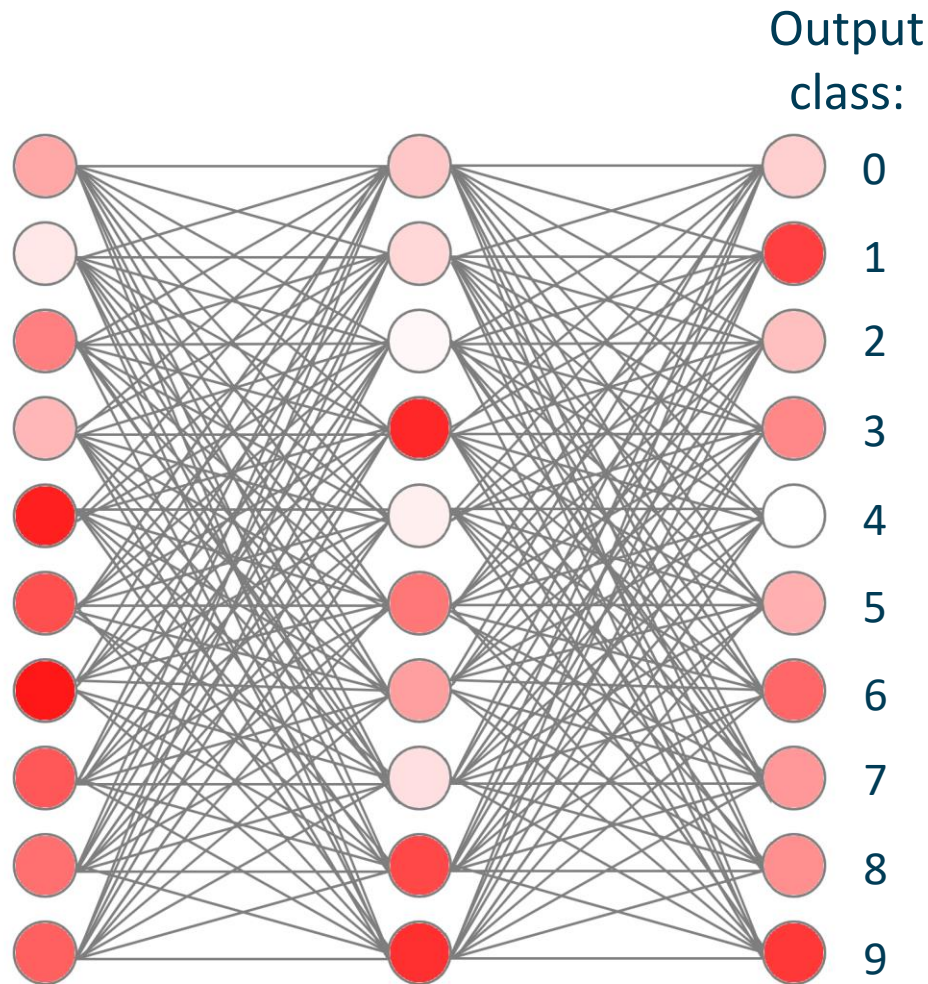
```
Neuron 4: -1.0289
Neuron 12: -0.739415
Neuron 14: -0.371611
Neuron 21: -0.317247
Neuron 17: -0.276744
Neuron 11: -0.270196
Neuron 0: -0.219338
Neuron 10: -0.175186
Neuron 2: -0.148427
Neuron 23: -0.096718
Neuron 5: 0.00594006
Neuron 20: 0.0105489
Neuron 16: 0.0293029
Neuron 7: 0.0422109
Neuron 8: 0.0614887
Neuron 3: 0.0865199
Neuron 22: 0.0926296
Neuron 15: 0.142041
Neuron 28: 0.145384
Neuron 6: 0.155659
Neuron 29: 0.171384
Neuron 27: 0.202519
Neuron 25: 0.208293
Neuron 18: 0.210346
Neuron 1: 0.310053
Neuron 9: 0.330291
Neuron 19: 0.442145
Neuron 13: 0.5458
Neuron 24: 0.736609
Neuron 26: 0.746254
```

Test sample:



```
*** Prediction results ***
9: 74.3302
1: 72.1666
6: 59.9834
3: 55.4489
8: 53.8577
7: 52.6429
5: 50.3713
4: 48.1366
2: 40.5041
0: 35.5131
```

Predizione errata



NCA: target class 5

```
Neuron 5: -1.04387
Neuron 10: -0.916032
Neuron 0: -0.272348
Neuron 28: -0.179179
Neuron 17: -0.173063
Neuron 8: -0.150815
Neuron 14: -0.101801
Neuron 22: -0.0522069
Neuron 27: -0.03033
Neuron 4: -0.00587485
Neuron 19: 0.0372027
Neuron 20: 0.0381081
Neuron 2: 0.0718714
Neuron 26: 0.0990467
Neuron 11: 0.124783
Neuron 15: 0.136197
Neuron 7: 0.150738
Neuron 16: 0.20611
Neuron 12: 0.210436
Neuron 25: 0.213273
Neuron 1: 0.222977
Neuron 3: 0.24752
Neuron 9: 0.261917
Neuron 29: 0.269208
Neuron 13: 0.31057
Neuron 6: 0.330572
Neuron 24: 0.350833
Neuron 18: 0.391213
Neuron 21: 0.464198
Neuron 23: 0.477743
```

Test sample:

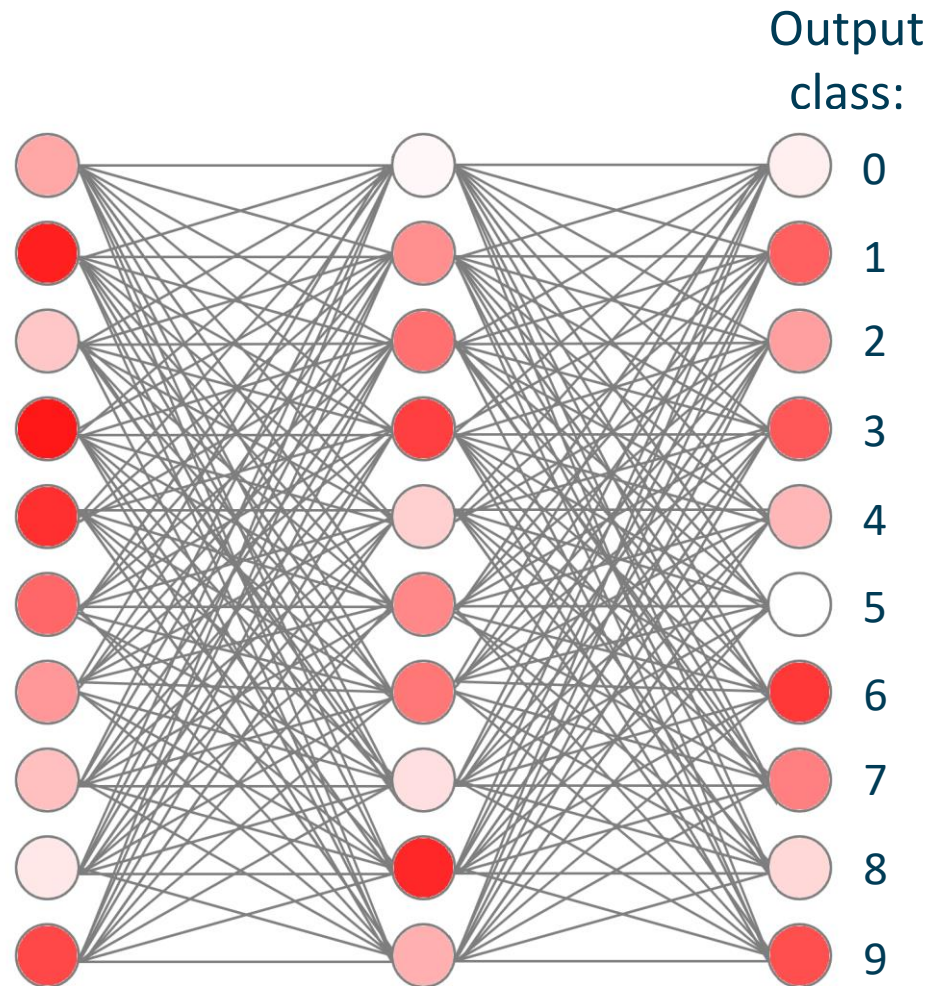


5

*** Prediction results ***

```
6: 74.3624
9: 67.818
3: 66.6521
1: 64.7591
7: 59.6513
2: 54.5155
4: 49.6328
5: 47.1191
8: 40.3436
0: 31.3057
```

Predizione errata



NCA: target class 6

```
Neuron 6: -0.463926
Neuron 29: -0.248391
Neuron 26: -0.237213
Neuron 10: -0.232065
Neuron 13: -0.205847
Neuron 18: -0.184362
Neuron 28: -0.176783
Neuron 24: -0.168313
Neuron 27: -0.144961
Neuron 0: -0.106042
Neuron 1: -0.0933814
Neuron 7: -0.0401178
Neuron 19: -0.038437
Neuron 22: -0.0313707
Neuron 8: -0.0138577
Neuron 9: -0.00885506
Neuron 12: 0.0048085
Neuron 16: 0.0487032
Neuron 5: 0.0558564
Neuron 15: 0.0778625
Neuron 25: 0.126795
Neuron 20: 0.134343
Neuron 17: 0.143324
Neuron 4: 0.197885
Neuron 23: 0.215961
Neuron 14: 0.267354
Neuron 2: 0.278346
Neuron 3: 0.30137
Neuron 11: 0.451423
Neuron 21: 0.461517
```

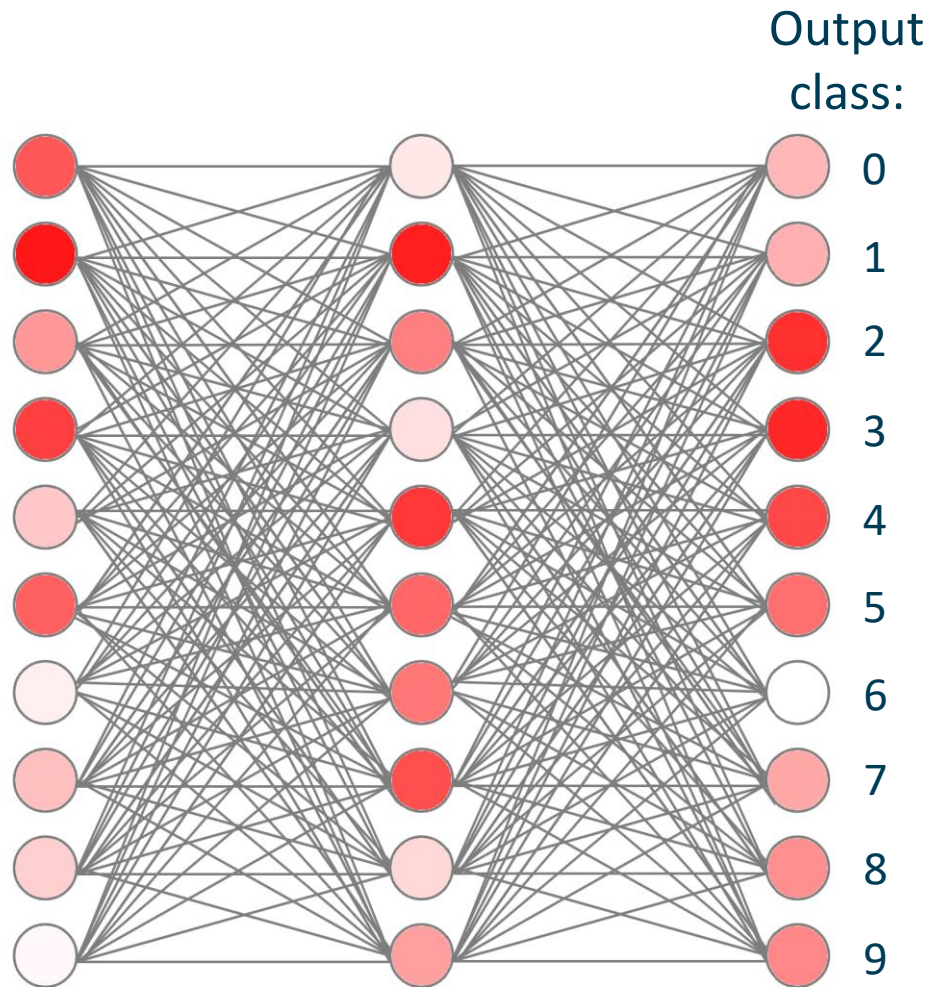
Test sample:



6

```
*** Prediction results ***
6: 76.9076
3: 71.3146
2: 69.2119
4: 62.9199
5: 53.502
9: 49.4465
8: 49.1337
7: 47.4886
1: 44.1114
0: 43.2952
```

Predizione corretta



NCA: target class 7

```
Neuron 7: -0.639839
Neuron 26: -0.603138
Neuron 18: -0.339952
Neuron 0: -0.325499
Neuron 5: -0.200597
Neuron 8: -0.186955
Neuron 10: -0.183073
Neuron 24: -0.0957883
Neuron 14: -0.0875681
Neuron 19: -0.0740496
Neuron 22: -0.060526
Neuron 28: 0.00667742
Neuron 27: 0.0244968
Neuron 2: 0.0617503
Neuron 4: 0.075596
Neuron 17: 0.0831392
Neuron 13: 0.0960611
Neuron 20: 0.121952
Neuron 29: 0.134041
Neuron 9: 0.150236
Neuron 23: 0.173555
Neuron 15: 0.185581
Neuron 1: 0.215042
Neuron 11: 0.236355
Neuron 25: 0.276231
Neuron 3: 0.279367
Neuron 16: 0.291091
Neuron 12: 0.292389
Neuron 21: 0.315293
Neuron 6: 0.371193
```

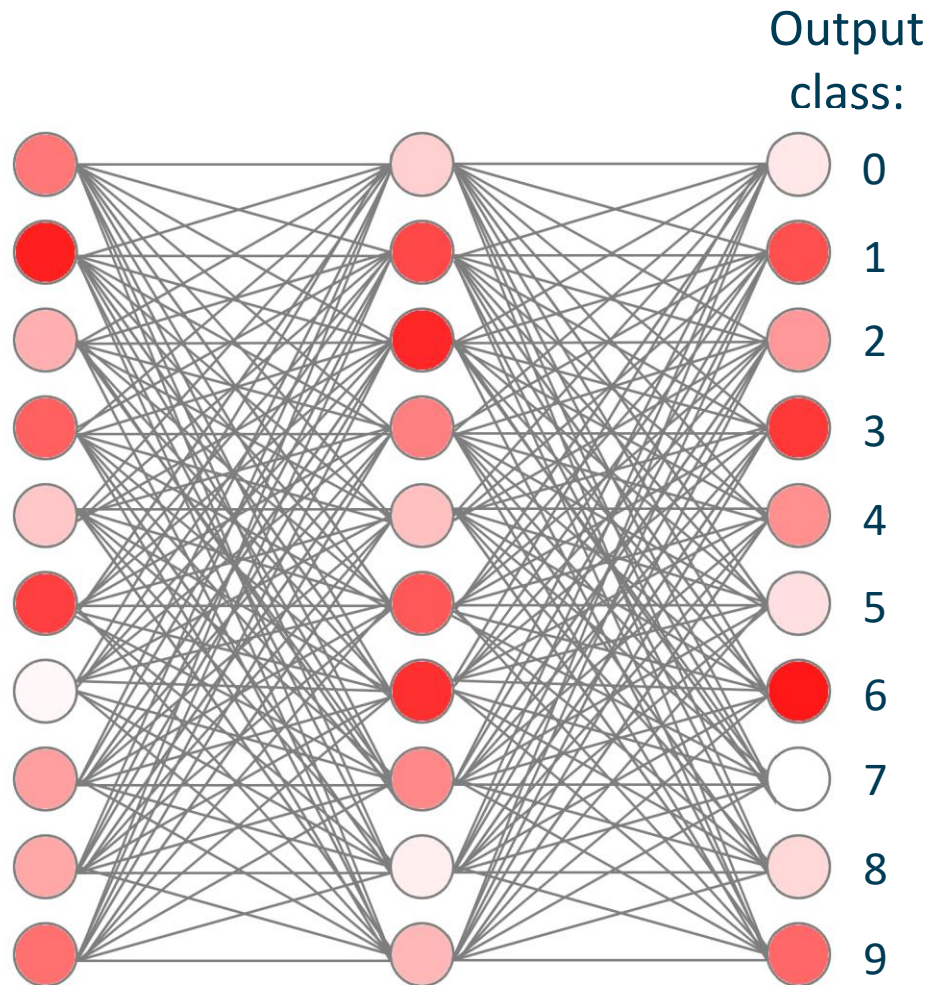
Test sample:



7

```
*** Prediction results ***
6: 80.3716
3: 69.3011
7: 68.6166
1: 64.1682
9: 59.6175
4: 54.7522
2: 53.8743
8: 37.8569
5: 36.8852
0: 26.209
```

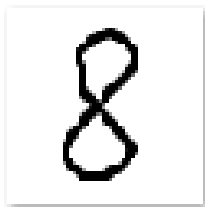
Predizione errata



NCA: target class 8

```
Neuron 8: -0.860048
Neuron 14: -0.830862
Neuron 12: -0.459069
Neuron 17: -0.372192
Neuron 18: -0.227058
Neuron 15: -0.181221
Neuron 0: -0.166294
Neuron 4: -0.123614
Neuron 26: -0.115593
Neuron 10: -0.0679804
Neuron 25: -0.0437769
Neuron 28: -0.0295811
Neuron 24: -0.0131392
Neuron 23: 0.0311175
Neuron 1: 0.0420627
Neuron 9: 0.0949439
Neuron 13: 0.10141
Neuron 5: 0.115136
Neuron 7: 0.145845
Neuron 3: 0.20264
Neuron 2: 0.209889
Neuron 27: 0.24399
Neuron 19: 0.283045
Neuron 21: 0.30695
Neuron 6: 0.331545
Neuron 22: 0.350614
Neuron 16: 0.350653
Neuron 29: 0.472474
Neuron 20: 0.551027
Neuron 11: 0.65718
```

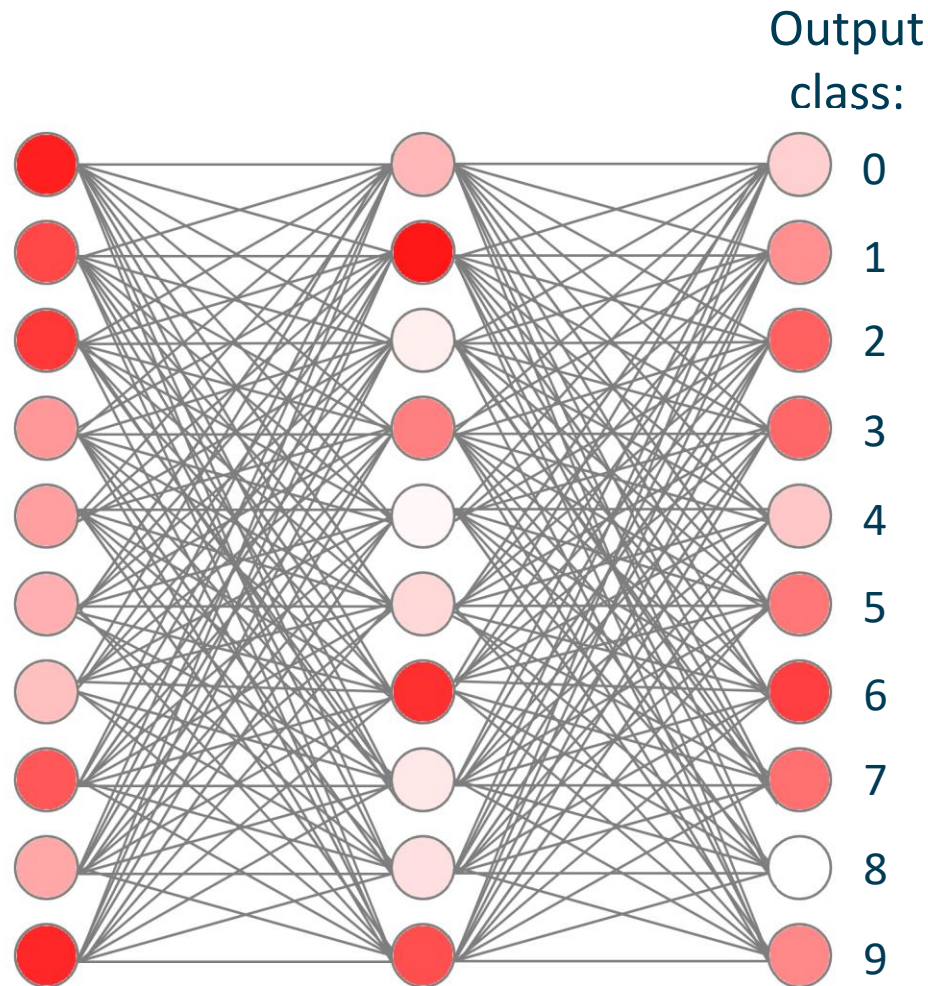
Test sample:



8

```
*** Prediction results ***
6: 74.4746
2: 63.7893
3: 63.2621
7: 59.3227
8: 57.879
5: 57.2954
9: 55.989
1: 52.6336
4: 42.1503
0: 39.2923
```

Predizione errata



NCA: target class 9

```
Neuron 9: -0.665698
Neuron 17: -0.445413
Neuron 4: -0.321508
Neuron 0: -0.294409
Neuron 16: -0.221389
Neuron 14: -0.185415
Neuron 11: -0.180112
Neuron 12: -0.131956
Neuron 10: -0.126044
Neuron 5: -0.123152
Neuron 21: -0.1158
Neuron 18: -0.0317543
Neuron 22: -0.0225046
Neuron 2: -0.0117437
Neuron 27: -0.0065303
Neuron 15: 0.0270281
Neuron 23: 0.0601304
Neuron 8: 0.0664822
Neuron 25: 0.074375
Neuron 20: 0.130977
Neuron 1: 0.134015
Neuron 6: 0.208054
Neuron 29: 0.242736
Neuron 7: 0.301972
Neuron 28: 0.327426
Neuron 3: 0.354756
Neuron 19: 0.430065
Neuron 26: 0.448335
Neuron 24: 0.457291
Neuron 13: 0.484834
```

Test sample:

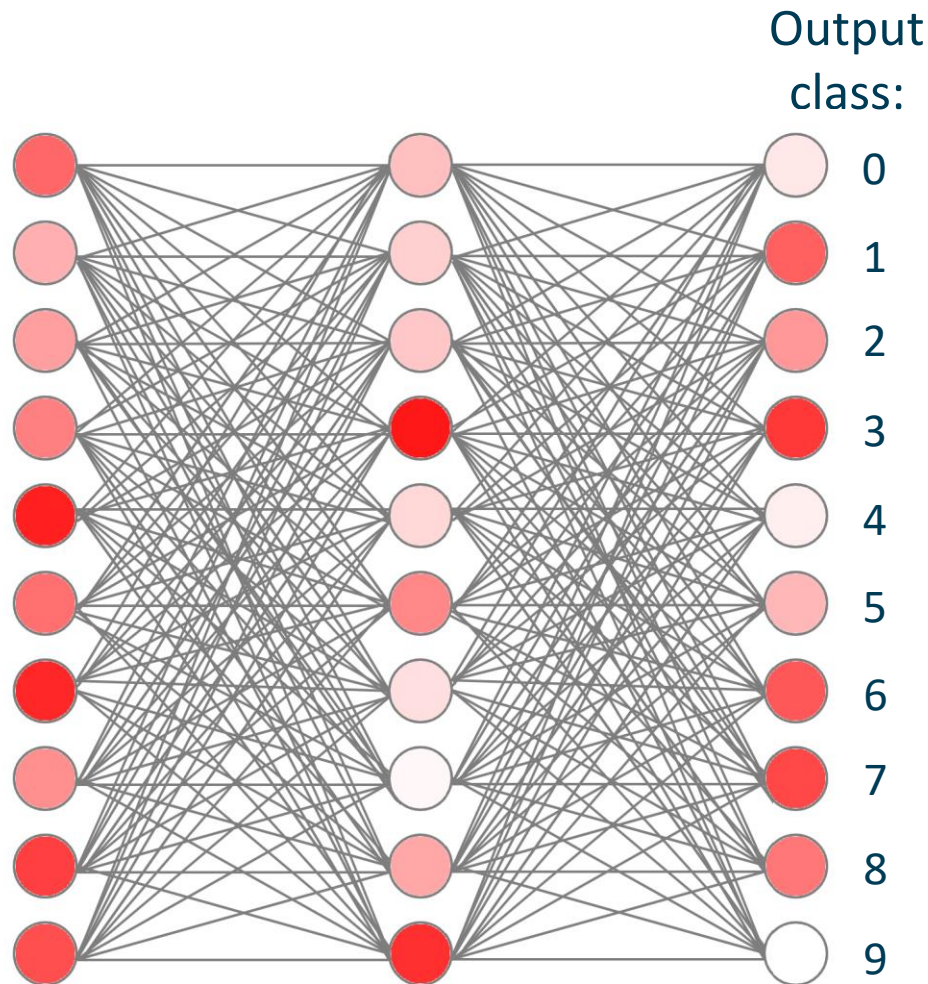


9

*** Prediction results ***

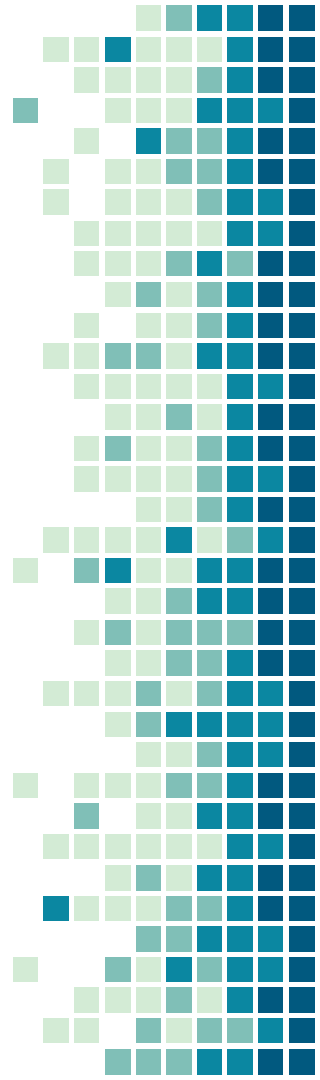
```
3: 77.4911
7: 71.3724
9: 67.3987
6: 63.6552
1: 58.5351
8: 54.1738
2: 49.2659
5: 42.1806
0: 29.3429
4: 26.6442
```

Predizione errata



Analisi dei risultati ottenuti

- Il classificatore mostra talvolta qualche difficoltà nel riconoscere correttamente i samples forniti perché essi sono stati generati a parte (e quindi estranei al dataset originale) di proposito, per testarne le capacità di generalizzazione.
- Anche quando la classe predetta dal classificatore risulta errata, tipicamente lo score relativo alla classe target risulta comunque alto: ne consegue che la differenza rispetto al target risulta bassa e quindi, in linea con la definizione di E e del criticality factor precedentemente illustrate, il rispettivo neurone dell'output layer mostrerà una bassa criticità.
- Di conseguenza il ranking relativo alla NCA risulta valido anche in caso di *misprediction*.



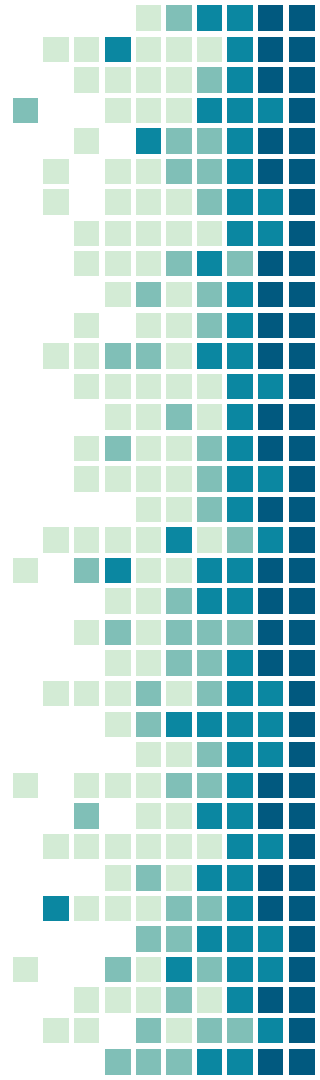
Approximate Computing

Chapter 3



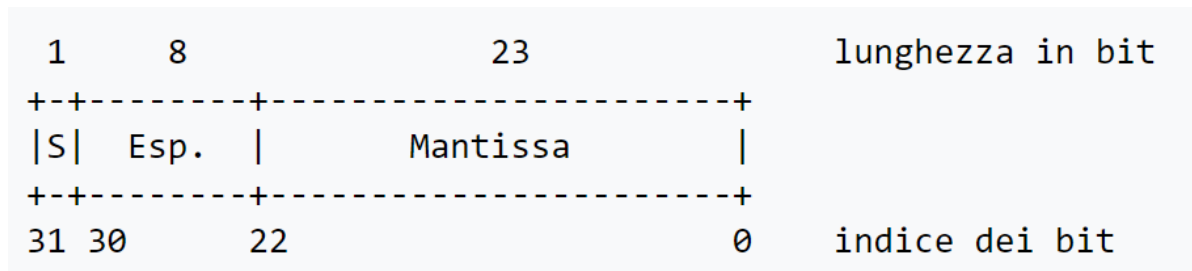
Tecnica di AC utilizzata

- La tecnica di Approximate Computing utilizzata consiste nell'approssimazione dei pesi dei neuroni meno critici.
- L'approssimazione, simulata via software, comporta l'utilizzo di un numero di bit inferiore per lo storage dei pesi del neurone approssimato.
- Il programma sviluppato permette di scegliere liberamente l'incidenza dell'approssimazione in termini di:
- **Numero di neuroni da approssimare** (sulla base del ranking NCA, in ordine crescente di criticità), da un minimo di 1 a un massimo di tutti i neuroni della rete.
- **Numero di bit da utilizzare**, da un minimo di 10 (1 solo bit per la mantissa), a un massimo di 31 (1 solo bit risparmiato per peso).



Note sullo standard IEEE 754

- Codifica dei numeri a precisione singola:
rappresentazione in una stringa di 32 bit:



- Segno: 1 bit (+/-)
- Esponente: 8 bit (0-255)
- Mantissa o Significando: 23 bit
(24 con quello implicito)

Categoria	Esp.	Mantissa
Zeri	0	0
Numeri denormalizzati	0	non zero
Numeri normalizzati	1-254	qualunque
Infiniti	255	0
Nan (not a number)	255	non zero

Note sullo standard IEEE 754 (continuo)

$$VAL = (-1)^{Segno} \cdot 2^{Esponente-127} \cdot [i, < Significando >]_{base2}$$

$$VAL = (-1)^{Segno} \cdot 2^{esponente-127} \cdot \left(1 + \sum_{i=1}^{23} b_{23-i} 2^{-i} \right)$$

- La mantissa è costituita dal numero binario 1, seguito dalla virgola e dalla parte intera del numero rappresentato, in forma binaria; la mantissa risulta così artificialmente compresa tra 1 e 2. Quando un numero è normalizzato, come risulta dal suo esponente, il primo bit della mantissa, pari a 1, viene omesso per convenienza: viene quindi chiamato bit nascosto, o bit implicito.

Implementazione: approssimazione pesi

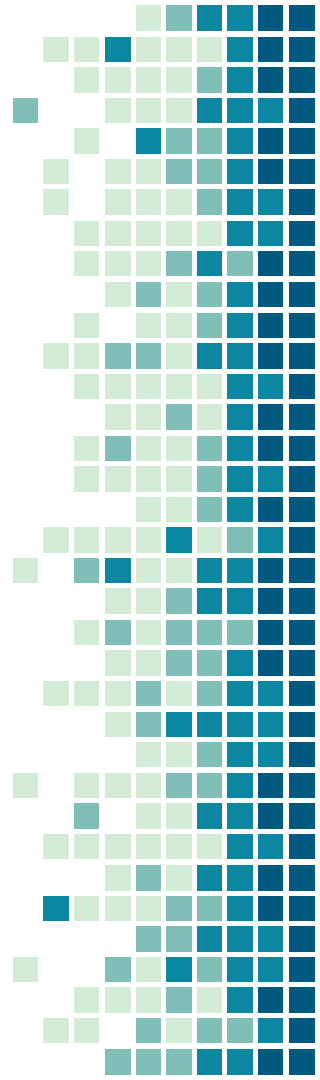
```
// bit approximation function
float roundb(float f, int bits) {
    union {
        // num.i and num.f are mapped on same bits
        int i;
        float f;
    } num;

    bits = 32 - bits;
    num.f = f;
    num.i = num.i + (1 << (bits - 1));    // round instead of truncate
    num.i = num.i & (-1 << bits);        // AND bitwise between mask and rounded value
    return num.f;
}
```

- **Approssimazione:** arrotondamento per eccesso ultimo bit da troncare.
- La rappresentazione binaria di -1 (intero) è costituita da tutti 1: shiftandolo a sinistra di n posti creo la maschera da applicare al valore.
- **Troncamento** utilizzando la maschera così creata tramite AND (&) bit a bit: così facendo simulo via software l'utilizzo di un numero inferiore di bit.

AC: scelta della config. approssimata

- Come già visto, il programma sviluppato permette di scegliere liberamente l'incidenza dell'approssimazione in termini di numero di neuroni da approssimare e numero di bit da utilizzare. Ai fini della valutazione delle prestazioni sono stati utilizzati i seguenti valori dei parametri:
- **Numero di neuroni da approssimare** (sulla base del ranking NCA, in ordine crescente di criticità) = 15.
- **Numero di bit da utilizzare** per i pesi dei neuroni approssimati = 16.



Performance evaluation

Chapter 4



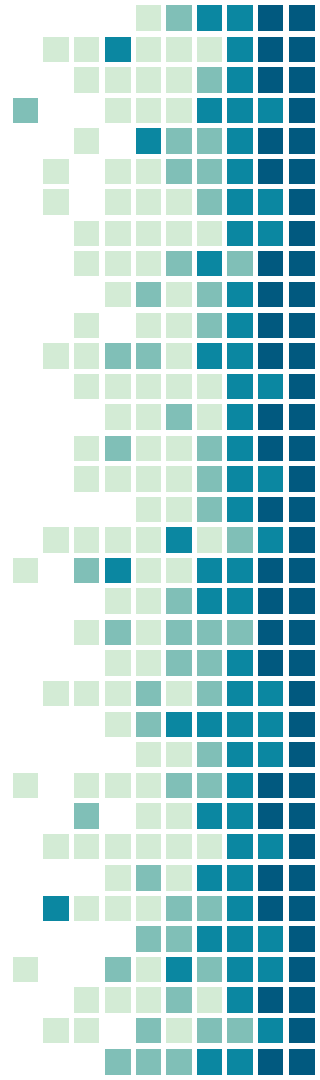
Valutazione config. approssimate

- Si procede adesso alla valutazione della **test accuracy** delle configurazioni approssimate sulla base dei parametri precedentemente visti. Per ciascuno dei ranking NCA precedentemente ottenuti, si valutano le seguenti configurazioni approssimate:
- **Approx:** valutazione test accuracy della rete approssimata.
- **Approx2:** valutazione test accuracy della rete approssimata dopo una nuova fase di training (*fine-tuning*) della stessa.
- **(aggiunta) Approx3:** valutazione test accuracy della rete approssimata allenata *from scratch* (da zero), per valutare la degradazione delle capacità di apprendimento dei neuroni in seguito all'approssimazione dei pesi.

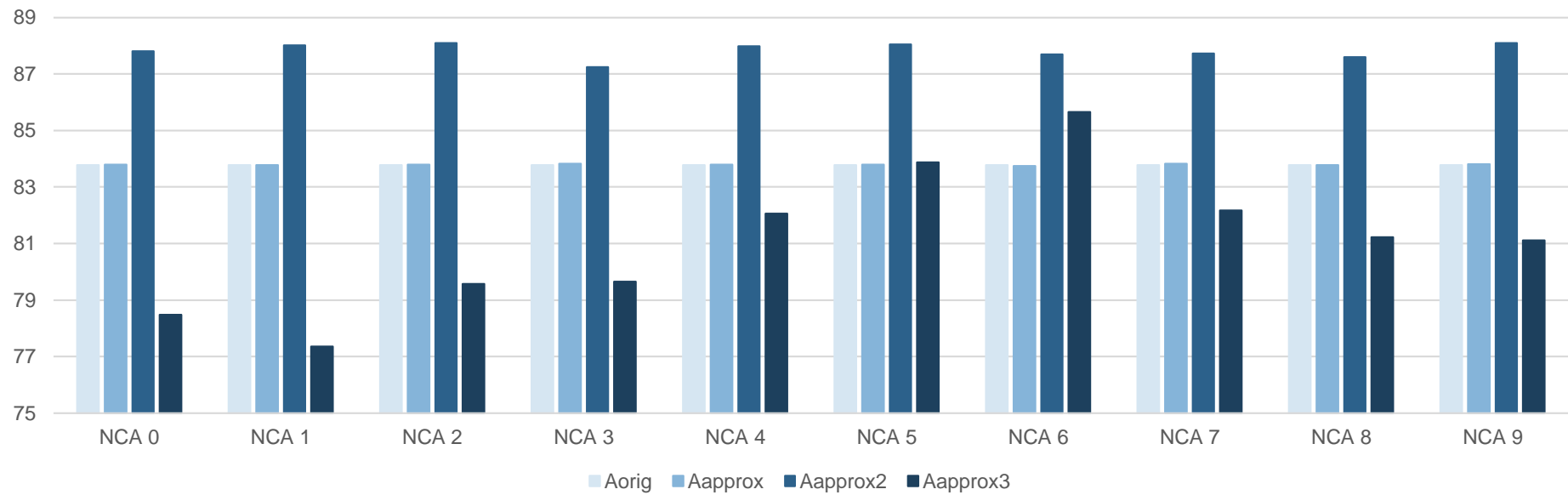


Training Aapprox2 e Aapprox3

- Come già visto, per i training della rete approssimata, sia per il *fine tuning* (Aapprox2) che *from scratch* (Aapprox3), sono stati utilizzati i seguenti valori per gli **hyperparameters**:
- Learning rate = 0.1
- Minibatch size = 64
- Number of epochs = 5
- L'**architettura**, ovviamente, resta invariata, con il numero di neuroni degli hidden layers impostato a 10 per entrambi, per un totale di 30 neuroni (10-10-10).



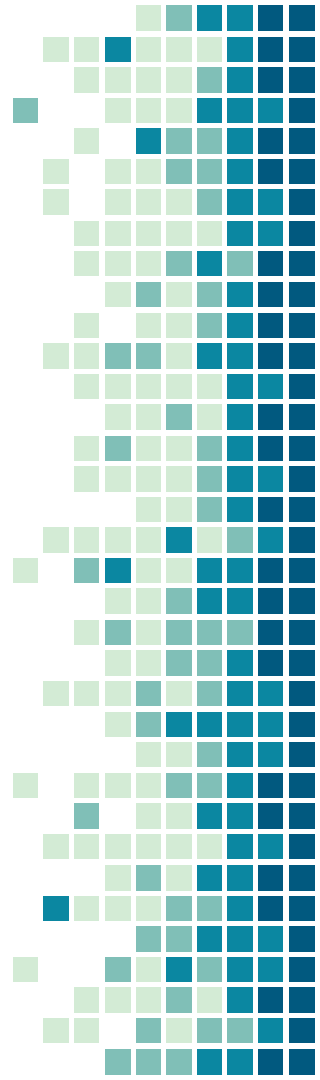
Risultati ottenuti: Test accuracy (%)



	NCA 0	NCA 1	NCA 2	NCA 3	NCA 4	NCA 5	NCA 6	NCA 7	NCA 8	NCA 9
Aorig	83,77	83,77	83,77	83,77	83,77	83,77	83,77	83,77	83,77	83,77
Aapprox	83,80	83,77	83,79	83,82	83,79	83,80	83,75	83,82	83,77	83,81
Aapprox2	87,80	88,01	88,09	87,24	87,98	88,04	87,70	87,72	87,59	88,10
Aapprox3	78,48	77,35	79,57	79,65	82,06	83,87	85,65	82,17	81,22	81,12

Analisi dei risultati ottenuti

- **Aapprox:** la Test accuracy della configurazione approssimata risulta molto simile a quella della configurazione originale, mostrando che l'approssimazione effettuata non influisce significativamente sulle prestazioni della rete.
- **Aapprox2:** sottoponendo la rete approssimata ad una nuova fase di training (*fine tuning*) la test accuracy risulta superiore rispetto a quella della configurazione originale, mostrando la capacità di migliorare il setting di pesi approssimati e non, con ulteriori epoche.
- **Aapprox3:** allenando la rete approssimata *from scratch*, invece, sebbene si raggiunga una buona test accuracy, essa risulta mediamente inferiore a quella della configurazione originale, mostrando inferiori capacità di apprendimento del modello.



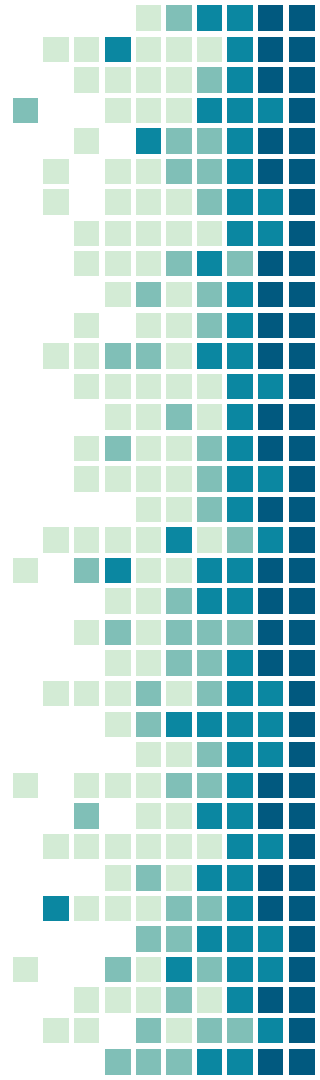
Configurazioni aggiuntive

Chapter 5

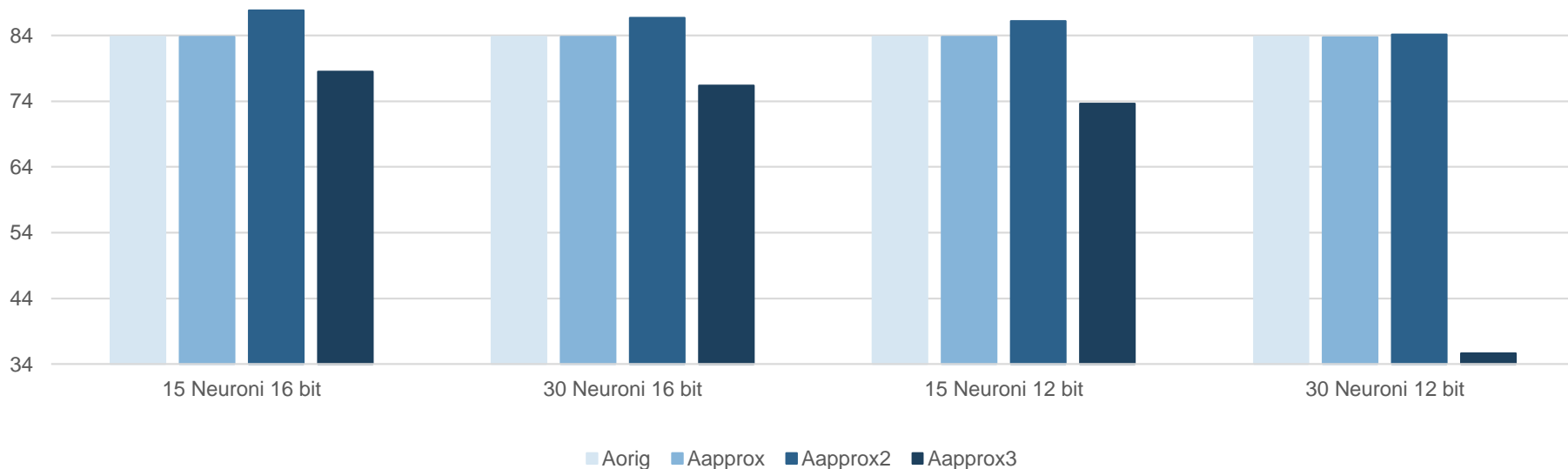


Valutazione config. aggiuntive

- Sulla base dei precedenti risultati si è ritenuto interessante valutare le prestazioni di alcune configurazioni aggiuntive, al variare dei due parametri configurabili nella tecnica di AC, ovvero **numero di neuroni da approssimare** e **numero di bit da utilizzare** per i pesi dei neuroni approssimati.
- A tale scopo, considerando un singolo ranking NCA (NCA 0), si sono prese in considerazione le seguenti configurazioni:
 - **15** neuroni approssimati, **16** bit utilizzati (appr. Originale);
 - **30** neuroni approssimati (intera rete), **16** bit utilizzati;
 - **15** neuroni approssimati, **12** bit utilizzati;
 - **30** neuroni approssimati, **12** bit utilizzati.



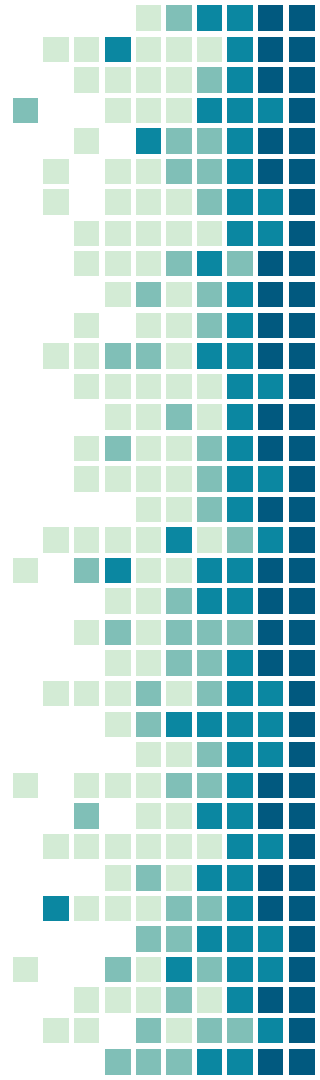
Risultati ottenuti: Test accuracy (%)



NCA 0	15 Neuroni 16 bit	30 Neuroni 16 bit	15 Neuroni 12 bit	30 Neuroni 12 bit
Aorig	83,77	83,77	83,77	83,77
Approx	83,80	83,79	83,80	83,76
Approx2	87,80	86,72	86,23	84,18
Approx3	78,48	76,40	73,63	35,68

Analisi dei risultati ottenuti

- Le performance degradano maggiormente all'aumentare del numero di neuroni approssimati.
- Confrontando le configurazioni **30N 16b** e **15N 12b** notiamo che quest'ultima ha prestazioni peggiori: ciò significa che l'incidenza dell'utilizzo di un numero inferiore di bit è maggiore rispetto a quella dell'aumento del numero dei neuroni approssimati.
- Aumentando il numero di neuroni approssimati e/o diminuendo il numero di bit utilizzati la configurazione *from scratch* (Aapprox3) non è più in grado di raggiungere la test accuracy della configurazione originale, come era accaduto in alcuni casi negli esperimenti precedenti.
- Nella configurazione **30N 12b** l'efficacia del *fine tuning* diminuisce.



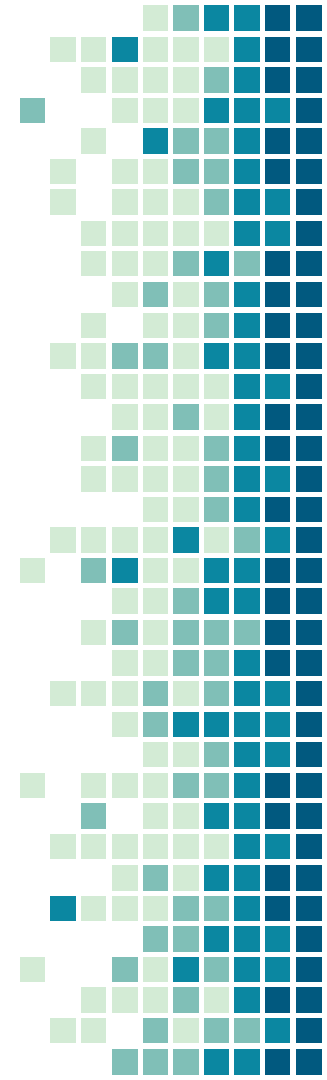
Conclusioni

Chapter 6



Neuron Criticality Analysis

- È stata provata l'**efficacia della NCA**, in quanto l'approssimazione dei soli neuroni meno critici comporta effettivamente una degradazione delle prestazioni inferiore rispetto alle configurazioni con tutti i neuroni approssimati che sono state testate.
- Sono state evidenziate le **differenze nei ranking NCA** al variare della target class: i risultati nei test sono stati differenti, soprattutto per quanto riguarda la configurazione allenata *from scratch* (Aapprox3).



Approximate Computing

- I risultati ottenuti evidenziano come i pesi possano essere rappresentati efficacemente con **16 bit**, senza perdite di prestazioni rilevanti.
- Inoltre, i buoni risultati ottenuti dalla configurazione allenata *from scratch* (Aapprox3) dimostrano che è possibile utilizzare una configurazione approssimata non solo nella fase di **test**, ma anche per il **training** (senza utilizzare un modello pre-trained non approssimato), con conseguenti risparmi nei costi dell'hardware e nei tempi di computazione.

