1. **Bepaal gewichten**

$$w_{ij} = \sum_{s=0}^{M-1} p_i^s p_j^s \,,\; i \neq j \,,\; 0 \leq i \leq N-1 \,,\; 0 \leq j \leq N-1$$

waarin M het aantal patronen dat geleerd wordt is, N het aantal neuronen, dat gelijk is aan het aantal patroon elementen, en p_i^s, p_j^s element i, j van patroon s is.

2. **Initialiseer de activatie met het onbekende patroon**

$$s_i(0) = x_i \,,\; 0 \leq i \leq N-1$$

waarin x_i element i van het input patroon is.

3. **Itereer totdat de activatie convergeert**

$$s_i(t+1) = f_h\left(\sum_{j=0}^{N-1} w_{ij} s_j(t)\right) \,,\; 0 \leq i \leq N-1$$

Hierin is f_h de 'hard-limiting threshold' functie .

Table 3: Het Hopfield Netwerk Algorithme

VERVORMDE TEXTSTRINGS HERKENNEN MET EEN SPATIO-TEMPOREEL PATROON HERKENNINGS (SPR) NEURAAL NETWERK

Ir. P.P. Meiler
Fysisch en Elektronisch Laboratorium TNO
Parallel Processing Groep
Postbus 96864
2509 JG 's-Gravenhage
Fax: 070-280961
Telefoon: 070-264221

Het FEL heeft een SPR neuraal netwerk geïmplementeerd om vervormde woorden en/of delen van zinnen, in dit geval de namen van schepen, te herkennen in een real-time toepassing. Deze namen kunnen vervormd worden door transmissie en/of type fouten. SPR legt de nadruk op de volgorde van de karakters in een woord. SPR is bestand tegen weggevallen en extra toegevoegde karakters en tegen verwisselde karakterparen. Er is ook een SPR leerstrategie ontwikkeld.

Trefwoorden: Fine-grain parallelism, Fout-tolerantie, Lerende systemen, Neurale Netwerken, Parallel Distributed Processing, Realtime applicaties, SPR netwerken, Woord herkenning, Woord reconstructie.

Categorie: Nieuwe technieken

INLEIDING

Het Fysisch en Elektronisch Laboratorium (FEL) is één van de drie instituten van de TNO hoofdgroep defensie onderzoek (HDO). Het FEL houdt zich vooral bezig met observatie, communicatie, informatie-technologie en operationele research. Het FEL verricht niet alleen opdrachten voor defensie, maar ook voor de industrie en andere research instituten.

Het werk op het gebied van de informatietechnologie wordt gedaan in de divisie Systeem ontwikkeling en Informatietechnologie. Veel van het onderzoek op het FEL heeft te maken met sensoren (Radar, Sonar, Infrarood, etc.). Een kenmerkende eigenschap van deze sensoren is dat ze zeer veel data produceren. Moderne sensoren produceren zoveel data dat het zowel voor mensen als voor conventionele computersystemen onmogelijk is om al deze informatie real-time te verwerken. Het FEL houdt zich bezig met onderzoek en ontwikkelling van computer systemen en algoritmen die in staat zijn om juist de interessante gegevens uit deze data te halen.

Het verwerken van de sensor data vraagt zeer veel rekenkracht. Daarom maakt het FEL gebruik van parallel processing technieken. Neurale netwerken vallen onder deze categorie. De ontwikkelingen op het gebied van neurale netwerken gaan zeer snel. Er zijn echter nog niet veel direct toepasbare applicaties op basis van neurale netwerken ontwikkeld. Het FEL is juist geïnteresseerd in het toepasbaar maken van de neurale netwerk technologie. Het FEL wil neurale netwerken gebruiken voor patroonherkenning, detectie en classificatie. Voorbeelden zijn radar puls clustering, tekst herkenning en doelclassificatie m.b.v. radar. Toekomstige aandachtsgebieden zijn herkenning van beweging in beelden, kwaliteits controle systemen en luchtverkeers begeleiding. Het werk van het FEL op het gebied van tekst herkenning met neurale netwerken is het onderwerp van dit paper.

De schepen van een eenheid van de Koninklijke Marine sturen veel berichten naar elkaar. De berichten zijn gecodeerd in een binaire code (b.v. ASCII). Alle berichten worden verstuurd over één enkel radio kanaal. Elk schip kan alle berichten van elk van de andere schepen ontvangen. Ergens in een bericht is een adres te vinden (de naam van het schip, van de kapitein, etc.). Dit adres bepaalt voor welk schip het bericht bestemd is. In een stress situatie nemen zowel de spanning van de bemanning als het aantal berichten toe. Er kunnen dan fouten gemaakt worden bij het verzenden en bij het herkennen van de berichten.

Het FEL heeft een opdracht ontvangen van de K.M. om een automatisch berichten herkennings systeem te ontwikkelen. Het systeem is, als onderdeel van de research op het gebied van parallel processing, als neuraal netwerk geïmplementeerd. De SPR (Spatio-temporal Pattern Recognition) aanpak is gekozen omdat SPR goed aansluit bij het feit dat een woord bestaat uit een reeks van karakters en omdat SPR goed geïmplementeerd kan worden op parallel processing hardware (b.v. transputers).

ARCHITECTUUR EN IMPLEMENTATIE VAN HET SPR NETWERK

De SPR architectuur en het gebruik ervan om het berichtenherkennings probleem op te lossen worden eerst behandeld. Daarna wordt de implementatie van het systeem in software en in hardware besproken.

De architectuur van het SPR netwerk

SPR is ontworpen om tijdvolgorde relaties in een reeks input-vectoren te herkennen, zie figuur 1. SPR berekent de overeenkomst (match) tussen een reeks input-vectoren en een aantal reeksen vectoren die opgeslagen zijn in de lagen van het netwerk. Elke laag van het netwerk bestaat uit een aantal aan elkaar gekoppelde Processor Elementen (PE's) en kan beschouwd worden als een filter dat afgestemd is op een bepaalde reeks input-vectoren.

Een PE berekent de match tussen een input-vector en de vector die in de PE is opgeslagen. Een PE heeft één ingang en één gewicht voor elk element van een vector. Een PE slaat een vector met lengte N op als de waarden van de N gewichten van die PE. De vergelijkingen (processing equations) die het gedrag van een PE bepalen zijn hieronder gegeven, zie ook figuur 2. Deze berekening wordt in elke PE van het netwerk uitgevoerd.

$$X_{new} = X_{old} + Af(-a.X_{old} + b.Tsi + c.Tf(\underline{Q}.\underline{W}))$$

Variabelen:
X_{new} = Nieuwe output van PE (op tijdstip t+1)
X_{old} = Oude output van PE (op tijdstip t)
Tsi = Time Sequence Input (output vorige PE, zelfde laag, op tijdstip t)
$\underline{Q}$ = Genormaliseerde input-vector
$\underline{W}$ = Genormaliseerde gewicht-vector

Parameters:

Av	= Attack waarde	$Av \geq 1$
Tv	= Treshold waarde	$0 \leq Tv \leq 1$
a	= Verval constante	$0 \leq a \leq 1$
b	= Tsi versterking	$0 \leq b \leq 1$
c	= Input-vector versterking	$0 \leq c \leq 1$

Functies:

Af(x)	= Av.x	Als x > 0	De Attack functie
	= x	Als x ≤ 0	
Tf(x)	= x	Als x ≥ Tv	De Treshold functie
	= 0	Als x < Tv	

Input-vector Q wordt tegelijkertijd aangeboden aan alle PE's in het netwerk. Elke PE heeft één Tsi-input, afkomstig van de voorafgaande PE in dezelfde laag. Het aantal verbindingen in het netwerk is O(N), N = aantal PE's.

Als een PE wordt geactiveerd met een constante waarde I (I = combinatie van Tsi en $\underline{Q}.\underline{W}$) zal de maximum output bereikt worden met een tijdconstante Top = 1/(a.Av). Als I = 0 zal de output 0 worden

met een tijdconstante Tneer = 1/a. Als Av > 1 is Top < Tneer, zie figuur 3. Tf(x) zorgt ervoor dat de PE niet reageert als $\underline{Q}.\underline{W} < Tv$, d.w.z. als $\underline{Q}$ en $\underline{W}$ niet genoeg op elkaar lijken.

De input-vectoren $\underline{Q}$ van een reeks worden één voor één aan het netwerk toegevoerd. De waarde van een Tsi output in een laag geeft de overeenkomst aan tussen de reeks waarop die laag is afgestemd en de reeks die aan het netwerk wordt aangeboden. De gevoeligheid van een PE voor een input-vector wordt bepaald door de Tsi van de voorafgaande PE.

Voorbeeld (figuur 4): Een laag is afgestemd op de reeks vectoren (A, B, C). De vectoren A, B en C worden aangeboden op tijdstippen T=1, T=2 en T=3. Op T=1 wordt PE_A door vector A geactiveerd. Op T=2 wordt PE_B door vector B geactiveerd. De output van PE_B op T=2 is groter dan die van PE_A op T=1 omdat PE_B extra wordt geactiveerd door de Tsi-input van PE_A. Om dezelfde reden is de output van PE_C op T=3 nog weer groter. Er is sprake van een opslinger effect (hoe meer iteraties, hoe groter de output). De output van een PE wordt kleiner als er geen input meer is (zie PE_A op T=1, T=2, T=3).

De output van de laatste PE van een laag wordt de Match Value (MV) genoemd en is een maat voor de match tussen de input-reeks en de reeks waarop die laag is afgestemd. Een comparator selecteert de laag met de grootste MV of geeft een No-Match signaal als voor elke laag geldt dat MV < MTrh (Match Treshold). Als er een vector van een input reeks wegvalt of als er een extra vector wordt tussengevoegd zal het opslinger effect nog steeds optreden, hoewel de MV dan kleiner zal zijn.

Text string herkenning met een SPR netwerk

Een letter wordt gerepresenteerd door een vector (b.v. als ASCII code). Een string van N letters wordt gerepresenteerd door een reeks van N vectoren. De gewicht-vector van een PE is gelijk aan de vector representatie van de letter die door die PE moet worden herkend.

De match tussen input-vector $\underline{Q}$ en gewicht-vector $\underline{W}$ wordt berekend als het inprodukt $\underline{Q}.\underline{W} = |\underline{Q}|.|\underline{W}|.\cos(\theta)$. Omdat alle vectoren evenveel invloed moeten hebben in deze berekening worden $\underline{Q}$ en $\underline{W}$ genormaliseerd zodat $|\underline{Q}|=|\underline{W}|=1$. Het normaliseren wordt gedaan door de input laag van het netwerk.

Een extra Tsi-input

Om het netwerk minder gevoelig te maken voor twee verwisselde karakters kan een extra Tsi-input toegevoegd worden. Stel een laag heeft N PE's. PE[N] heeft dan een Tsi_1 afkomstig van PE[N-1] en een Tsi_2 afkomstig van PE[N-2]. De processing equation ziet er dan zó uit:

$X_{new} = X_{old} + Af(-a.X_{old} + b_1.Tsi_1 + b_2.Tsi_2 + c.Tf(\underline{Q}.\underline{W}))$

Stel dat een laag is ingesteld om de reeks 'ABCDEF' te herkennen en dat de reeks 'ACBDEF' wordt aangeboden. PE_D zal dan door PE_B (via Tsi_2) én door PE_C (via Tsi_1) worden geactiveerd. Op deze manier is de volgorde van de input-vectoren 'B' en 'C' minder belangrijk voor de activatie van PE_D.

Normalisatie van de Match Value (MV) van een laag.

In een standaard SPR netwerk heeft elke laag hetzelfde aantal PE's. Text strings hebben echter een willekeurige lengte. Als een laag P PE's heeft en de door die laag te herkennen text string (= reeks vectoren) heeft K karakters, worden de gewichten van de eerste (P-K) PE's van die laag op nul gezet.

Stel laag L3 herkent een string van 3 karakters en laag L6 herkent een string van 6 karakters. Bij een perfecte matchende input zal de MV van L6 groter zijn dan de MV van L3, omdat L6 6 iteraties heeft gehad om op te slingeren en L3 maar 3. De MV's zijn dus verschillend, hoewel de inputs gelijkwaardig zijn. Dit kan opgelost worden door de MV's van elke laag te normaliseren. De Maximum MV (MMV_n) van een laag met lengte N kan worden berekend. Een Output PE (OPE) wordt aan elke laag toegevoegd. Deze OPE vermenigvuldigt de output van de laatste PE van een laag met een factor $1/MVV_n$. Op deze manier is de output van de OPE altijd maximaal 1 voor een perfecte input reeks, onafhankelijk van de lengte van de reeks.

Onthouden van de maximum PE output waarde

Als er een bepaalde vaste PE moet worden gekozen om een indicatie te geven over de match tussen een input-reeks en de te herkennen reeks, is de laatste PE van een laag de beste keus. Soms geeft een andere PE echter een betere indicatie. Voorbeeld: Een laag herkent de reeks R='HARLINGEN'. Er zijn twee inputs: A='HARLI####' en B='#####NGEN'. A zou beter moeten scoren dan B, want A heeft 5 karakters gemeenschappelijk met R, en B maar 4. Als we naar de output van de laatste PE van de laag kijken scoort B echter beter dan A.

Als A wordt aangeboden gebeurt het volgende: De eerste 5 PE's laten een toenemende activatie zien, want de eerste 5 karakters van A matchen met R. Stel MO_A is de Maximale Output waarde (die van PE_5). De laatste 4 PE's laten een afnemende activatie zien, want de laatste 4 karakters van A matchen niet met R. Stel FO_A is de uiteindelijk bereikte output waarde (i.e. die van PE_9). Bij B gebeurt het volgende: De eerste 5 PE's reageren niet, want de eerste 5 karakters van B matchen niet met R. De laatste 4 PE's laten een toenemende activatie zien, want de laatste 4 karakters van B matchen met R. Stel FO_B is de uiteindelijk bereikte output waarde (i.e. die van PE_9). In dit geval is $MO_B=FO_B$. Zie figuur 5. In deze figuur is $MO_A=2.3$, $FO_A=1.0$, $MO_B=1.9$ en $FO_B=1.9$.

Het resultaat is dat $FO_B>FO_A$ (hoewel $FO_B<MO_A$), zodat B wint. De oplossing is: één Maximum PE (MPE) toevoegen aan iedere laag. Deze MPE is verbonden met de outputs van alle PE's in die laag en onthoudt de hoogste PE output waarde die ooit in die laag is voorgekomen, zie figuur 6. We gebruiken nu de output van de MPE i.p.v. de output van de laatste PE. Dan geldt dat $MPE_A=MO_A$, $MPE_B=MO_B$ en $MO_A>MO_B$. Dan is $MPE_A>MPE_B$ en dus wint input A.

Implementatie als een software simulatie

Er is een programma gemaakt dat een SPR netwerk zoals beschreven in dit paper (inclusief de uitbreidingen) simuleert. Het is geschreven in Turbo Pascal versie 5.0 en draait op een PC-AT, optioneel met 80287 coprocessor. Het programma bestaat uit ± 4000 regels source code. Het programma biedt een interactieve test en ontwikkel omgeving en kan een compleet overzicht geven van wat er in het netwerk gebeurt (zowel numeriek als grafisch).

Het programma kan twee namen per seconde verwerken als het een netwerk simuleert dat 11 namen, elk bestaande uit 12 karakters, kan herkennen. Een goede manier om de performance te verbeteren is om het programma te koppelen aan de ANZA+ neural network simulator van HNC. Deze simulator kan 6 miljoen verbindingen per seconde simuleren, tegen O(10⁴) voor een PC-AT.

Implementatie met een parallel processing systeem

Omdat er geen communicatie is tussen de lagen onderling kunnen één of meerdere lagen per processor geïmplementeerd worden zonder dat er communicatie tussen de processoren onderling nodig is. Er is geen interconnectie-bottleneck (zoals bij een fully interconnected netwerk) omdat het aantal verbindingen O(N) is en niet O(N²), met N=het aantal PE's in het netwerk. De SPR architectuur is scalable, i.e. de snelheid van het systeem neemt lineair toe met het aantal processoren. Wij denken eraan het netwerk te implementeren op een transputer systeem [INMOS, 1988].

HET INSTELLEN VAN DE GEWICHT-VECTOREN

De gewicht-vectoren kunnen off-line berekend worden. In het geval van textstring herkenning is de gewicht-vector van een PE gelijk aan de vector representatie van het karakter dat door die PE moet worden herkend. Stel laag N heeft P PE's en herkent de naam 'ALKMAAR'. PE's [N,1] t/m PE[N,P-7] doen niet mee. PE[N,P-6] herkent 'A', PE[N-5] herkent 'L', etc.

Er kan ook gebruik gemaakt worden van een supervised leer algoritme. Alleen die gedeeltes van een text die de naam van een schip representeren worden gebruikt om het netwerk te trainen. Het algoritme gaat uit van een netwerk met M lagen en R te herkennen namen, met R≤M. Het algoritme wijst één of meer lagen toe aan elke naam. Als een naam vaak een bepaalde afwijking vertoont, kan er een extra laag voor die naam, inclusief de fout, worden toegewezen. Op deze manier kunnen er L lagen gebruikt worden voor één naam. Een post-processor voegt de uitgangen van deze L lagen samen tot één uitgang. Als R>M worden alleen de M meest voorkomende namen opgeslagen.

Het eigenlijke leren gaat als volgt: Als de MV van de winnende laag groter is dan MTrh (een Treshold waarde) worden de gewicht-vectoren van die laag aangepast. Dit aanpassen gebeurt voor iedere gewicht-vector door er een fractie van de input-vector bij op te tellen en het resultaat te normaliseren. Als MV<MTrh wordt aangenomen dat de input reeks nog niet is opgeslagen en wordt er een nieuwe laag toegewezen aan deze reeks. Als alle lagen al in gebruik zijn wordt die laag, die het minst vaak gewonnen heeft, overschreven door de nieuwe reeks. Dit houdt in dat de activatie-historie van elke laag bijgehouden moet worden.

Dit algoritme voorkomt het informatie-overflow probleem. Dit probleem treedt bijvoorbeeld op in Hopfield netwerken als een netwerk meer dan 0.15 N patronen op moet slaan (met N = het aantal PE's in het netwerk). De performance van het Hopfield netwerk degradeert dan snel. Het SPR leer algoritme lijkt wat dit betreft op Grossberg's ART algoritme [Grossberg, 1988].

RESULTATEN

Twee metingen aan een SPR netwerk, ingesteld voor het herkennen van 11 namen van 12 karakters, worden besproken. De gebruikte netwerk parameters zijn:

Av=1.3 Tv=0.5 a=0.8
b1=0.4 b2=0.3 c=1.0

Beide metingen bestaan uit 21 meetpunten. Een meetpunt wordt berekend door elk van de 11 namen 5 maal door het netwerk te laten verwerken. Het netwerk moet dus 1155 namen verwerken voor één meting.

Random karakter vervorming

Bij deze meting heeft ieder karakter van een input string een kans P om te worden vervormd (dwz. vervangen door een random karakter). P gaat van 0% tot 100% in stappen van 5%. Voorbeeld: 'ABCDEFGHIJ', P=50% -> 'AXJDPFGAIM'. De score voor een meetpunt met vervormings-percentage P is het percentage van de 55 namen dat correct herkend is. Uit de test blijkt dat het netwerk nog 95% van de namen herkent als P=30% en dat de performance geleidelijk afneemt, zie figuur 7. Door de beperkte sample grootte (11 namen) is er bij P=100% toch nog een kans van ±9% dat de correcte naam wordt aangewezen. De kans om een verkeerde naam uit te kiezen wordt groter als het aantal namen om uit te kiezen toeneemt. Met gebruik van een treshold kan ook een No-Match signaal gegenereerd worden.

Random karakter-paar verwisseling

Bij deze meting worden, met een kans P, de karakters van een karakterpaar verwisseld. String 'ABCD' bestaat uit de paren 'AB', 'BC' en 'CD'. P gaat van 0% tot 100% in stappen van 5%. Voorbeeld: 'ABCDEFGHIJK', P=20 -> 'ACBDEFGIHJK'. Als de score wordt berekend zoals bij de random karakter vervorming is deze altijd groter dan 98%, zelfs bij P=100%. Dit komt omdat het netwerk inderdaad goed bestand is tegen karakter verwisseling, maar ook door de beperkte sample grootte.

Daarom wordt een andere meetmethode gebruikt. Stel AVB=Activation Value van de Best matchende laag en AVS=Activation Value van de Second-best matchende laag. De score is het relatieve verschil

tussen AVB en AVS: score = (AVB-AVS)/AVB. De uiteindelijke score voor een meetpunt is het gemiddelde van de scores voor elk van de 55 verwerkte namen. De meting laat zien dat het netwerk goed onderscheid maakt tussen de Best en Second-best matchende lagen. De score neemt geleidelijk af van 0.65 tot 0.45.

CONCLUSIES

Het SPR netwerk kan namen herkennen waarvan één of meer van de karakters verkeerd zijn, als er karakters weggevallen of tussengevoegd zijn, of als er karakters worden verwisseld.

Het leer algoritme is eenvoudig. Information overflow kan niet optreden. Als de capaciteit van het netwerk te klein is om alle namen te leren, worden alleen de meest voorkomende namen geleerd.

Een SPR netwerk voor het naamherkennings probleem is geïmplementeerd. Het SPR programma draait op een IBM PC-AT of compatible.

Een SPR netwerk is geschikt voor implementatie op een parallel processing systeem. Omdat er geen communicatie is tussen de lagen van het netwerk, neemt de snelheid van het systeem lineair toe met het aantal processoren.

REFERENTIES

[INMOS, 1988] INMOS, "The transputer family", INMOS limited, Bristol, UK

[Grossberg, 1988] Grossberg, S., "Nonlinear Neural Networks: Principles, Mechanisms, and Architectures", in: Neural Networks, Volume 1, Number 1, pp. 17-61, Pergamon Press, 1988

LITERATUUR

[1] HNC Incorporated, "ANZA Plus User's Guide and Neurosoftware documents", Release 2.1, San Diego, California, 1988

[2] Houtekamer, G.E. & Reijns, G.L., "Computer Performance", (L96), Computer Architecture Laboratory, Faculty of Electrical Engineering, Delft University of Technology, 1986

[3] INNS, "Abstracts of the first annual INNS meeting, Boston, 1988", Neural Networks, Volume 1, Supplement 1, Pergamon Press, 1988

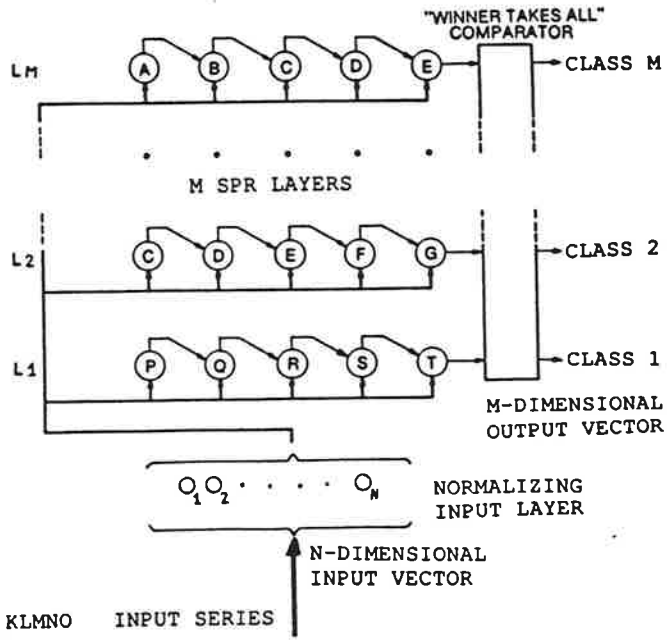
[4] Kohonen, T., "Self-organization and Associative memory", Springer-Verlag, Berlin, 1984

[5] Lippmann, R.P., "An Introduction to Computing with Neural Nets", IEEE ASSP Magazine, April 1987

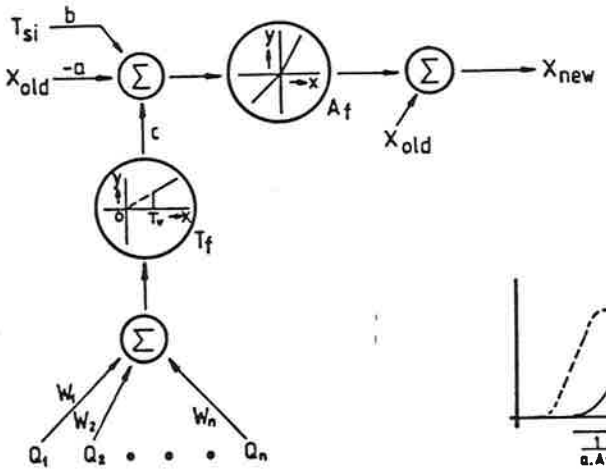
[6] McClelland, J.L. & Rumelhart, D.E., "Parallel distributed processing", Vol. I, II and III, Cambridge MA., MIT press, 1986

[7] Rosenberg, C.R. & Sejnowski, T., "NETtalk: A Parallel Network That Learns to Read Aloud", John Hopkins Univ., Technical Report JHU/EECS-86/01, 1986

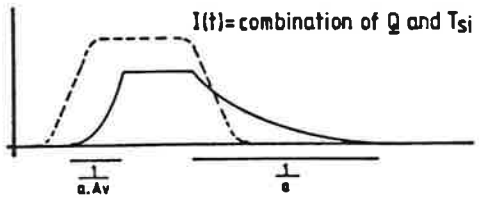
[8] Soucek, B. & Soucek, M., "Neural and Massively Parallel Computers", John Wiley & Sons, Inc, 1988



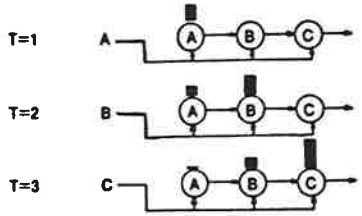
Figuur 1 "Architectuur van een SPR neuraal netwerk"



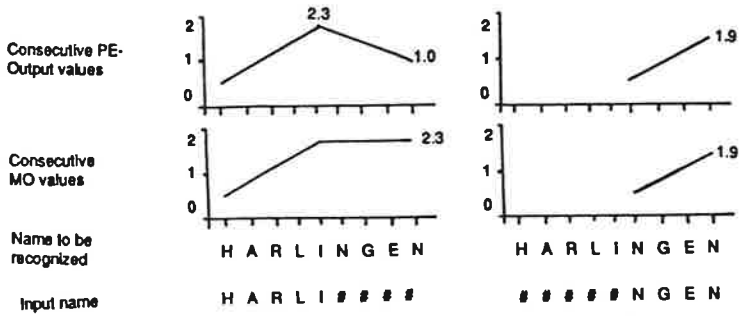
Figuur 2 "Structuur van een PE"



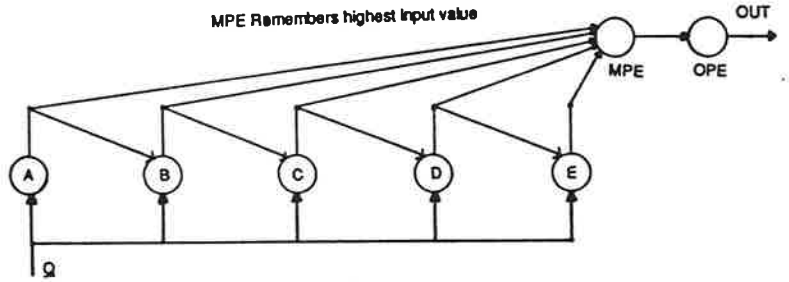
Figuur 3 "PE reactie op input"



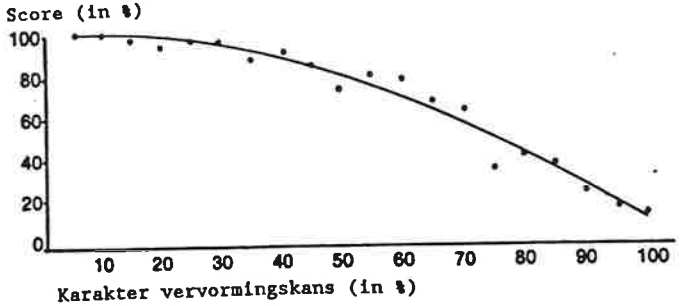
Figuur 4 "SPR string matching"



Figuur 5 "PE outputs en MO waarde tijdens string matching"



Figuur 6 "Toevoeging van een MPE"



Figuur 7 "Random karakter vervorming"