Impiego della rete di Hopfield nei sistemi di OCR.

Premessa

La presente ricerca si propone di valutare i benefici che potrebbero eventualmente derivare dall'inclusione di una rete di Hopfield in un sistema di OCR, in relazione ad alcune architetture ibride proposte negli anni passati (fra cui [15], [16], [17], [18]). Si tratta di un lavoro di valenza prevalentemente storica in quanto l'inquadramento del problema applicativo si rifà a studi risalenti alla seconda metà agli anni '80, mentre le applicazioni considerate sono del decennio successivo, con la sola eccezione di [18] che è del 2005.

Curiosamente il primo problema da affrontare consiste nello stabilire cosa si intenda per Hopfield Neural Net (HNN) e poi quali caratteristiche siano richieste per l'impiego nei sistemi di OCR.

John J. Hopfield ha infatti proposto due modelli di rete a due anni di distanza l'uno dall'altro. Il modello del 1982 consiste in una rete binaria, auto-associativa, ricorrente, basata sull'impiego di una matrice di pesi simmetrica (tale cioè che $w_{ij} = w_{ji}$), a diagonale nulla ($w_{ii} = 0$), addestrata ricorrendo alla regola di Hebb (vedi [1]).

Il modello del 1984 è invece caratterizzato da una serie di adattamenti aventi lo scopo di renderlo idoneo ad un funzionamento continuo ovvero ad una implementazione hardware mediante tecnologia analogica, fatto di notevole interesse nel quadro della tecnologia disponibile nei primi anni '80 (vedi [2]).

In un panorama pesantemente condizionato dalle considerazioni espresse da Minsky e Papert nella loro celebre analisi [21], i suddetti articoli ([1] e [2]) si concentrarono prevalentemente sulle questioni teoriche, cercando di giustificare i modelli proposti facendo appello ai paradigmi biologico e fisico, delegando ad altri autori gli aspetti applicativi salvo approfondire il problema del commesso viaggiatore ([3]).

Ai suddetti articoli per quanto concerne l'inquadramento teorico si può aggiungere il di Hertz, Krogh e Palmer [6], cui rimanda lo stesso Hopfield (nella voce redatta per Scholarpedia [5])
non avendo mai scritto un libro di suo pugno sull'argomento e che a tutti gli effetti può essere considerato il testo ufficiale di riferimento sulle reti di Hopfield.

Per quanto concerne la tematica di questa indagine, sembrerebbe che John J. Hopfield non si sia mai interessato alla possibilità di impiegare le sue reti per compiti di OCR; tema che non compare neppure in [6]; 'idea di impiegare la HNN nell'ambito delle applicazione di riconoscimento caratteri andrebbe ricondotta al celebre articolo di Lippmann *An Introduction to Computing with Neural Nets*[7], conclusione a cui si può approdare seguendo a ritroso la catena delle citazioni.

L'articolo di Lippmann presenta sei modelli di reti neuronali utilizzabili come classificatori di pattern fissi, evidenziando pregi e difetti di ognuno nel quadro di una tassonomia (tabella 1), che distingue in primo luogo fra reti digitali (input binario) e reti analogiche (input continuo) e quindi fra modelli supervised (che presuppongono la preventiva etichettatura dei pattern in fase di apprendimento) e unsupervised (che non ne necessitano).

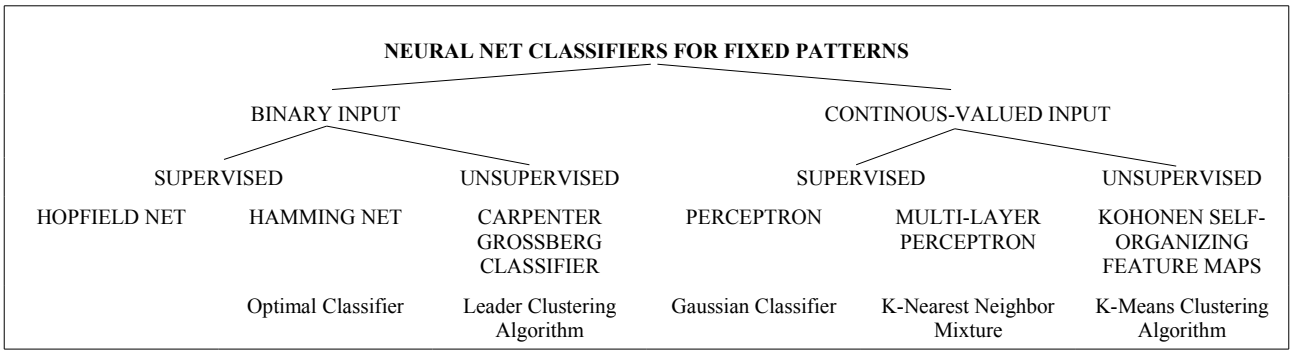

Tabella 1: A taxonomy of six neural nets that can be used as classifiers. Classical algorithms which are most similar to the neural net models are listed along the bottom.

La prima distinzione è fondamentale da un punto di vista tecnologico perché condiziona la possibilità di implementare i corrispondenti circuiti in hardware, mentre il secondo aspetto è determinante in riferimento alle possibilità applicative: potendo le reti supervised essere impiegate come classificatori o memorie associative a differenza delle altre che si prestano piuttosto ad applicazioni incentrate sulla "ricerca delle regolarità". Quest'ultima considerazione non è tassativa, essendo in genere abbastanza facile trasformare gli algoritmi unsupervised in supervised; per cui l'autore non trova difficoltà a proporre tutti e sei i modelli come classificatori di pattern fissi.
Per quanto concerne la HNN Lippmann si spinge a proporre una procedura atta a renderla idonea all'impiego come classificatore. Testualmente: «*When the Hopfield net is used as a classifier, the output after convergence must be compared to the M exemplars to determine if it matches an exemplar exactly . If it does, the output is the class whose exemplar matched the output pattern. If it does not then a "no match" result occurs*».
Viene pertanto suggerita l'adozione di un'architettura ibrida, di fatto l'antesignana della classica configurazione in tandem adottata in quasi tutti i sistemi di OCR che ricorrono alla HNN (vedi ad esempio: [15], [16], [17], [18]), in cui alla rete di Hopfield viene demandata l'attività di pre-processing, in particolare l'eliminazione del rumore e il miglioramento della qualità delle immagini dei grafemi, lasciando ad una rete di Hamming la classificazione vera e propria.
Il suddetto schema si basa sul presupposto che la rete di Hopfield presentasse un elevato livello di immunità al rumore, tanto da consentire affidabilmente la ricostruzione dei pattern originali a partire da versioni degradate, anche in presenza di una notevole quantità di rumore. Convinzione che traeva origine da una serie di esperimenti di ricostruzione di immagini degradate in cui le HNN si erano distinte producendo risultati spettacolari (vedi figura 2.1, [6], pag 12).
La robustezza della HNN pareva inoltre suffragata da una serie di analisi teoriche di vari autori che dimostravano che le reti di Hopfield potevano tollerare, delle piccole deroghe vincoli imposti originariamente circa la simmetricità della matrice, la diagonale nulla, ecc (vedi [6] §3).
Per quanto, alla luce delle considerazioni precedenti, l'idea del tandem sembri estremamente plausibile nasconde un'insidia poiché i grafemi sono caratterizzati da proprietà statistiche alquanto differenti da quelle delle immagini impiegate nelle dimostrazioni citate, problema di cui ci si accorge solo eseguendo dei test. Ma procediamo con ordine. La valutazione della idoneità della HNN all'impiego in applicazioni di OCR e degli eventuali benefici derivanti dal suo utilizzo implicano necessariamente l'esecuzione di test comparativi.

Il benchmark

Poiché tutto sembrava ricondurre all'articolo di Lippmann [7], è stato naturale prenderlo come riferimento; per cui nella progettazione del benchmark è stato fatto il possibile per attenersi alle indicazioni riportate nel testo citato, implementando gli algoritmi seguendo le direttive riportate nei Box illustranti il funzionamento dei vari modelli.
Si è poi cercata un'applicazione che fosse al contempo pertinente e significativa, senza però richiedere un eccessivo lavoro preparatorio, in modo da potersi concentrare sugli aspetti inerenti i temi della classificazione dei pattern e dell'apprendimento, senza spendere troppo tempo sul tema della segmentazione dei grafemi.
Volendo evitare il rischio di ricadere in un problema giocattolo, la scelta è caduta sulla più semplice applicazione di riconoscimento di caratteri avente un interesse commerciale, vale a dire il problema della lettura di stringhe di caratteri numerici, stampati usando un font fisso, scelto fra quelli espressamente progettati per l'utilizzo in applicazioni di OCR (figura 1).
La suddetta applicazione offre diversi vantaggi: in primo luogo la facilità di reperimento delle grandi quantità di campioni necessari per l'addestramento delle reti e per i test.
I suddetti font-OCR sono di impiego così comune che c'è solo l'imbarazzo della scelta: biglietti dell'autobus, assegni, bolle di trasporto, ecc. In generale tutte le applicazioni che prevedono l'utilizzo di documenti standardizzati, che debbono essere leggibili sia da utenti umani che da

sistemi automatici, tipicamente a scopo di controllo.
I documenti considerati sono estremamente omogenei, le scritte da riconoscere si trovano sempre nella stessa posizione, spesso inserite all'interno di un'area di salvaguardia costituita da ampi margini tutt'intorno alla stringa; fatto che semplifica notevolmente la segmentazione del grafema, cioè la copiatura dell'immagine del carattere nella memoria di lavoro in vista della classificazione.
Per i test è stata impiegata una collezione di 1407 campioni, rappresentativi di tutte le cifre numeriche, anche se non nella stessa proporzione trattandosi di un campione "naturale" ottenuto acquisendo alcuni documenti, di tipo omogeneo, stampati usando il font OCR-B (standard ANSI INCITS 49-1975 (R2002)).

La stesura degli algoritmi
Com'è abbiamo anticipato nella stesura degli algoritmi si è fatto il possibile per attenersi alle indicazioni fornite nell'articolo di riferimento [7], cui si rimanda per i dettagli.
In particolare, circa la rete di Hopfield è stata impiegata una HNN di tipo binario come quella originariamente proposta in [1]. A differenza di quanto suggerito da Lippmann si è però optato per una rete monostadio realizzata "aumentando" il pattern di input con l'aggiunta di dieci bit (ingressi fittizi), uno per ognuna delle possibili classi.
La rete di Hamming è stata realizzata seguendo sostanzialmente le indicazioni di Lippmann, salvo eliminare la soglia ft nello step 2 e sostituirla con la funzione ft'(x) = (x <= 0)? 0: x; nello step 3.
Per quanto concerne la rete di Carpenter-Grossberg di cui nel tempo sono state proposte varie versioni (vedi [23]); poiché l'articolo di riferimento [7] presenta una ART-1 ci si è attenuti a tali indicazioni, anche se è noto che i modelli successivi sopperiscono a tutta una serie di problemi che si sono manifestati eseguendo i test. Unica deroga, l'algoritmo è stato modificato in modo da renderlo supervised (forzando in fase di apprendimento la candidatura della classe j in conformità all'etichetta associata al grafema considerato).
Il Perceptron (SLP) è stato realizzato ricorrendo all'algoritmo di Widrow-Hoff, mentre nella back-propagation si fatto ricorso all'algoritmo di accelerazione della convergenza Jacobs [10] utilizzando 36 celle nascoste..
Infine, la rete di Kohonen è stata trasformata in supervised utilizzando la tecnica descritta in [8].
Al fine di rendere confrontabili i risultati input e output sono stati standardizzati in modo che tutte le reti accettassero in input una matrice 12 x 8 pixel, restituendo i risultati in un array di 10 celle.

Risultati dei test
I test sono stati organizzati in due tornate: nel corso della prima tutti gli algoritmi sono stati addestrati utilizzando dieci campioni, uno per ciascuna cifra numerica, identici per tutti i modelli considerati; dopo di che si sono verificati i risultati ottenuti classificando i 1408 campioni disponibili. I risultati ottenuti sono riportati nella Appendice-A.
Il test è stato quindi ripetuto addestrando di volta in volta le reti con un numero crescente di pattern casuali (scelti in modo da garantire per ogni classe lo stesso numero di campioni in sede di apprendimento), allo scopo di verificare l'incidenza della numerosità del campione sulle prestazioni dei vari modelli di rete. I risultati ottenuti sono riportati nella Appendice-B.

Prima di addentrarci nella discussione dei risultati ricordiamo, a titolo di riferimento, che nelle applicazioni di OCR commerciali, del tipo considerato (riconoscimento delle sole cifre numeriche in riferimento ad un font prefissato), è normale richiedere e attendersi un tasso di errore assai inferiore a 1/1000. Ciò potrebbe sembrare un requisito molto stringente se non fosse che solitamente le stringhe che si desidera leggere sono costituite ciascuna da almeno una decina di cifre per cui un tasso di errore dello 0,1%, riferito ai grafemi, equivale a sbagliare la lettura di un documento, fatto più che sufficiente per rendere l'applicazione inaccettabile..
Conseguentemente tutti i risultati riportati in appendice Appendice-A debbono essere considerati

largamente insoddisfacenti e i migliori in Appendice-B appena accettabili. Si tratta naturalmente di un risultato atteso e facile da spiegare.
A dispetto di tutti gli sforzi dei tipografi e dei fabbricanti di stampanti per garantire la massima uniformità di stampa, a causa delle caratteristiche intrinseche del materiale di supporto e dell'usura del meccanismo di stampa, indipendentemente dalla tecnologia impiegata, è praticamente impossibile stampare due grafemi identici. Se si guarda con una lente sufficientemente potente i caratteri stampati anche nella stessa pagina appariranno tutti leggermente diversi gli uni dagli altri.
Tale problema è aggravato dall'impossibilità di allineare il retino di stampa con quello di acquisizione; ragione per cui i grafemi che si vorrebbe classificare, se anche fossero stati stampati in maniera perfetta, risulterebbero comunque diversi gli uni dagli altri a causa delle differenze di fase fra i retini. Tali problemi sono ben noti così come la loro soluzione consistente nell'addestrare le reti presentando un numero più o meno elevato di esemplari per ogni classe, in modo da permettere alla rete di costruirsi un modello della variabilità dei grafemi. C'è purtroppo un altro inconveniente in quanto non tutte i modelli di rete neuronale sono in grado di beneficiare delle informazioni supplementari fornite proseguendo l'apprendimento, non solo, le caratteristiche peculiari dei singoli esemplari presentati, pur essendo scelti in maniera casuale, se il campione è troppo piccolo possono favorire una rete a discapito di un'altra poiché la sequenza ottimale di presentazione dei campioni per l'addestramento non è necessariamente la stessa per tutti i modelli.
Per ovviare ai suddetti inconvenienti ed effettuare una valutazione il più possibile obiettiva si è deciso di eseguire due prove: un primo test presentando ad ogni rete la medesima sequenza di pattern, costituita da un singolo esemplare di ciascun carattere, in successione crescente da '0' a '9'.
Nella seconda sessione si è cercato invece di valutare il beneficio derivante dalla presentazione di sequenze di campioni più lunghe, per quanto concerne i modelli che possono trarne beneficio (cioè tutti tranne la rete di Hamming).
Inaspettatamente il primo test ha fornito dei dati significativi solo per quattro delle sei reti.
Esaminando le matrici di confusione riportate nell'Appendice-A, emerge sia pure di poco la prestazione della rete di Hamming (27 errori su 1408 pattern). Seguono a ruota gli altri modelli con l'eccezione della back-propagation e della HNN di cui si dirà poi. Un risultato poco entusiasmante, assai lontano degli obiettivi auspicati (1 errore su 1000), utile tuttavia a farci capire che per ottenere le prestazioni richieste dal mercato non si può prescindere dal ricorso a procedure di apprendimento che prevedano la presentazione di numerosi esemplari di ciascuna classe.
Fa eccezione la rete di Hamming la quale, essendo basata sul confronto diretto del grafema da riconoscere con ciascuno dei prototipi ideali delle diverse classi, per la sua stessa natura non può beneficiare di una procedura di apprendimento basata sulla presentazione di sequenze di pattern. Volendo migliorare le prestazioni della rete di Hamming il meglio che si può fare è utilizzare come titoli noti le matrici di stampa, ovvero i pattern ottenuti dai modelli matematici dei grafemi, riscalati alla risoluzione considerata per il confronto. Nelle suddette condizioni la rete di Hamming funziona al meglio in quanto implementa il classificatore binario ottimale; tuttavia anche in assenza delle specifiche dei font è possibile ottenere dei buoni risultati avendo semplicemente cura di scegliere come campioni dei pattern di buona qualità, come appare dalla matrice nell'angolo in basso a sinistra di pagina 11 che è caratterizzata da un tasso di errore notevolmente più basso di quello della prima matrice ottenuta facendo riferimento a pattern scelti a caso. In conclusione: la rete di Hamming è la soluzione più indicata quando è richiesto il riconoscimento di grafemi appartenenti ad un font fisso ed è disponibile un solo campione per classe.
Il pessimo risultato ottenuto dai perceptron e quello ancora peggiore della back-propagation sono imputabili a problemi connaturati alle regole di apprendimento da essi utilizzate.
La regola di Widrow-Hoff presuppone l'aggiornamento di tutte le celle dopo la presentazione di ciascun campione, indipendentemente dall'etichetta (classe) ad esso associata, per cui se i pattern presentati sono parzialmente correlati (hanno cioè dei pixel attivi in comune), si può verificare un fenomeno di interferenza, per cui le modifiche di volta in volta apportate vanno a detrimento di

quelle effettuate poc'anzi (un po' come Penelope che disfaceva di notte ciò che aveva tessuto di giorno). Ragione per cui presentando i grafemi nell'ordine da “0” a “9” e quindi procedendo con il test, molti “0” vengono classificati come “9” o come “8” ma non come “6”, pur essendo quest'ultimo grafema identico al “9” a meno di una rotazione.
Considerazioni analoghe valgono per la back-propagation dove gli stessi fenomeni si manifestano in maniera ancora più accentuata poiché il corretto funzionamento del modello presuppone una inizializzazione casuale delle matrici di pesi (al fine di garantire, ripetendo l'apprendimento, di non ricadere negli stessi minimi locali delle sessioni precedenti). Ma ora, affinché il processo di discesa del gradiente abbia effetto e si svolga regolarmente occorre che le modifiche dei pesi siano lente e graduali; requisito palesemente in contraddizione con l'obiettivo di avere un classificatore che funzioni “al meglio” dopo la presentazione di un singolo esemplare di ogni classe, come richiesto nel primo test.
Al termine della presentazione dei primi dieci pattern, cioè della prima epoca di apprendimento, il meccanismo di aggiornamento dei pesi è ancora all'inizio della fase di avvio. A causa della costante di smorzamento α, avente lo scopo di evitare comportamenti oscillatori nella dinamica della rete[1], i pesi sono ancora praticamente quelli iniziali per cui la BP si trova a lavorare completamente fuori specifica, conseguentemente non si è ritenuto utile riportare nell'Appendice-A la matrice di confusione, poiché i risultati ottenuti sono praticamente quelli conseguenti all'inizializzazione casuale dei pesi.
Come risulta dall'esame dei risultati riportati in Appendice-B, sono stati necessarie almeno 70 epoche di apprendimento e la presentazione di circa 700 grafemi prima che il meccanismo della back-propagation vada a regime, dopo di che i miglioramenti si susseguono ad un ritmo sostenuto, come si osserva confrontando i risultati di cui sopra con quelli ottenuti all'epoca 80.
Si noti, tuttavia, che la curva del tasso di errore non ha un andamento monotonico decrescente, talvolta accade che il tasso di errore cresca invece di diminuire, come accade in corrispondenza dell'epoca 100. Procedendo ad oltranza è tuttavia possibile rendere il tasso di errore molto piccolo, come si evince dall'esame dei risultati riportati in Appendice-C.
Anche la rete di Kohonen e il single layer perceptron traggono vantaggio da un addestramento prolungato. Aumentando il numero di campioni, forniti per ogni classe, il tasso d'errore tende a generalmente a diminuire, anche se occasionalmente può risalire (quando vengono presentati dei pattern anomali poiché distorti o inquinati da rumore).
Fa eccezione invece la ART in cui il tasso d'errore scende (per 20 campioni) per poi risalire. Tale comportamento è una diretta conseguenza della strategia di apprendimento originariamente adottata da Carpenter e Grossberg nella formulazione originale della Adaptive Resonance Theory (ART-1). Considerata una generica classe di equivalenza Cj, la rete infatti apprende modificando la matrice di pesi in modo da creare una maschera costituita da tutti, e soli, gli input che sono attivi in tutti gli esemplari noti della suddetta classe, imitando una strategia di apprendimento che era stata già ampiamente testata nell'ambito dell'apprendimento simbolico[2], salvo accorgersi in un secondo tempo dell'esistenza di una sostanziale differenza fra le entità considerate dagli algoritmi di apprendimento simbolico e le feature di basso livello gestite dalle reti neurali, consistente nella diversa incidenza del rumore presente nei pattern. Nel primo caso il “rumore” è costituito prevalentemente dalla presenza nella descrizione del pattern di entità simboliche ininfluenti, che giustamente possono e debbono essere ignorate, per cui è corretto concentrarsi sulle poche caratteristiche comuni a tutti gli esemplari noti della classe, pervenendo ad una definizione più adeguata.
Nel secondo caso siamo in presenza di rumore “selvaggio” che può alterare il valore di un qualsiasi pixel, significativo o meno, in modo del tutto casuale, per cui è evidente che se si protrae il processo

1 Vedi Box 6, Step 4, nell'articolo di Lippmann.
2 La procedura di apprendimento utilizzata dalla ART-1 ricorda la *dropping condition rule* utilizzata per individuare le *maximally-specific conjuntive generalizations* nell'apprendimento induttivo simbolico (vedi [24] §3).

di apprendimento abbastanza a lungo eliminando progressivamente tutti i pixel che assumono valore nullo prima o poi ci si ritroverà con una maschera costituita da soli zeri.
Per ovviare al suddetto problema già negli anni '80 Carpenter e Grossberg modificarono la regola di apprendimento, quasi in contemporanea con la pubblicazione dell'articolo di Lippmann, introducendo prima la rete ART2 e poi ART-3 (vedi [23] e [19] § 16.2.1).

La rete di Hopfield

La HNN sviluppata seguendo le indicazioni fornite nel Box-1 dell'articolo di Lippmann [7], si comporta esattamente come nel testo citato, salvo per un dettaglio: la capacità di memorizzazione notevolmente inferiore alle previsioni teoriche.
In base alle simulazioni effettuate da Hopfield ricorrendo al metodo Montecarlo [1] e all'analisi teorica di Amit, Gutfreund, Sompolinsky[3] [20], la capacità di memorizzazione della HNN dovrebbe attestarsi attorno a 0,14N, dove N indica il numero di nodi che la compongono. Avendo standardizzato una matrice di input di 96 elementi, per cui ci si aspetterebbe una capacità di circa 96 · 0.14 = 13,4 grafemi, compatibile con le necessità dell'applicazione considerata (che richiede la memorizzazione di 10 cifre numeriche), sfortunatamente, ad onta di ogni sforzo non è stato possibile memorizzare nella rete più di sette pattern; fatto che ha precluso ogni possibilità di procedere con i test!

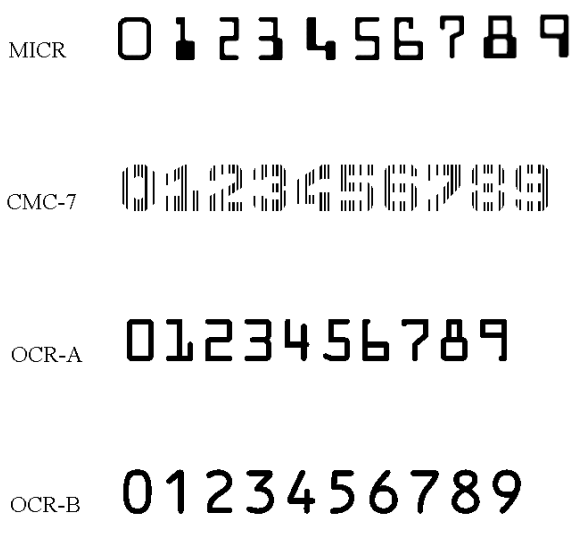

Figura 1) Esempi di font espressamente progettati per il riconoscimento automatico.

Anche passando ad una matrice più grande (24 x 16) in risultato non cambia: anche in assenza di rumore presentando uno zero viene richiamato un pattern simile ad un otto, il due richiama il tre, il tre richiama il cinque, mentre gli altri grafemi vengono invece richiamati correttamente.
Il fenomeno riscontrato potrebbe essere dovuto ad eccessiva correlazione fra i grafemi, del resto lo stesso Hopfield, già nel suo primo articolo [1], metteva in guardia contro il rischio costituito dagli attrattori spuri che si possono formare nel caso in cui la distanza di Hamming fra i pattern memorizzati sia insufficiente. Tesi suffragata dall'analisi illustrata in [26].
Ciò spiegherebbe molte cose: dagli ottimi risultati ottenuti memorizzando immagini complesse (come quelli in [6], pag 12), ai risultati positivi riportati da vari autori che hanno compiuto degli esperimenti impiegando grafemi disegnati ad hoc (come quelli utilizzati dallo stesso Lippmann e illustrati in figura 5 dell'articolo citato), anziché dei caratteri "naturali" acquisiti con uno scanner. Potrebbe quindi darsi che più o meno inconsapevolmente gli autori abbiamo "giocato" con le forme dei caratteri sino a trovare un insieme capace di soddisfare le loro esigenze.
Per quanto suggestiva, purtroppo la suddetta ipotesi non suggerisce alcuna via d'uscita: data l'impossibilità di modificare i documenti che sono già stati stampati (e che si vorrebbe leggere), tanto meno di intervenire sulle caratteristiche grafiche dei font in commercio per adattarli alle esigenze dell'algoritmo di apprendimento.
Persino la creazione di nuovi font offre poco margine di intervento poiché il successo commerciale di questi si basa su considerazioni di natura prevalentemente estetica, ed essendo una sequenza di grafemi stilisticamente omogenei più gradevole alla vista, i grafici tendono a disegnare i caratteri riproponendo un ristretto numero di elementi stilistici (grazie, spessore e lunghezza delle aste, ecc.),

3 Vedi anche [6] § 2.5 e § 10.

per cui è naturale che i grafemi di uno stesso font presentino un elevato livello di correlazione.
Anzi, è probabile che il font OCR-B, essendo stato progettato espressamente per il riconoscimento automatico di caratteri, sia già caratterizzato da un grado di ortogonalità di gran lunga superiore alla media, per cui è difficile che cambiando font le prestazioni possano migliorare. Comunque per fugare ogni dubbio, il test di memorizzazione è stato ripetuto impiegando altri due font (OCR-A e MICR, vedi figura 1), scelti fra quelli appositamente progettati per l'OCR e dotati apparentemente di una maggiore ortogonalità, ottenendo all'incirca gli stessi risultati.
Per quanto concerne la letteratura specifica, inerente l'impiego della rete di Hopfield in applicazioni di OCR, molti autori glissano sul problema della memorizzazione dei grafemi, altri affermano di essere ricorsi a procedimenti di ortogonalizzazione come quello di Gram-Schmidt [15], oppure a complesse operazioni di preprocessing basate sul ricorso a DCT, FFT, Wavelet, ecc.[16][17][18]. Tutti algoritmi molto costosi dal punto di vista computazionale.
Ipotizzando l'impiego di N input e M output (M << N), la corrispondente HNN richiederà N^2 pesi[4] contro i gli M * N, di rete di perceptron, per cui il costo di una singola iterazione della HNN risulta N/M volte più costosa in termini di tempo. Se si aggiunge il fatto che per ottenere la convergenza della HNN occorrono in media K iterazione, a parità di condizioni, il calcolo del valore di uscita della HNN è almeno K*N/M volte più costoso di quello del corrispondente perceptron.
Posto: M = 10, N = 100, K = 5, si deduce che nelle condizioni del benchmark il costo computazionale di una rete di Hopfield è almeno 50 volte superiore a quello di un perceptron!
Se a ciò si aggiungono i costi del preprocessing richiesto per l'ortogonalizzazione dei pattern o il calcolo della FFT è molto difficile giustificare il ricorso alla HNN sul piano dell'efficienza.
Piuttosto che stravolgere il modello di riferimento, memorizzando dei pattern trasformati che poco hanno a che vedere con i grafemi originali, ho preferito accogliere il suggerimento riportato in [14] e in [19]: addestrare la rete di Hopfield ricorrendo alla Delta Rule.
La rete così modificata viene talvolta chiamata Optimized Hopfield Net (OHN) o anche rete di Sigillito dal nome di chi ne ha patrocinato l'utilizzo (vedi [13], [14]), anche se una soluzione molto simile era già stata suggerita precedentemente da altri (vedi [12] §4).
Alla base di tutto c'è l'osservazione che le celle della rete di Hopfield sono di fatto dei perceptron (Rojas [19] §13.4, [12] pag. 93-95). La regola di apprendimento di Sigillito si ricava analogamente a quella di Hopfield minimizzando la funzione di errore:

$$E = \tfrac{1}{2} \sum_s^{\rho} \sum_i^n (e_i^s - \sum_k^n w_{ik}\, e_k^s)^2$$

Derivando E rispetto a w_{ik} si deduce la seguente regola di apprendimento iterativa:
supponendo che inizialmente sia $w_{ik} = 0$

$$\Delta^s w_{ij} = \eta\,(e_i^s - \sum_k^n w_{ik}\, e_k^s)\, e_j^s \qquad i, j = 1, 2, ..., n \qquad s = 1, 2, ..., \rho$$
$$w_{ij} = w_{ij} + \Delta^s w_{ij}$$
$$w_{ii} = 0 \qquad i = 1, 2, ..., n$$

dove η è la costante che determina la velocità di apprendimento. Tipicamente $\eta = 1/N$, dove N rappresenta il numero degli ingressi ovvero dei nodi trattandosi di una rete auto-associativa.
Apportate le debite modifiche alla regola di apprendimento, l'OCR ha subito preso a funzionare, consentendo addirittura di ritornare alla matrice di input 8 x 12 con cui sono stati fatti tutti test.
Presentando dieci volte la solita sequenza di 10 campioni del primo test si ottiene l'esito riassunto nella matrice di confusione sottostante (28 errori su 1408 pattern), un risultato di poco migliore di quello ottenibile riaddestrando il perceptron seguendo un'analoga procedura.

Optimized Hopfield Net, campioni 10, file: hopf.xar tempo: 184s, memoria 100352 Byte.

4 I benefici derivanti dal computo di una sola delle due matrici triangolari identiche sono vanificati dall'inefficienza derivante dall'impossibilità di ricorrere alle istruzioni vettoriali disponibili in tutti i processori moderni nonché dalla maggiore complessità del codice di controllo.

	0	1	2	3	4	5	6	7	8	9		Cml-1
0	398		1	4				1			404	403
1		98									98	501
2			126								126	627
3				113		17					130	757
4					71						71	828
5						68					68	896
6						1	81				82	978
7			1					135			136	1114
8	2								115		117	1231
9			1							175	176	1407
	400	98	129	117	71	86	81	136	115	175	1408	

Perceptron, campioni 10, file: slp10_10.xar, tempo: 168s, occupazione di memoria: 7760Byte.

	0	1	2	3	4	5	6	7	8	9		Cml-1
0	397			1							398	397
1		98		1							99	496
2			126								126	622
3				115		28					143	765
4					71						71	836
5						56					56	892
6						2	81				83	975
7								136			136	1111
8	3		3						115		121	1232
9										175	175	1407
	400	98	129	117	71	86	81	136	115	175	1408	

La OHN è superiore sotto tutti i punti di vista alla HNN: per quanto concerne la convergenza solitamente bastano due o tre cicli per pervenire ad uno stato stabile, contro i cinque, dieci della rete di Hopfield. Anche la capacità di memoria è alquanto maggiore tanto che è possibile memorizzare almeno 36 grafemi, nella rete 12 x 8, senza problemi.
Tuttavia nelle condizioni del secondo test (addestramento mediante presentazione di pattern scelti casualmente, garantendo però la presentazione di un numero di campioni uguale per tutte le classi), la OHN sembrerebbe trarre meno benefici da un addestramento prolungato rispetto ad altri modelli di rete, come si rileva confrontando i risultati riportati nella matrice seguente con quelli della matrice nell'angolo in basso a destra a pagina 12.

Optimized Hopfield Net, file hopf10.xar, 100 campioni, epoca 10, JImg0.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		Cml-1
0	398		2	3							403	402
1		98									98	500
2			126								126	626
3				113		14					127	753
4					71						71	824
5				1		71					72	896
6						1	81				82	978
7								136			137	1115
8	2								115		117	1232
9										175	175	1407
	400	98	129	117	71	86	81	136	115	175	1408	

Proseguendo l'addestramento ad oltranza, analogamente a ciò che accade con il perceptron il tasso di errore inizialmente scende (fino alla epoca 500), per poi tornare a risalire.

Optimized Hopfield Net, file h02k0.xar, 2000 campioni, epoca 500, JImg0

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**		Cml-1
0	400		1	1		1					403	402
1		98									98	500
2			128								128	628
3				115							115	743
4					71						71	814
5						85					85	899
6							81				81	980
7								136			136	1116
8									115		115	1231
9				1						175	176	1407
	400	98	129	117	71	86	81	136	115	175	1408	

Poiché non è possibile valutare la tolleranza al rumore di un modello di rete neuronale facendo riferimento ad una sola prova (come nel caso delle matrici di confusione riportate in Appendice-C), i test sono stati ripetuti dieci volte in modo da ottenere delle statistiche significative, utilizzando per ciascuna rete le matrici di pesi che avevano fornito i risultati migliori nei test in Appendice-B.

La tabella sottostante riporta i risultati ottenuti eseguendo i test di riconoscimento aggiungendo il 10% di rumore casuale, cioè commutando lo stato del 10% dei pixel scelti a caso.

	media/1407	varianza/1407	media/1000	varianza/1000	matrice
Hopfield	15,3	9,34	10,8	4,65	h02k0.xar
Back-Propag.	1,7	0,46	1,2	0,23	buoni1.xar
Kohonen	1,11	1,61	0,71	0,77	ko10000.xar

Si lascia al lettore il compito di valutare l'opportunità di impiegare la rete di Hopfield in eventuali applicazioni di OCR.

Conclusione

A mio avviso gran parte dell'entusiasmo e della popolarità delle reti di Hopfield, è ascrivibile al fascino esercitato dalla procedura di ricostruzione dei pattern a partire da una loro versione deteriorata, di indubbia spettacolarità e alla facilità con cui è possibile implementare tale rete in hardware[5]. Si noti che ancor oggi la maggior parte degli articoli concernenti l'implementazione hardware di reti neuronali ha per oggetto la rete di Hopfield.

Sul finire degli anni '80, molti ricercatori intrapresero lo studio della rete di Hopfield, forti di stime in base alle quali implementando reti neuronali in silicio, ad imitazione di quelle biologiche, avrebbe dovuto essere possibile ottenere una potenza di calcolo di due o tre ordini di grandezza superiore a quella fornita dalle simulazione software[6], per cui era legittimo ritenere che le inefficienze del modello di Hopfield (peraltro già denunciate, ad esempio in [12] § 3, pag. 70), potessero essere ampiamente compensate dalla potenza di calcolo ottenibile grazie al maggior parallelismo.

Attualmente, prescindendo dalla possibilità di una implementazione hardware, volendo realizzare un programma di OCR basato sull'impiego di reti neuronali, a mio avviso esistono varie alternative più efficienti della rete di Hopfield, a cominciare dalla back-propagation che in questi anni è diventata praticamente uno standard, grazie alla sua robustezza e all'affidabilità dimostrata in innumerevoli applicazioni.

5 Vedi [6] § 3.4 e [25] §8, §9.

6 Secondo le stime di Kosko: 2000 MIPS ricorrendo a circuiti VLSI contro i ~10 MIPS del software ([25] pag. 231).

Appendice-A

Hamming, campioni 10, file: hamm2.xar, tempo: 102s, occupazione di memoria: 8560Byte.

	0	1	2	3	4	5	6	7	8	9	
0	399			1							400
1		98		1							99
2			128								128
3				115		23					138
4					71						71
5						63					63
6							81				81
7			1					136			137
8	1								115		116
9										175	175
	400	98	129	117	71	86	81	136	115	175	1408

Totale Errori: 27/1408

Perceptron, campioni 10, file: y0.xar, tempo: 168s, occupazione di memoria: 7760Byte.

	0	1	2	3	4	5	6	7	8	9	
0	337			1							338
1		98									98
2			124								124
3				111							111
4					71						71
5						73					73
6							81				81
7			3	1				136			140
8	25		1	4		13			115		158
9	38		1							175	214
	400	98	129	117	71	86	81	136	115	175	1408

Totale Errori: 87/1408

Carpenter-Grossberg, campioni 10, file: art0.xar, tempo: 112s, occupazione di memoria: 15360Byte

	0	1	2	3	4	5	6	7	8	9	
0	400			1							402
1		98									98
2			125								125
3				115		28					143
4					71						71
5						58					58
6							81				81
7			4	1				136			141
8									114		114
9										175	175
	400	98	129	117	71	86	81	136	115	175	1408

Totale Errori: 34/1408

Kohonen, campioni 10, file: koho0.xar, tempo: 102s, occupazione di memoria: 7680Byte

	0	1	2	3	4	5	6	7	8	9	
0	399			1							400
1		98		1							99
2			128								128
3				115		28					143
4					71						71
5						58					58
6							81				81
7			1					136			137
8	1								115		116
9										175	175
	400	98	129	117	71	86	81	136	115	175	1408

Totale Errori: 32/1408

Hamming, campioni 10, file: ham10.xar, tempo: 102s, occupazione di memoria: 8560Byte.

	0	1	2	3	4	5	6	7	8	9	
0	400										400
1		98		1							99
2			128	1							129
3				115		1					116
4					71						71
5						85					85
6							81				81
7								136			136
8									115		115
9			1							175	176
	400	98	129	117	71	86	81	136	115	175	1408

Totale Errori: 4/1408

Nota:
le appendici A e B riportano le matrici di confusione corrispondenti ai risultati dei test; in esse le righe corrispondono ai valori osservati, le colonne a quelli attesi.

Appendice-B

Perceptron, campioni 10, file: y1.xar, rErrAve: 0,274

	0	1	2	3	4	5	6	7	8	9	
0	350			1							351
1		98									98
2			127								127
3				69							69
4					71						71
5				29		85					114
6						1	75				76
7			1	16				136			153
8	50			1			6		115		172
9			1	1						175	177
	400	98	129	117	71	86	81	136	115	175	1408

Perceptron, campioni 20, file: y2.xar, rErrAve: 0,274

	0	1	2	3	4	5	6	7	8	9	
0	396			1							397
1		98									98
2			128								128
3				115							115
4					71						71
5				1		85					86
6							78				78
7								136			136
8	4					1	3		115		123
9			1							175	176
	400	98	129	117	71	86	81	136	115	175	1408

Perceptron, campioni 40, file: y4.xar, rErrAve: 0,220

	0	1	2	3	4	5	6	7	8	9	
0	400			1							401
1		98									98
2			129								128
3				113							113
4				2	71						71
5						85					87
6							81				81
7								136			136
8						1			115		116
9			1	1						175	177
	400	98	129	117	71	86	81	136	115	175	1408

Perceptron, campioni 60, file: y6.xar, rErrAve: 0,208

	0	1	2	3	4	5	6	7	8	9	
0	400			1							399
1		98									98
2			129	1							130
3				115							115
4					71						71
5						86					86
6							81				81
7								136			136
8									115		115
9										175	176
	400	98	129	117	71	86	81	136	115	175	1408

Perceptron, campioni 80, file: y8.xar, rErrAve: 0,186

	0	1	2	3	4	5	6	7	8	9	
0	400										400
1		98									98
2			129	2							131
3				115							115
4					71						71
5						86					86
6							81				81
7								136			136
8									115		115
9										175	175
	400	98	129	117	71	86	81	136	115	175	1408

Perceptron, campioni 100, file: y10.xar, rErrAve: 0,165 Tempo: 168s

	0	1	2	3	4	5	6	7	8	9	
0	400		1								401
1		98									98
2			128	2							130
3				115		1					116
4					71						71
5						85					85
6							81				81
7								136			136
8									115		115
9										175	175
	400	98	129	117	71	86	81	136	115	175	1408

Carpenter-Grossberg, campioni 10, file: art1.xar

	0	1	2	3	4	5	6	7	8	9	
0	400			1							401
1		98									98
2			128								128
3				111		1					112
4					71						71
5				4		83					87
6						2	81	1	19		103
7			1	1				135			137
8									96		96
9										175	175
	400	98	129	117	71	86	81	136	115	175	1408

Carpenter-Grossberg, campioni 60, file: art6.xar

	0	1	2	3	4	5	6	7	8	9	
0	399										399
1		98									98
2			128	1					3		133
3				115		11			4		130
4					71						71
5						75					75
6							81		84		165
7			1					136			137
8									24		24
9				1						175	176
	400	98	129	117	71	86	81	136	115	175	1408

Carpenter-Grossberg, campioni 20, file: art2.xar

	0	1	2	3	4	5	6	7	8	9	
0	400										
1		98									
2			128	1		3					
3				115							
4					71						
5						83					
6							81				
7								136			
8									115		
9			1	1						175	
	400	98	129	117	71	86	81	136	115	175	1408

Carpenter-Grossberg, campioni 80, file: art8.xar

	0	1	2	3	4	5	6	7	8	9	
0	399										399
1		98									98
2	1		128	1					3		133
3				115		11			4		130
4					71						71
5						75					75
6							81		84		165
7			1					136			137
8									24		24
9				1						175	176
	400	98	129	117	71	86	81	136	115	175	1408

Carpenter-Grossberg, campioni 40, file: art4.xar

	0	1	2	3	4	5	6	7	8	9	
0	400										400
1		98		1							99
2			128								128
3				115		79			4		198
4					71						71
5						7					7
6							81		11		92
7			1					136			137
8									100		100
9				1						175	176
	400	98	129	117	71	86	81	136	115	175	1408

Kohonen, campioni 10, file: koho1.xar

	0	1	2	3	4	5	6	7	8	9	
0	400			1					13		414
1		98		1							99
2			129								129
3				115		1					116
4					71		4				75
5						85					85
6							75				75
7								136			136
8							2		102		104
9										175	175
	400	98	129	117	71	86	81	136	115	175	1408

Kohonen, campioni 20, file: koho2.xar

	0	1	2	3	4	5	6	7	8	9	
0	368										368
1		98									98
2			127	1							128
3				115							115
4					71						71
5						86					86
6							81				81
7			2					136			138
8	32								115		147
9				1						175	176
	400	98	129	117	71	86	81	136	115	175	1408

Kohonen, campioni 40, file: koho4.xar

	0	1	2	3	4	5	6	7	8	9	
0	399										399
1		98									98
2			128	1							129
3				115							115
4					71						71
5						86					86
6							81				81
7			1					136			137
8	1								115		116
9				1						175	176
	400	98	129	117	71	86	81	136	115	175	1408

Kohonen, campioni 60, file: koho6.xar

	0	1	2	3	4	5	6	7	8	9	
0	400										400
1		98		1							99
2			128	1							129
3				115							115
4					71						71
5						86					86
6							81				81
7			1					136			137
8									115		115
9										175	175
	400	98	129	117	71	86	81	136	115	175	1408

Kohonen, campioni 80, file: koho8.xar

	0	1	2	3	4	5	6	7	8	9	
0	400			1							401
1		98		1							99
2			128								128
3				115							115
4					71						71
5						86					86
6							81				81
7			1					136			137
8									115		115
9										175	175
	400	98	129	117	71	86	81	136	115	175	1408

Kohonen, campioni 100, file: koho10.xar

	0	1	2	3	4	5	6	7	8	9	
0	400			1							401
1		98		1							99
2			128								128
3				115							115
4					71						71
5						86					86
6							81				81
7								136			136
8									115		115
9			1							175	176
	400	98	129	117	71	86	81	136	115	175	1408

Back-Propagation, epoca 70, campioni 700, file: b70.xar

	0	1	2	3	4	5	6	7	8	9	
0	385										385
1		98	1	1							100
2	7		128	1							136
3				114							114
4					41						41
5				1		86					87
6					29		81				110
7								136			136
8	8				1				115		124
9										175	175
	400	98	129	117	71	86	81	136	115	175	1408

Back-Propagation, epoca 80, campioni 800, file: b80.xar

	0	1	2	3	4	5	6	7	8	9	
0	399										399
1		98	1	1							100
2			128	1							129
3				115							115
4					71						71
5						86					86
6							81				81
7								136			136
8									115		116
9										175	175
	400	98	129	117	71	86	81	136	115	175	1408

Back-Propagation, epoca 100, campioni 1000, file: b100.xar tempo: 165s

	0	1	2	3	4	5	6	7	8	9	
0	400			1							401
1		98	1	1							100
2			128								128
3				115							115
4					71						71
5						86					86
6							81				81
7								136			136
8									115		115
9			1							175	176
	400	98	129	117	71	86	81	136	115	175	1408

Appendice-C

Back-Propagation, file: buoni1.xar senza rumore

	0	1	2	3	4	5	6	7	8	9	
0	400										399
1		98									98
2			129								129
3				116							114
4					71						71
5						86					88
6							81				81
7								136			136
8									115		115
9										175	116
	400	98	129	116	71	86	81	136	115	175	1407

Back-Propagation, file: buoni1.xar rumore 10%

	0	1	2	3	4	5	6	7	8	9	
0	399										399
1		98									98
2			129								129
3				113		1					114
4					71						71
5				3		85					88
6							81				81
7								136			136
8									115		115
9	1									175	116
	400	98	129	116	71	86	81	136	115	175	1407

5 Errori su 1407

Perceptron file: slp2.xar senza rumore

	0	1	2	3	4	5	6	7	8	9	
0	400										400
1		98									98
2			129								129
3				116							116
4					71						71
5						86					86
6							81				81
7								136			136
8									115		115
9										175	116
	400	98	129	116	71	86	81	136	115	175	1407

Perceptron file: slp2.xar 10% di rumore

	0	1	2	3	4	5	6	7	8	9	
0	390		1								391
1		98									98
2			127					2			129
3				114					1		115
4					71						71
5	2			2		86			1		91
6	1						80				81
7			1					134		1	136
8	3						1		113	1	118
9	4									173	177
	400	98	129	116	71	86	81	136	115	175	1407

21 Errori su 1407

Kohonen, file ko10000.xar senza rumore

	0	1	2	3	4	5	6	7	8	9	
0	400										400
1		98									98
2			129								129
3				116							116
4					71						71
5						86					86
6							81				81
7								136			136
8									115		115
9										175	116
	400	98	129	116	71	86	81	136	115	175	1407

Kohonen, file ko10000.xar 10% di rumore

	0	1	2	3	4	5	6	7	8	9	
0	398										398
1		98									98
2			129								129
3				116							116
4					71						71
5						86					86
6							81				81
7								136			136
8	2								115		117
9										175	175
	400	98	129	116	71	86	81	136	115	175	1407

2 Errori su 1407

Bibliografia

[1] J. J. Hopfield, *Neural Networks and Physical Systems with Collective Computational Abilities.* Proceeding of the National Academy of Sciences, Vol. 79, pp. 2554-2558, Apr. 1982

[2] J. J. Hopfield, Neurons with Graded Response Have Collective Computational Abilities, Proceeding of the National Academy of Sciences, Vol. 81, pp. 3088-3092, Apr. 1984

[3] J. J. Hopfield, D. W. Tank,*"Neural" computation of decisions in Optimization Problems*, Biological Cybernetics, Vol. 52, pp 141-152, 1985

[4] J. J. Hopfield, *Pattern recognition computation using action potential timing for stimulus representation*, Nature, Vol. 376, 6 July 1995, pp. 33-36.

[5] J. J. Hopfield (2007), Scholarpedia, 2(5):1977.

[6] John Hertz, Anders Krogh, Richard G. Palmer, *Introduction to the theory of neural computation*, 1991, Addison-Wesley, Redwood City CA.

[7] Richard P. Lippmann, *An Introduction to Computing with Neural Nets*, IEEE ASSP, April 1987.

[8] Luigi Capra, *Segmentazione di caratteri e riconoscimento mediante reti neuronali*, tesi di laurea in Scienze dell'Informazione, Università degli Studi di Torino, anno accademico 1991/92.

[9] Silvio Cammarata, *Reti neuronali: una introduzione all'"altra" intelligenza artificiale*, ETAS libri, Milano, 1990, pag. 173.

[10] Robert A. Jacobs, *Increased Rates of Convergence Through Learning Rate Adaptation*, Neural Networks, Vol. 1, pp. 295-307, Pergamon Press.

[11] D. E. Rumelhart, G. E. McClelland, *Parallel Data Processing: Exploration in Microstructure of Cognition*, Cambridge (USA), MIT Press, 1986 (Vol. 1 e 2).

[12] D. E. Rumelhart, G. E. McClelland, *Exploration in Parallel Distributed Processing*, documento di lavoro del gruppo PDP, 1987.

[13] V. Sigillito, *Associative Memories and Feedforward Networks: A Synopsis of Neural-Network, Research at the Milton S. Eisenhower Research Center*, Johns Hopkins APL Technical Digest, vol. 10, no. 3, pp. 254—261, 1989.

[14] Gregory A. Piñero, *Hopfield Networks: a simple OCR application*, Neural-Networks 605.455 (Lab3), April-May 2006.

[15] Abderrahmane Namane, PatricK Meyrueis, *Multiple classifier for degraded printed characters recognition*, Actes du dixième Colloque International Francophone sur l'Ecrit et le document.

[16] Arun K. Pujari, C. Dhanunjaya Naidu, B. C. Jinaga, *An adaptive Character Recognizer for Telugu Scripts using Multiresolution Analysis and Associative memory*.

[17] P. Marincola, *Prospettive e applicazioni delle reti neurali nei sistemi per il riconoscimento di testi.* Rapporto interno Stelit S.p.A. Marzo 1991, Roma.

[18] Hasan M. Al Shalabi, Mowafak F. Hasan and Abbas M. Ali, A New Data base Scheme Arabic Handwriting Recognition by Hopfield Neural Networks Algorithm, Journal of Computer Science 1 (2): 204-206, 2005.

[19] Raul Rojas, *Neural Networks - A Systematic Introduction*, Springer-Verlag, Berlin, New-York, 1996.

[20] Amit, D., H. Gutfreund, and H. Sompolinsky (1985), "*Storing Infinite Numbers of Patterns in a Spin-Glass Model of Neural Networks*", Physical Review Letters, Vol. 55, No. 14, pp. 1530-1533.

[21] Marvin Minsky, Seymour Papert, *Perceptrons*, Cambridge (USA), MIT Press, 1969.

[22] Marvin Minsky, Seymour Papert, *Perceptrons; Expandend Edition*, Cambridge (USA), MIT Press, 1988.

[23] Andrea Baraldi, Ethem Alpaydin, *Constructive Feedforward ART Clustering Networks – Part I e Part II*, IEEE Transaction on Neural Networks, Vol. 13, No. 3, May 2002, pp. 645-677.

[24] Thomas G. Dietterich and Ryszard S. Michalski, *A Comparative Review of Selected Methods for Learning from Examples*, in R.S Michalski, J.G. Carbonell, T.M. Mitchell, *Machine Learning. An Artificial Intelligence Approach*, Springer-Verlag Berlin, 1984, cap. 3, pp. 41-82.

[25] Bart Kosko, Neural Networks for Signal Processing, Prentice Hall, Englewood Cliff, NJ, 1992.

[26] Statish Kumar, Sanjay Saini and Prem Prakash, *Alien Attractors and Memory Annihilation of Structured Sets in Hopfield Networks*, IEEE Transaction on Neural Networks, Vol. 7, No. 5, September 1996, pp. 1305-1308.