**IT’S HARD FOR NEURAL NETWORKS TO LEARN THE**

**GAME OF LIFE**

**Review**

In acest articol, autorii au incercat, folosind o retea neuronala relativ mica, sa invate Game of Life, prezicand n pasi in viitor. Pornind de la o retea minimala cu 2n+1 layere si 23n + 2 parametri antrenabili, ce in teorie ar trebui sa fie suficienta, articolul ne arata contrarul.

**Studii anterioare**

Studii recente au aratat ca retelele neuronale pot fi “curatate” renuntandu-se la ponderile inutile, mentinand performanta asemanatoare cu cea initiala. Astfel, apare conceptul de “lottery ticket”, adica o subretea care, antrenata izolat, poate atinge perfomanta retelei originale.

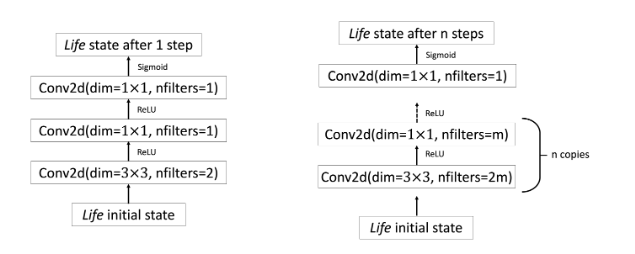
S-a dovedit ca retelele minimale nu converg catre o solutie decat daca sunt initializate aproape de una.

Cresterea adancimii si a numarului de parametri pot determina imbunatatirea performantei retelei neuronale. De asemenea, prelucrarea ponderilor inainte de antrenare sporeste perfomanta. Astfel, cei trei factori pot determina daca reteaua converge catre o solutie buna sau nu.

**Experimente**

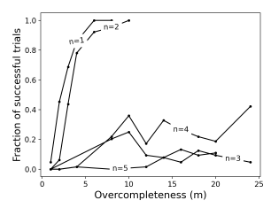
Fiind data o matrice x ce contine valori de 1 sau 0 (celula vie sau moarta), definim G(x) ca urmatorul pas al jocului, conform regulilor mentionate. Definim al n-lea pas ca fiind G^n (x). Regula locala de update evalueaza un grid de 3x3 pentru a determina starea celulei din centru. Un strat convolutional cu 2 filtre 3x3 care influenteaza al doilea strat, cu un filtru de 1x1 , rezolva eficient problema unui pas al Game of Life. Se foloseste functia de activare ReLU pentru a preveni vanishing gradient problem in cazul generalizarii pentru n pasi. Al doilea strat convolutional foloseste functie de activare sigmoid, pentru a decoda outputul.

Generalizand, se obtine reteaua neuronala minimala cu 2n + 1 layere si 23n + 2 ponderi ca in figura urmatoare:



Definim L(n,m) ca fiind o arhitectura a GoL, unde n este numarul de pasi, iar m este numarul de filtre.

Pentru a cuantifica eficacitatea unei arhitecturi de rețea neuronală dată, măsurăm probabilitatea ca o inițializare aleatorie a rețelei sa convearga către o soluție după ce ii sunt aratate un milion de exemple de antrenament. Deoarece L(n, m) poate implementa doar o regulă de actualizare 3 × 3 în fiecare pas de calcul, pentru ca L(n, m) să rezolve problema GoL în n pași, trebuie să învețe regula ei de bază. Astfel, considerăm că o instanță a lui L(n, m) are succes atunci când învață corect regula și, prin urmare, poate prezice G^n (x) cu acuratețe de 100% pentru toate stările inițiale x. Orice exemplu din L care nu are acuratețe perfectă nu a învățat regula de bază și este considerat fără succes. Obiectivul nostru este stabilirea lui P[succesul lui L(n, m) | n, m].



Probabilitatea ca o arhitectura L(n,m) sa invete jocul cu succes. Pentru n > 1, nicio instanta a lui L nu invata cu succes cu arhitectura minimala( m = 1)

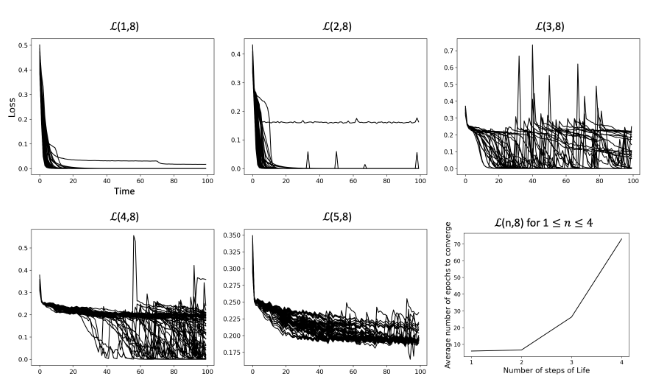
Antrenam 64 de instante ale lui L(n,m) cu 1 ≤ n ≤ 5 si 1 ≤ m ≤ 24. Se omit anumite combinatii din cauza limitelor computationale.

Observăm că din arhitecturile minimale (m = 1), doar instanțele pentru L(1,1) converg spre o soluție, cu o rată de succes de aproximativ 4,7%. Instanțe ale

arhitecturilor pentru n-step-Life cu n ≥ 3 necesită o supracompletitudine mai mare de 24, cel mai înalt grad de supracompletitudine test de catre autorii lucrarii. Acest lucru arata faptul ca valoarea lui m creste odata cu cresterea numarului de pasi n.

Observăm că pentru o supracompletitudine ridicată, arhitecturile pentru n = 4 depășesc n = 3, iar n = 5 performează în mod similar cu n = 3. În timp ce toate cele trei n necesită mult mai mulți parametri decât arhitectura minima pentru a converge constant, ne-am aștepta ca n = 3 necesită mai puțin de n = 4, iar acesta mai puțin de n = 5. Este posibil ca parametrizarea GoL sa aiba un comportament asemanator pentru 3 ≤ n ≤ 5, ceea ce poate face dificila invatarea pentru 3 ≤ n ≤ 5.

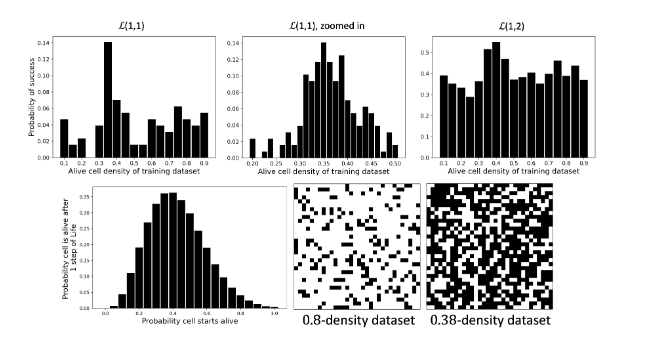
Autorii articolului ne ilustreaza care este momentul convergentei la o solutie pentru 1 ≤ n ≤ 5 si m = 8:



Primele 5 grafice au ca axa verticala loss-ul iar axa orizontala corespunde numarului de epoci pentru antrenare. Ultimul graf indica punctul de convergenta (prima epoca cu loss < 0.01) pentru fiecare L(n,8), reteaua neuronala ajungand la 100% acuratete.

**Setul de date pentru antrenare optim**

Autorii articolului au construit o clasa de date de antrenare numita d-density, un set de 32x32 celule ce au fiecare o probabilitate d de a fi vii. Se antreneaza 128 de instante ale L(1,2) cu densitate 0.1 ≤ d ≤ 0.9 ce creste treptat cu 0.05. De asemenea, se antreneaza L(1,1) pe acelasi set de date, dar cu densitate 0.2 ≤ d ≤ 0.5, ce creste cu 0.0125.



Pentru L(1,1) se observa o crestere semnificativa atunci cand d=0.35 ( 14% rata de success), triplu fata de d=0.3 si dublu fata de d=0.4. Pentru L(1,2), performanta retelei creste cand d=0.4 . Facand media, putem considera densitatea optima ca fiind d0 ≈ 0.38. La aceasta densitate ne putem astepta sa maximizam numarul de celule vii dupa un singur pas al GoL.

În concluzie, constatăm că rețelele arhitecturii L care sunt antrenate să prezică configurația Game of life după n paşi, au nevoie de o configuraţie iniţială ce necesită un grad de supracompletitudine care creste odata cu n pentru a învăța în mod constant regulile jocului. Totodata, initializarile ponderilor sunt sensibile la mici perturbari. În final, se poate constata faptul ca obtinerea solutiei corecte depinde de proprietatile setului de date.