**Evolving Game of Life: Neural Networks, Chaos, and Complexity**

**Review**

Descriem pe scurt componentele existente în „Conway’s Game of Life”: există o grilă/matrice de celule, fiecare este fie „vie” sau „moartă”, fiecare respectă un set de reguli pe masură ce trece timpul si programul avansează.

Setul de reguli pentru fiecare celulă:

* Dacă o celulă „vie” are cel mult 1 sau cel putin 3 vecini „vii”, atunci,

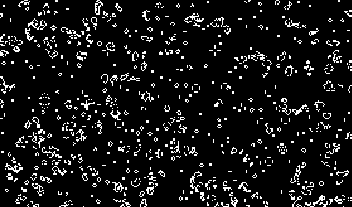
în următorul pas al algoritmului, celula „moare” (pentru singurătate sau suprapopulare).

* Dacă o celulă „moartă” are exact 3 vecini „vii”, în urmatorul pas al

algoritmului, celula va deveni „vie”.

* Altfel, celula își păstrează starea.

În imaginea următoare, este reprezentat un exemplu, dintr-un pas oarecare, dintr-un „Conway’s Game of Life”.

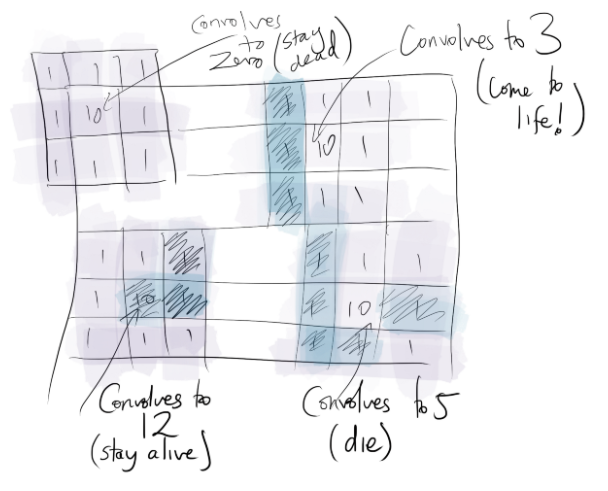


În codul programului (programul pentru demo) există următoarele 2 linii:

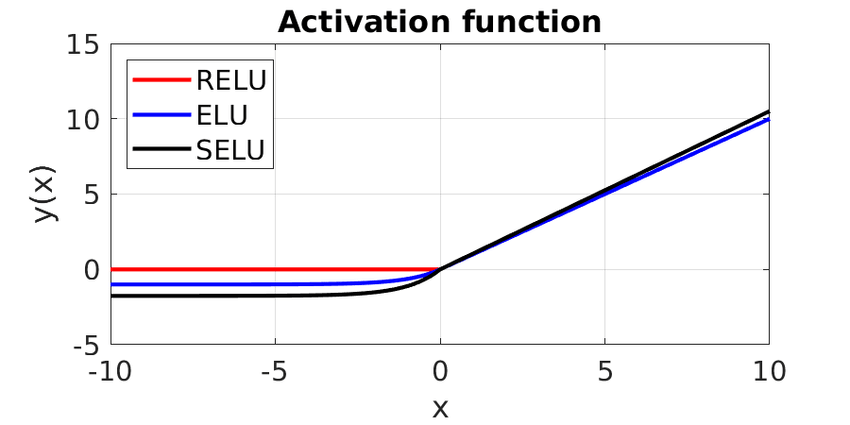


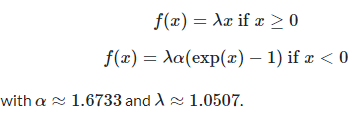
Aceste două linii reprezintă convoluția și neliniaritatea, iar împreună definesc setul de reguli.

Următoarea imagine ilustrează modul de lucru al convoluției (kernel-ul este de 3x3, 1 pentru vecini și 10 în mijloc; celulele albastre înseamnă celule „vii”, cele spre gri înseamnă celule „moarte”):

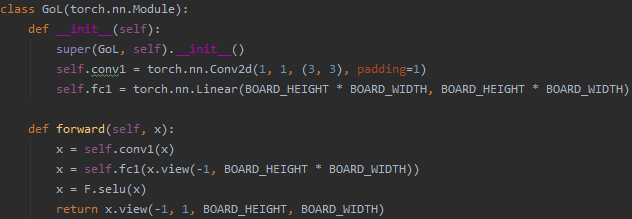


Poate o rețea neurală să învețe Game of Life? Avem date: date de training nelimitate(oricâte jocuri, toate începând de la o configurație inițială random), un kernel de convoluție cu ponderi și o neliniaritate. Autorul codului, pentru a face o coborâre pe gradient eficientă, a modificat neliniaritatea cu un strat/straturi total conectate între ele și funcția de activare SELU, aceasta dând cele mai bune rezultate din punct de vedere anecdotic.





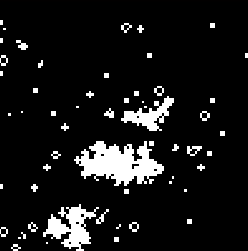
Modelul final este următorul:



A fost folosit Adam pentru optimizare. Datele sunt procesate ca și perechi de tipul (date, date viitoare). Amestecarea datelor din dataloader a avut ca efect prevenirea ca diferite bucăți de date să nu ia un gradient oarecare în direcții random.

Mean squared error a funcționat bine. Cele mai multe celule sunt moarte si loss-ul pentru o prezicere 0 ar fi 0. Autorul codului spune că probabil rețeaua a preferat și încă preferă să prezică celule moarte. Așa că a crescut loss-ul pentru tranzițiile ratate cu un factor constant.

Această modificare a funcționat, iar rezultatul jocului a semănat cu un Game of Life prin aspecte comune: “blocks, beehives, blinkers and toads – even some primitive gliders – or at least approximations thereof”.



Tom Grek, autorul codului, a încercat diferite lucruri pentru a obține rezultate cât mai bune: a folosit 2 straturi liniare ca si neliniaritate unde s-a creat un blocaj între ele (100x100 pixeli, apoi comprimat de la 10.000 la 5.000, apoi a forțat rețeaua să se reconstruiască dintr-o versiune comprimată). Din păcate acest lucru a avut un impact negativ și nu a fost folosit.

În concluzie, rețelele neuronale pot simula complexitatea dar nu o și au, Tom Grek spunând că “ar trebui să fie posibilă combinarea dintre o rețea complexă de agenți inteligenți și un comportament imprevizibil specific oamenilor”. Un exemplu de acest fel ar fi Spiking Neural Network (SNN) care este asemănător cu modul de funcționare al creierului uman în situații critice.