

Rețele neuronale convoluționale, Big Data și Deep Learning în analiza automată de imagini

Mihnea Horia VREJOIU

Institutul Național de Cercetare-Dezvoltare în Informatică – ICI București
B-dul Mareșal Alexandru Averescu, Nr. 8-10, 011455, București, România
mihnea.vrejoiu@ici.ro

Rezumat: În ultimii ani se vorbește din ce în ce mai mult despre inteligența artificială. Ceea ce stă de fapt în spatele inteligenței artificiale astăzi poate fi rezumat pe scurt prin sintagma „rețele neuronale artificiale“, la care mai nou a fost adăugat adjectivul „adânci“. Aplicațiile bazate pe acestea au ajuns să egaleze și chiar să depășească performanța umană în multe domenii. Unul dintre primele domenii în care au fost dezvoltate și în care au căpătat o largă răspândire este cel al vederii artificiale, respectiv al recunoașterii / clasificării de imagini. Fără pretenția de a acoperi complet subiectul, ne propunem în lucrarea de față o trecere în revistă, încercând să surprindem cât mai intuitiv posibil câteva elemente și repere esențiale asupra istoriei și evoluției rețelelor neuronale artificiale, cu noua perspectivă oferită în ultima perioadă de disponibilitatea datelor masive (*Big Data*) utilizate în conjuncție cu acestea, ca factor de importanță majoră complementar, sinergic și convergent alături de calitatea și performanțele algoritmilor *deep learning* implicați. De asemenea, analizăm elementele și mecanismele care definesc și compun rețelele convoluționale în general, funcționarea și specificul acestora cu aplicare în vederea artificială, precum și două dintre primele astfel de arhitecturi de referință, AlexNet și VGGNet, cu particularitățile acestora și tehnici utilizate în procesele de antrenare, validare și testare.

Cuvinte cheie: rețea neuronală (adâncă), rețea (neurală) convoluțională, deep learning, big data, vedere artificială, recunoașterea / clasificarea de imagini.

Convolutional Neural Networks, Big Data and Deep Learning in Automatic Image Analysis

Abstract: In recent years, there is an increasing amount of talk about artificial intelligence. What actually stands behind artificial intelligence today can be briefly summarized by the syntagm „artificial neural networks“, to which the adjective „deep“ has recently been added. Applications based on these have come to equate and even surpass human performance in many areas. One of the first fields in which they have been developed and in which they have gained a wide spread is artificial vision, respectively image recognition / classification. Without claiming to completely cover the subject, in this paper we propose a review, trying to capture as much intuitively as possible some essential elements and milestones of the history and evolution of artificial neural networks, with the new perspective offered in the last period by the availability of massive data (*Big Data*) used in conjunction with them as a major complementary, synergistic and convergent factor along with the quality and performance of the deep learning algorithms involved. Also, we analyze the elements and mechanisms that define and compose the convolutional networks in general, their functioning and their specificity with application in artificial vision, as well as two of the first such reference architectures, AlexNet and VGGNet, with their peculiarities and techniques used in the training, validation and testing processes.

Keywords: (deep) neural network, convolutional (neural) network, deep learning, big data, artificial vision, image recognition / classification.

1. Introducere

În ultimii ani, interesul și investițiile în sisteme cognitive și inteligență artificială au cunoscut o creștere explozivă, care continuă și va continua, atât în mediul academic, de cercetare științifică și dezvoltare tehnologică, precum și în cel comercial și de afaceri, de la cele mai mici până la marile corporații. Cele mai renumite nume din tehnologie (cum sunt: Google, Facebook, Microsoft, Yahoo,

Baidu, IBM, DropBox, etc.) investesc masiv în inteligența artificială pe care o înglobează în propriile modele de afaceri și o utilizează din ce în ce mai mult în oferta lor: asistenți virtuali (de ex. Siri), vedere artificială, recunoașterea vorbirii, traducere automată și o mulțime de alte aplicații. Mai mult, au apărut inițiative precum consorțiul „Partnership on AI“ [49], care grupează ca parteneri fondatori Amazon, Apple, Deepmind, Facebook, Google, IBM, Microsoft, precum și alți aproximativ 80 parteneri din 13 țări. Toate acestea se bazează astăzi practic pe tehnologia oferită de rețelele neuronale artificiale „adânci“ și învățarea profundă (*deep learning*). Unul dintre primele domenii în care acestea au fost dezvoltate și în care au căpătat o largă răspândire este cel al vederii artificiale, respectiv al recunoașterii / clasificării de imagini. Fără pretenția de a acoperi complet subiectul, ne propunem în lucrarea de față o trecere în revistă, încercând să surprindem cât mai intuitiv posibil, câteva elemente și repere esențiale asupra istoriei și evoluției rețelelor neuronale artificiale, cu noua perspectivă oferită în ultima perioadă de disponibilitatea datelor masive (Big Data) utilizate în conjuncție cu acestea, ca factor de importanță majoră complementar, sinergic și convergent alături de calitatea și performanțele algoritmilor deep learning implicați. În secțiunea următoare a lucrării de față ne propunem o trecere în revistă asupra rețelelor neuronale și a evoluției lor de la modelul simplu al perceptronului la rețele de tip perceptron multistrat și algoritmi de antrenare pentru acestea (*backpropagation* și metoda gradientului). În secțiunea 3 abordăm noua perspectivă oferită de disponibilitatea datelor masive (*Big Data*), ca factor de importanță majoră complementar, sinergic și convergent alături de calitatea și performanțele algoritmilor *deep learning* implicați astăzi în rețelele neuronale adânci, punctând importanța crucială a creării setului de date ImageNet și a competiției asociate ILSVRC (2010-2017) [48] pentru dezvoltarea acestora în domeniul vederii artificiale. Secțiunea 4 este dedicată rețelelor neuronale convoluționale, cu prezentarea particularităților și a elementelor componente, iar în secțiunea 5 sunt analizate câteva arhitecturi convoluționale de referință. Secțiunea 6, care încheie lucrarea, sintetizează câteva concluzii.

2. Rețele neuronale artificiale. Perceptronul. Perceptronul multistrat.

Un neuron artificial modelează practic într-un mod simplificat, printr-o funcție matematică, un neuron dintr-o rețea neuronală biologică (mai mulți neuroni conectați) cum este, de exemplu, creierul uman. Neuronul artificial reprezintă unitatea elementară într-o rețea neuronală artificială. Așa cum neuronul biologic are dendrite și axoni, neuronul artificial are o structură arborescentă simplă, cu noduri de intrare și un nod de ieșire conectat la toate nodurile de intrare. Neuronul artificial primește una, sau mai multe valori de intrare, analog potențialelor excitatoare / inhibitoare postsinaptice aplicate dendritelor neuronului biologic. Acestea sunt sumate, producând la ieșire o valoare de activare, analog al potențialului de acțiune transmis de-a lungul axonului în cazul neuronului biologic. Fiecare intrare x_i este ponderată individual (cu o pondere w_i) în sumă, iar rezultatul acesteia este transmis la ieșire printr-o funcție neliniară, denumită funcție de activare.

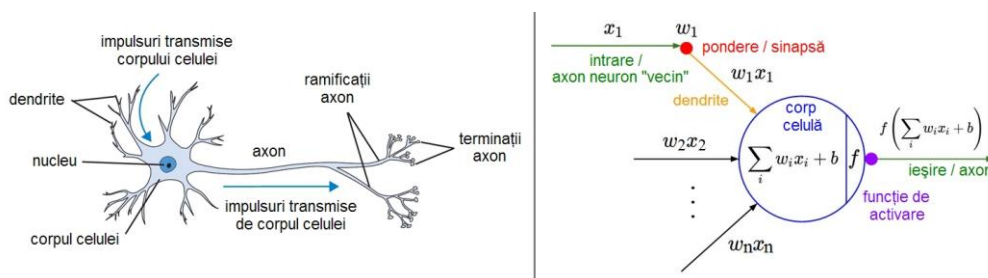


Figura 1. Ilustrarea modelării simplificate a neuronului biologic de către neuronul artificial (Adaptare după: Karpathy, 2015 [27])

Perceptronul

Cea mai simplă rețea neuronală artificială este „perceptronul“, format dintr-un singur neuron artificial. Modelul perceptronului a fost introdus de Frank Rosenblatt (1958) [36], având la bază (și) cercetările lui Warren McCulloch și Walter Pitts (1943) [33] vizând un model matematic pentru funcționarea neuronilor din creierul animal.

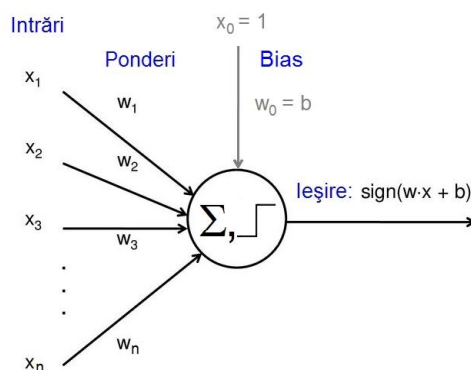


Figura 2. Perceptronul

Astfel, perceptronul calculează suma ponderată a n valori numerice de intrare x_i :

$$y = w \cdot x + b = \sum w_i \cdot x_i + b; \quad i = 1 \div n \quad (1)$$

și apoi, asupra rezultatului numeric astfel obținut, aplică o funcție neliniară, de ex. funcția semn (signum):

$$\text{sign}(y) = \begin{cases} -1 & \text{pentru } y < 0; \\ 0 & \text{pentru } y = 0; \\ +1 & \text{pentru } y > 0. \end{cases} \quad (2)$$

Termenul liber b (*bias / offset*), reprezintă valoarea de prag de activare a perceptronului și poate fi considerat ca o a $(n+1)$ -a intrare, suplimentară, de valoare $x_0 = 1$ și pondere $w_0 = b$ constante și permite translatarea funcției de activare la stânga sau la dreapta, după cum este necesar în procesul de antrenare (instruire).

Perceptronul poate realiza clasificarea binară (în două clase, denumite codificate de exemplu prin valorile 0 și 1) în cazul datelor liniar separabile aplicate la intrare, prin separarea valorilor de ieșire peste sau sub un anumit prag. Acesta, precum și ponderile asociate intrărilor trebuie stabilite empiric în etapa de configurare pentru fiecare problemă de clasificare în lipsa unui algoritm de antrenare. Acest neajuns a fost însă depășit odată cu algoritmul iterativ de adaptare (antrenare) a perceptronului propus de Rosenblatt, 1962 [37] pentru stabilirea valorilor ponderilor. Algoritmul pornește de la un set de ponderi aleatoare, nenule, mici, comparând în fiecare iterație ieșirea obținută pentru fiecare vector de intrare de antrenare cu clasa reală / corectă căreia îi aparține și urmărind ca prin ajustarea ponderilor să se elimine fiecare eventuală eroare de clasificare. Astfel, dacă pentru exemplul curent de instruire ieșirea corectă trebuia să fie 0 dar s-a obținut 1, atunci se diminuează toate ponderile corespunzătoare intrărilor având valoarea 1, iar pentru situația inversă acestea sunt crescute. Dacă la finalul unei iterații (după un număr de pași egal cu numărul vectorilor de instruire) nu s-a (mai) produs nicio modificare a ponderilor, înseamnă că toți vectorii de instruire respectivi au fost corect clasificați, iar vectorul ponderilor la care s-a ajuns reprezintă o soluție a problemei de instruire respective și algoritmul se termină cu succes. Dacă însă s-a modificat cel puțin o pondere, se trece la iterația următoare. În cazul în care într-un număr k prestabilit de iterații (suficient de mare, stabilit empiric) nu s-a ajuns la clasificarea corectă a tuturor vectorilor de instruire, înseamnă că cele două clase nu sunt liniar separabile și algoritmul este oprit, problema neputând fi rezolvată cu ajutorul perceptronului. Tot Rosenblatt, 1962 [37] a enunțat și demonstrat „Teorema de convergență a perceptronului”: Pentru un set de instruire format din două subseturi de vectori corespunzătoare fiecare pentru una din două clase liniar separabile, algoritmul de antrenare va converge după n iterații, rezultând un vector al ponderilor $w(n+k) = w(n)$, $\forall k > 0$, ca soluție a problemei de instruire.

Acest prim tip de modele liniare a definit primul val în domeniul rețelelor neuronale artificiale, cibernetica. Perceptronul are însă multiple limitări, dintre care cea mai cunoscută este aceea că nu poate modela funcția logică binară XOR, ale cărei valori 0 și 1 pentru toate combinațiile

posibile ale variabilelor de intrare binare nu sunt liniar separabile. Aceste limitări au făcut ca pentru o vreme rețelele neuronale să fie relativ abandonate.

Perceptronul multistrat. Antrenarea cu backpropagation

O rețea de perceptroni conectați formează „perceptronul multistrat“ (*multi-layer perceptron* – MLP). Acesta este în general reprezentat de o succesiune de straturi de neuroni, compusă dintr-un strat de intrare, unul sau mai multe așa numite straturi ascunse (*hidden layer*) și un strat de ieșire, conectate secvențial între ele. Straturile se comportă fiecare analog neuronului biologic. Ieșirile unui strat reprezintă intrări pentru stratul următor. Straturile sunt (în general) complet conectate, fiecare perceptron de pe un strat fiind conectat cu toți perceptronii de pe stratul anterior (complet conectat). Fiecare conexiune are o pondere individuală proprie. Astfel, vorbim în acest context de modele conecționiste, ca o a doua paradigmă și un al doilea val în domeniul rețelelor neuronale artificiale, al procesării paralele distribuite. Pe de altă parte, în cazul arhitecturilor de tip perceptron multistrat cu mai multe straturi ascunse, se poate vorbi deja despre o „rețea neuronală adâncă“ („*deep neural network*“), în timp ce în cazul unui singur strat ascuns vorbim de „rețea neuronală superficială“ („*shallow neural network*“).

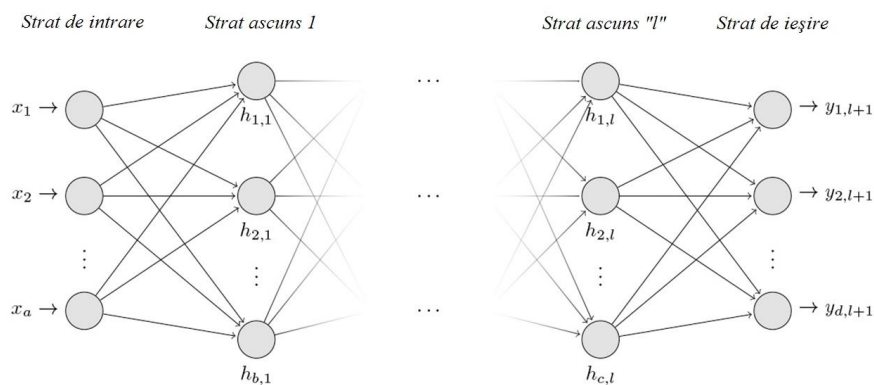


Figura 3. Arhitectura generală a perceptronului multistrat

Rețeaua de tip perceptron multistrat prezentată mai sus mai este de tip *feed-forward*, deoarece legăturile se fac de la intrare spre ieșire, strat după strat, exclusiv spre înainte. Valorile rezultate în straturi superioare nu influențează în niciun fel valorile de ieșire ale straturilor inferioare. Dacă în cazul perceptronului intrările sunt utilizate pentru a calcula imediat rezultatul, în cazul rețelelor multistrat *feed-forward* se efectuează prelucrări ale datelor secvențial pe fiecare strat unul după altul. Totodată, se folosesc alte funcții de activare, derivabile (al căror rezultat nu mai trece brusc de la negativ la pozitiv, ci are o trecere continuă), care să permită antrenarea. În general este utilizată o funcție de tip „sigmoidă“, cum sunt funcția logistică:

$$S(x) = 1 / (1 + e^{-x}), \quad (3)$$

sau tangenta hiperbolică:

$$S(x) = \tanh(x) = (e^x - e^{-x}) / (e^x + e^{-x}). \quad (4)$$

Rețeaua de tip perceptron multistrat *feed-forward* poate clasifica datele de intrare într-un număr prestabilit de clase. Vectorii de intrare X (de dimensiune a) trebuie să conțină valori specifice care să „codifice“ caracteristicile relevante pentru identificarea și discriminarea între categoriile/clasele de date de intrare care trebuie clasificate. Este deci necesar ca aceste caracteristici să fie anterior identificate / determinate / cunoscute pentru categoria din care face parte problema respectivă. Identificarea și extragerea caracteristicilor (*feature extraction*) este în general o sarcină netrivială. Datele sunt propagate exclusiv înainte prin rețea, strat după strat, de la intrare spre ieșire. Numărul ieșirilor (d) este dat de numărul prestabilit de valori rezultate posibile (numărul de clase) pentru seturile de date aplicate la intrare. Valorile ponderilor tuturor conexiunilor implicate sunt

stabilite prin învățare supervizată, pe baza unor seturi de vectori de intrare de antrenare/instruire „etichetați“ (a căror apartenență la o anumită clasă dintre cele posibile este apriori cunoscută). Metoda cea mai utilizată este cea a retropropagării erorii (*backpropagation*), propusă inițial de Paul Werbos, 1974 [47] în teza sa de doctorat la Harvard. Acesta a descris algoritmul de antrenare supervizată a rețelelor neuronale artificiale multistrat prin propagarea înapoi a erorilor dinspre straturile finale spre straturile inferioare și ajustarea corespunzătoare a ponderilor conexiunilor pentru minimizarea acestor erori. După ce o vreme nu a avut practic nicio răspândire, David Rumelhart et al., 1986 [38] au „redescoperit“ și utilizat cu succes algoritmul *backpropagation* la antrenarea rețelelor multistrat prin procesare paralelă distribuită (*parallel distributed processing*), făcând ca acesta să devină din acel moment foarte cunoscut și utilizat. Metoda poate fi rezumată ca un proces continuu de antrenare / învățare supervizată a rețelei, prin ajustarea și „acordul fin“ la fiecare iterație al structurii acesteia (respectiv a valorilor ponderilor conexiunilor neuronilor săi, care au fost în primă instanță inițializate cu valori aleatorii, mici). Este practic vorba despre un proces iterativ de optimizare prin care se urmărește diminuarea la fiecare iterație a erorii, respectiv a diferenței între ieșirea așteptată și cea obținută în iterația respectivă, estimată utilizându-se o funcție cost (sau *loss*). În acest mod se potențează practic acele intrări și caracteristici care sunt cele mai relevante pentru obținerea ieșirii dorite. Cel mai utilizat algoritm de optimizare pentru ajustarea ponderilor este metoda gradientului (*gradient descent*), cu diferite variante ale acesteia, pe baza căreia se decide la fiecare pas care anume ponderi, cum și cât să fie ajustate în sensul urmăririi gradientului descrescător al erorii.

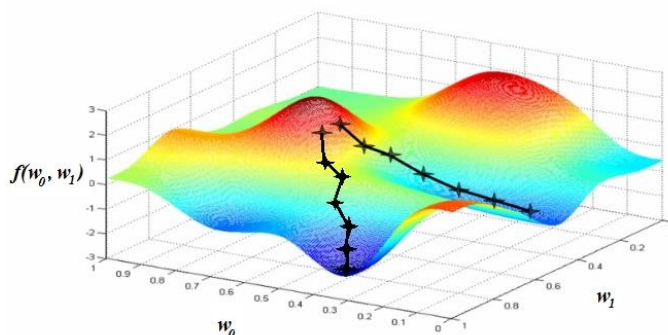


Figura 4. Metoda gradientului în backpropagation
(Adaptare după: Vryniotis, 2013 [44])

Pe măsură ce eroarea scade, fiecare nouă iterație devine mai rafinată, procesul putând necesita mii de iterații până când ieșirea calculată să se apropie suficient de mult de cea așteptată, moment în care se poate considera că rețeaua este complet antrenată.

Prin antrenarea unei rețele de tip perceptron multistrat se stabilesc practic legături între valori ale vectorilor de intrare (de instruire) și anumite valori de ieșire cunoscute asociate acestora, cumva ca în cazul definirii funcțiilor prin tabele de valori. În acest context, a fost formulată și demonstrată „teorema de aproximare universală“ (George Cybenko, 1989 [5]; Kurt Hornik, 1991 [22]), care afirmă că o rețea de tip perceptron multistrat *feed-forward* cu un (singur) strat ascuns și un număr finit de neuroni poate modela (aproxima polinomial) practic orice funcție continuă pe subseturi compacte din \mathbf{R}^n oricât de complexă, cu o precizie oricât de bună dacă sunt utilizați suficienți neuroni pe stratul ascuns și funcții de activare potrivite.

Astfel, cu algoritmul de învățare supervizată utilizând *backpropagation* și metoda gradientului, rețelele de tip MLP au însemnat un mare pas înainte. Totuși, trebuie observat faptul că pentru cazul mai multor straturi ascunse, algoritmul de învățare *backpropagation* bazat pe metoda gradientului are limitări serioase. Aceasta se datorează formei sigmoideale a funcției de activare a neuronilor care determină la derivare o saturare rapidă, ducând la efectul numit „dispariția/ estomparea gradientului“ (*vanishing gradient*) de la strat la strat înapoi. Prin urmare, în cazul rețelelor de tip perceptron multistrat antrenate cu *backpropagation* și metoda gradientului, numărul de straturi ascunse (adâncimea rețelei) nu poate fi efectiv prea mare, ponderile de pe primele straturi neputând fi practic antrenate astfel. Precizarea apriori a numărului de straturi ascunse (1 sau 2) necesare pentru o

rezolvare optimă a problemei de aproximare este relativ dificilă; de asemenea și a numărului de neuroni din stratul/straturile ascuns/ascunse. Există estimări, de ex. Kolmogorov (1957) [28], care arată că pentru aproximarea unei funcții de n variabile, sunt necesari $n(2n+1)$ neuroni în stratul ascuns, iar în cazul utilizării a două straturi ascunse, $(2n+1)$ neuroni. Totuși aceste estimări nu conduc întotdeauna la o soluție optimă. Mai trebuie făcute observațiile că numărul de neuroni necesar pe stratul ascuns crește practic exponențial cu numărul de intrări, iar antrenarea rețelelor intră în categoria problemelor „NP-complete“ care nu pot fi rezolvate prin aproximări polinomiale în timp. Aceste limitări, precum și existența unor algoritmi și metode de clasificare cu performanțe suficient de bune mai rapide computațional (arbori de decizie - 1985, *random forest* - 1995, SVM - 1996) au determinat ca evoluția pe linia MLP să pară un drum înfundat, iar rețelele neuronale să fie practic lăsate de-o parte pentru o vreme de către cei mai mulți dintre cercetătorii implicați.

3. Deep Learning și Big Data. Setul de imagini ImageNet

Totuși, în anul 2006 apare o schimbare importantă de paradigmă când Geoffrey Hinton [19; 20] introduce câteva idei revoluționare pentru inițializarea parametrilor rețelelor neuronale cu valori mai apropiate de valorile optime utilizând mașini Boltzmann restricționate (*Restricted Boltzmann Machines* – RBM). Se realizează mai întâi o antrenare nesupervizată strat cu strat a rețelei pentru ca ponderile să ajungă să modeleze structura intimă a datelor. Apoi, ponderile astfel inițializate sunt ajustate utilizându-se *backpropagation*. Cu o astfel de rețea preantrenată cu RBM s-a obținut o eroare de circa 1% pe setul de date MNIST (70.000 de imagini conținând fiecare câte o cifră, 60.000 de antrenare și 10.000 de test). În 2007, Yoshua Bengio [1] a introdus în locul RBM *autoencoder-e*, dezvoltate ulterior cu diferite variații, cum ar fi *denoising* și *sparse autoencoder*. În 2010, James Martens [32] a prezentat un alt algoritm pentru găsirea parametrilor, utilizând (și) derivate de ordinul 2, fără învățare nesupervizată prealabilă, care obținea rezultate mai bune decât cele de până atunci. Mai mult, tot în același an, utilizând direct algoritmul clasic de *back-propagation*, cu o rețea adâncă și lată, pe GPU, utilizând la antrenare (regiuni din) imagini ușor deformate elastic, fără niciun fel de algoritm ajutător pentru inițializarea ponderilor, Dan Cireșan [2] a obținut 0,35% eroare pe setul MNIST. Același grup, utilizând rețele neuronale convoluționale și *max pooling* [3] fără vreo altă antrenare prealabilă, a stabilit în 2011 recordul pe MNIST, de 0,23%, care reprezintă o eroare de clasificare mai mică decât cea umană. Se poate vorbi astfel despre un al treilea val, *deep learning*, în domeniul rețelelor neuronale artificiale, în ultimii ani acestea cunoscând o revigorare și o dezvoltare impresionantă, potențată și de evoluția tehnologică spectaculoasă în ceea ce privește capacitățile și vitezele de calcul și stocare, utilizarea procesoarelor grafice (*graphics processing unit* – GPU) pentru calcule paralele cu matrici, dispozitivele de achiziție de date etc., precum și disponibilitatea volumelor din ce în ce mai mari și mai variate de date (*Big Data*) în tot mai multe domenii. Au fost dezvoltate noi modele neuronale și arhitecturi de rețele neuronale adânci, inspirate fie din neuroștiințe, fie din rațiuni de inginerie computațională, tehnologia *deep learning* devenind la ora actuală practic sinonimă cu conceptele de inteligență artificială și învățare automată (*machine learning*).

Deși evoluția tehnologică în privința puterii și capacității de calcul și mai cu seamă apariția și dezvoltarea procesoarelor grafice (GPU) puternice a oferit o bază importantă pentru dezvoltarea arhitecturilor și algoritmilor de rețele neuronale adânci și obținerea unor performanțe superioare, acestea nu au putut totuși face un salt spectaculos până când nu s-a putut dispune și de seturi de date reale pentru antrenare, validare și testare suficient de mari, care să surprindă complexitatea și varietatea din lumea reală. De exemplu, în domeniul vederii artificiale, un moment de referință l-a reprezentat crearea ImageNet [48], o colecție de imagini de înaltă rezoluție din lumea reală etichetate pe baza dicționarului WordNet [34] de cuvinte din limba engleză organizat ierarhic pe logica „*machine-readable*“ în categorii de sinonime cognitive (*synsets*), realizat de George Miller de la Princeton începând cu sfârșitul anilor 1980, cu mai mult de 155.000 de cuvinte indexate. Această întreprindere a fost demarată în 2007 și coordonată de Fei-Fei Li, inițial la Princeton, astăzi conducător de cercetări la Google Cloud, profesor asociat și director al laboratoarelor de inteligență artificială și respectiv de vedere artificială la Universitatea Stanford. Pentru completarea setului de date inițial au fost necesari doi ani și jumătate. Au fost etichetate 3,2 milioane de imagini,

separate în 5.247 de categorii, ordonate în 12 subarbori ca: „mammal“, „vehicle“, „furniture“ ș.a.m.d., respectiv circa 500-600 imagini per nod. Setul de imagini ImageNet a fost făcut public inițial la conferința de top Computer Vision and Pattern Recognition – CVPR 2009, ca poster de cercetare [6]. Astăzi a ajuns la peste 14 milioane de imagini din peste 21.800 de categorii/clase, cu peste 500 de imagini pentru fiecare dintre acestea.

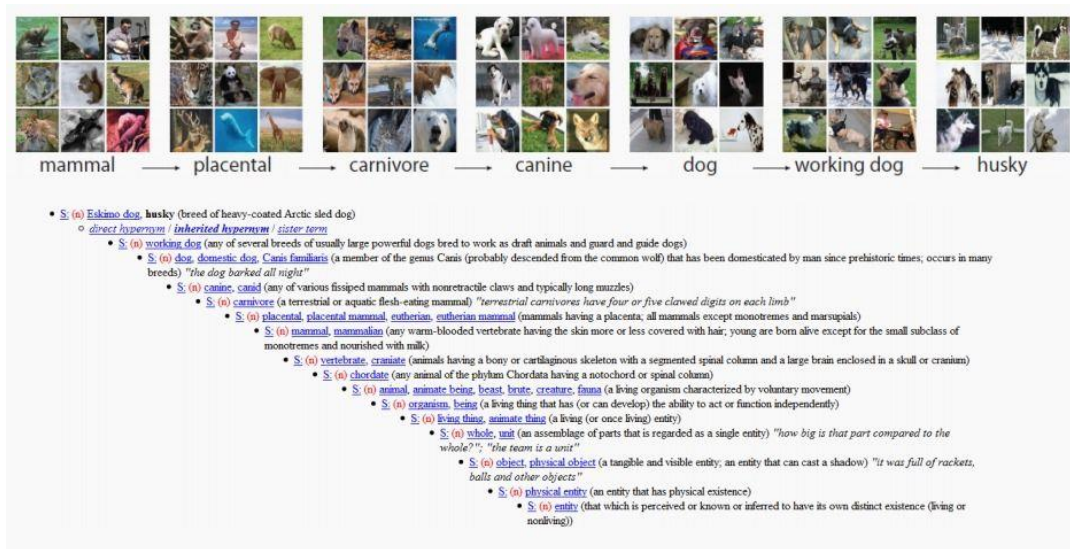


Figura 5. Structura ierarhică a ImageNet pliată pe cea a WordNet (Sursa: Gershgorn, 2017 [12])

ImageNet a devenit una din referințele cele mai utilizate pe plan mondial pentru măsurarea performanțelor modelelor și arhitecturilor neuronale convoluționale în recunoașterea și localizarea obiectelor în imagini, precum și un real catalizator pentru evoluția explozivă a acestora, inclusiv prin organizarea competiției anuale ImageNet Large Scale Visual Recognition Challenge - ILSVRC (2010-2017). În cadrul acesteia au fost utilizate subseturi de aproximativ 1,2 milioane de imagini de antrenare, 50.000 de imagini de validare și 150.000 de imagini de testare, cu rezoluții diferite, reprezentând aproximativ câte 1.000 de exemple din 1.000 de categorii/clase diferite. Competiția din 2012 a fost câștigată pentru prima dată de o arhitectură neuronală convoluțională adâncă, denumită AlexNet [29] (SuperVision Group: Geoffrey Hinton, Ilya Sutskever și Alex Krizhevsky – Universitatea din Toronto), cu 8 straturi, care a produs un salt spectaculos în calitatea clasificării/recunoașterii cu o rată de eroare de 15,4%, mai bună cu 10,8% decât cea mai bună performanță anterioară, și cu 41% mai bună ca următorul clasat. A urmat o explozie a utilizării rețelelor convoluționale în diferite scopuri (adnotarea automată a fotografiilor, conducerea automată a vehiculelor, recunoașterea vorbirii, traducerea automată etc.) și totodată o continuă îmbunătățire a acestora, ajungându-se la obținerea de performanțe comparabile cu – și chiar peste – capacitatea umană (eroare 5-10%). Matthew D. Zeiler (ulterior fondator al Clarifai) și Rob Fergus au câștigat competiția din 2013 cu ZFNet [45] tot cu 8 straturi dar cu o rată de eroare de 11,2%, iar în 2014 câștigători au fost GoogLeNet/Inception-v1 [43] (Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Drago Anguelov, Dumitru Erhan, Andrew Rabinovich – Google) și VGGNet [41] (Visual Geometry Group – Karen Simonyan și Andrew Zisserman – Universitatea Oxford, ulterior cooptați de Google în laboratorul DeepMind) cu 22, respectiv 16-19 straturi și rate de eroare de 6,7% și respectiv 7,3%. Dimensiunile arhitecturilor câștigătoare au crescut, ajungând în 2015 la 152 de straturi, cu rată de eroare de 3,57% (ResNet [18] – Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun – Microsoft Research Asia) și la peste 200 de straturi în 2016 și 2017 și rate de eroare de 2,99% și respectiv de 2,251% (DenseNet [25] și SENet [24]).

În doar 7 ani, acuratețea câștigătoare în clasificarea obiectelor din setul de date ImageNet a crescut de la 71,8% la peste 97%, depășind capacitățile umane (cca. 90-95%) și dovedind efectiv că date mai multe conduc la decizii mai bune. Este unanim recunoscut astăzi faptul că apariția ImageNet a contribuit substanțial la schimbarea de paradigmă în abordarea cercetărilor în domeniul inteligenței

artificiale care a adus ca importanță datele la același nivel cu algoritmi. Între timp, odată ce s-a dovedit că algoritmi și arhitecturile *deep learning* necesită date vaste de genul ImageNet, au mai fost introduse alte astfel de seturi de date de către Google (Open Images – cu 9 milioane de imagini în 6.000 de categorii), Microsoft și Institutul Canadian pentru Cercetări Avansate, ulterior de către Facebook și Amazon, care și-au creat propriile seturi de date interne. Totodată au apărut tot felul de seturi similare pentru alte tipuri de date: de la video, la vorbire, jocuri etc.

Succesul actual al tehnologiilor *deep learning* se datorează în mare măsură (și) faptului că în ultimii ani au devenit disponibile cantități uriașe de date în mai toate domeniile, coroborat cu evoluția tehnologică în ceea ce privește capacitatea de stocare și gestionare a datelor și viteza de accesare și de procesare a acestora odată cu dezvoltarea memoriilor și suporturilor de stocare rapide și a noilor generații de procesoare multicore și de procesoare grafice (GPU), care a permis și dezvoltarea de modele mai mari, cu mai multe straturi și mai mulți neuroni. Se consideră astăzi la modul grosier că un algoritm de *deep learning* supervizat va performa acceptabil dacă este antrenat cu circa 5.000 de exemple per categorie și va egala sau surclasa performanța umană dacă este antrenat pe un set de date de cel puțin 10 milioane de exemple etichetate (Ian Goodfellow, 2016 [15]). Pentru a avea același succes cu seturi de date mai restrânse, sunt efectuate astăzi cercetări prin care să se valorifice cantitățile mari de date neetichetate cu metode de învățare nesupervizată sau semisupervizată.

4. Rețele neuronale convoluționale (RNC)

Un precursor inspirațional pentru rețele neuronale convoluționale l-a reprezentat *neocognitronul* propus de Kunihiko Fukushima, 1980 [8], (după ce în 1975 introdusese *cognitronul* [7]). Este vorba despre o structură neuronală ierarhică multistrat capabilă să recunoască tipare vizuale prin învățare, prin analogie cu ceea ce se întâmplă în cortexul vizual uman (Hubel & Wiesel, 1959 [26]), compusă din straturi alternative de celule primare (S) antrenate să răspundă selectiv la o anumită caracteristică locală (*local feature*) prezentă în câmpul lor receptor (*receptive field*) și straturi de celule complexe (C) care permit abateri în poziționarea/localizarea caracteristicilor.

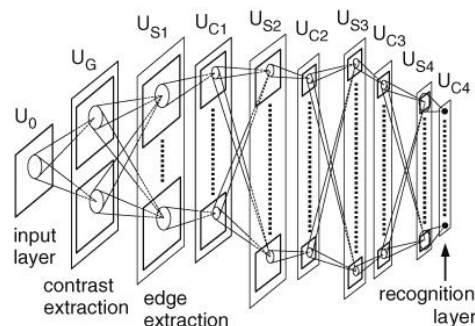


Figura 6. Arhitectura neocognitronului (Sursa: Fukushima, 2007 [11])

Fiecare strat C are ca intrări ieșirile stratului S anterior care sunt fixe, iar stratul S are intrări variabile care se modifică în cursul învățării, fiecare celulă S ajungând să răspundă selectiv la o anumită caracteristică apărută în câmpul ei receptor asociat. În straturile inițiale sunt extrase caracteristici locale simple (ca: muchii, linii cu diferite orientări), în timp ce în straturile finale se ajunge, prin integrarea celor extrase în straturile anterioare, la caracteristici mai „globale” (ca tipare de învățare). Conexiunile de intrare ale celulelor C provin de la ieșirile unui grup localizat de celule S din stratul anterior care extrag aceeași caracteristică dar în poziții ușor modificate. Celule C răspund dacă măcar una dintre respectivele celulele S este activată la ieșire. Astfel se asigură o anumită invarianță la mici translații, generându-se însă totodată și un efect de încetșare la propagarea între straturile succesive. Fiecare strat S sau C este împărțit în substraturi denumite „plane de celule” (matrici 2-D de celule plasate retinotopic și partajând același set de intrări și oferind astfel o simetrie translațională pentru acestea), corespunzător caracteristicilor la care respectivele celule răspund. De-a lungul timpului au fost propuse diferite modificări și variante ale neocognitronului. De exemplu, prin adăugarea de conexiuni inverse (dinspre straturile superioare înapoi), s-a obținut abilitatea recu-

noașterii corecte chiar și a unor tipare (parțial) acoperite/mascate și pot fi reconstruite părțile ascunse ale acestora (Fukushima, 1987; 2005 [9; 10]). Chiar și în cazurile în care apar simultan mai multe tipare, neocognitronul își focalizează atenția asupra fiecăruia, unul câte unul și le recunoaște corect.

În fine, prima aplicație de succes a rețelelor neuronale convoluționale (RNC) datează din anii '90 și este reprezentată de arhitectura LeNet-5 (Yann LeCun, 1989; 1998 [30; 31]).

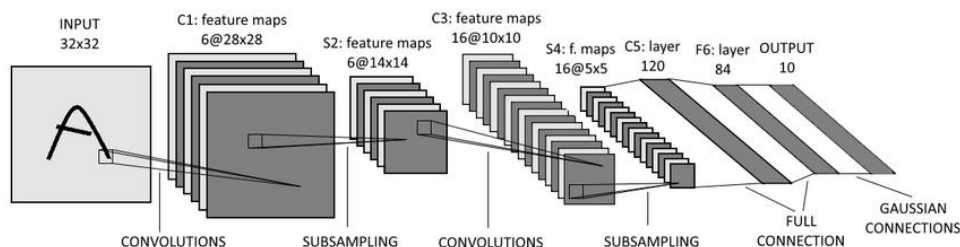


Figura 7. Arhitectura LeNet-5 (Adaptare după: LeCun, 1998 [31])

Aceasta a căpătat o largă utilizare practică în recunoașterea cifrelor scrise de mână pentru „citirea” automată a cecurilor și a altor formulare similare completate cu cifre în mediul bancar.

Rețelele neuronale convoluționale (RNC) sunt astăzi cele mai utilizate pentru detecția și segmentarea obiectelor în imagini, precum și pentru recunoașterea / clasificarea obiectelor și scenelor. Principalele avantaje ale RNC sunt:

- elimină necesitatea extragerii manuale a caracteristicilor, acestea fiind învățate în mod automat direct din datele de antrenare;
- nivelul de performanță la recunoaștere este extrem de ridicat (comparabil cu, sau peste cel uman);
- pot fi reînvățate pentru noi sarcini de recunoaștere și permit construcții ulterioare pe baza unor (părți inițiale din) RNC preantrenate (*transfer learning*).

Foarte pe scurt, o rețea convoluțională este practic compusă din două părți. Prima parte alternează straturi neuronale cu conexiuni locale de tip convoluțional și respectiv de agregare pe bază de maxim sau medie (*max pooling / average pooling*) cu reducere dimensională (*subsampling*). Această parte „convoluțională” realizează extragerea automată a caracteristicilor specifice (*features extraction*) din datele de intrare. Partea finală a unei astfel de rețele adânci este reprezentată de unul sau mai multe straturi complet conectat(e) de tip perceptron multistrat (MLP) care realizează partea de clasificare utilizând ca intrări ieșirile părții inițiale convoluționale (caracteristicile extrase automat de aceasta) obținute pentru fiecare imagine aplicată intrării rețelei. Cu alte cuvinte, prima parte, cea convoluțională, transformă în mod automat imaginea brută aplicată la intrarea RNC într-un vector de caracteristici care la rândul său este aplicat ca intrare clasificatorului MLP final (și care în particular poate fi și un SVM sau orice altă variantă clasică utilizând ca intrare vectori de caracteristici). Vom detalia în cele ce urmează principalele componente și mecanisme implicate în RNC.

Straturile convoluționale

Convoluția discretă a două funcții 1-D f și g definite pe mulțimea numerelor întregi Z este dată de:

$$(f * g)[j] = \sum f[i] \cdot g[j - i] = \sum f[j - i] \cdot g[i], \quad (5)$$

iar în 2-D relația poate fi scrisă la modul general:

$$(a * b)[i, j] = \sum \sum a[m, n] \cdot b[i - m, j - n]. \quad (5')$$

Indicii de sumare (i în primul caz, m și n în cel de-al doilea) iau valori de la minus la plus infinit. În particular, convoluția 2-D a două matrici pătrate de exemplu este ilustrată în figura următoare.

$$\begin{array}{ccc}
 \begin{array}{c} A(4 \times 4) \\
 \begin{array}{|c|c|c|c|}
 \hline 1 & 1 & 0 & 1 \\
 \hline 1 & 0 & 0 & 1 \\
 \hline 0 & 1 & 1 & 0 \\
 \hline 1 & 1 & 1 & 1 \\
 \hline
 \end{array}
 \end{array}
 & * &
 \begin{array}{c} B(2 \times 2) \\
 \begin{array}{|c|c|}
 \hline 1 & 0 \\
 \hline 1 & 1 \\
 \hline
 \end{array}
 \end{array}
 =
 \begin{array}{c} C(3 \times 3) \\
 \begin{array}{|c|c|c|}
 \hline 2 & 1 & 1 \\
 \hline 2 & 2 & 1 \\
 \hline 2 & 3 & 3 \\
 \hline
 \end{array}
 \end{array}
 \end{array}$$

Figura 8. Exemplu de convoluție a două matrici pătrate A (4×4) și B (2×2), rezultând matricea pătrată C (3×3) (Adaptare după: Zhang, 2018 [46])

Fiecare element din matricea rezultat C este obținut ca sumă a produselor elementelor corespunzătoare din aceleași poziții din matricea (filtru, sau nucleu / *kernel*) de convoluție B și din regiunea din matricea inițială A acoperită de B, prin glisarea matricii B în poziții succesive peste toată matricea A pe orizontală și pe verticală cu același pas (*stride*) $s = 1$.

Se poate observa o reducere dimensională în urma aplicării operației de convoluție spațială. Dacă notăm cu a, b și c dimensiunile matricilor pătrate A, B și respectiv C (în exemplul de mai sus $a = 4$, $b = 2$, $c = 3$), atunci:

$$c = a - b + 1. \quad (6)$$

De exemplu, pentru aceeași matrice A de dimensiune 4x4, dacă B ar fi de dimensiune 3x3, atunci C ar fi de dimensiune 2x2 (pentru un același pas $s = 1$).

Pentru a se evita acest efect de reducere a dimensiunilor spațiale, uneori este utilizată o metodă de completare pe contur a matricii inițiale prin adăugarea uneia sau mai multor linii și respectiv coloane, după caz, cu elemente de valoare 0 (zero) în jurul acesteia (*zero padding*) și aplicarea filtrului pe noua matrice astfel generată. Fie p numărul elementelor cu valoare zero astfel adăugate pe fiecare direcție, la fiecare din margini; atunci:

$$c = a - b + 2p + 1, \quad (7)$$

iar pentru ca dimensiunea spațială să nu se modifice (adică $c = a$), trebuie ca:

$$p = (b - 1) / 2. \quad (8)$$

În cazurile prezentate mai sus, s-a considerat pasul de avansare pe orizontală și verticală a „ferestrei” B peste matricea A, (*stride*), $s = 1$, acesta putând lua însă și alte valori, dar cel mult egale cu dimensiunea 1-D a filtrului, b). În general, dimensiunea rezultatului aplicării unui astfel de filtru este dată de relația:

$$c = (a - b + 2p) / s + 1, \quad (9)$$

cu trunchiere (rotunjire în jos) la valoarea întreagă dacă este cazul.

Imaginile digitale sunt reprezentate ca matrici spațiale 2-D de numere întregi, cu precizarea că mai intervine și numărul de plane/canale de culoare, sau adâncimea de culoare, care de exemplu în cazul imaginilor color RGB este 3. Astfel, filtrele de convoluție aplicate pe o astfel de intrare RGB care reprezintă un volum 3-D ($H_i \times W_i \times 3$), vor trebui să fie și ele, fiecare, volume 3-D ($H_f \times W_f \times 3$). Rezultatul aplicării unui singur astfel de filtru/nucleu (*kernel*) de convoluție va fi însă 2-D ($H_o \times W_o$). În cazul aplicării unui număr de K astfel de filtre, rezultatul devine și el un volum 3-D ($H_o \times W_o \times K$), pe lângă dimensiunile spațiale de tip lățime și înălțime obținute prin convoluție adăugându-se și o a treia dimensiune dată de numărul K al acestor filtre.

În cazul straturilor convoluționale, spre deosebire de cele complet conectate precum la perceptronul multistrat (MLP), un neuron de pe un astfel de strat este conectat numai cu câțiva alți neuroni de intrare (de pe stratul anterior), grupați într-o vecinătate localizată denumită câmp receptor (*receptive field*), analog mecanismelor vizuale din creier. Aceasta se realizează efectiv prin aplicarea unui filtru/nucleu de convoluție glisat peste distribuția de intrare. Elementele filtrelor/ nucleelor

lor de convoluție (care sunt, așa cum am arătat, matrici 3-D) sunt reprezentate de valorile ponderilor conexiunilor respective între câmpul receptor de pe stratul anterior, de dimensiuni spațiale egale cu cele ale filtrului și neuronul corespunzător din stratul curent.

În general, dacă pe o imagine de intrare cu C canale de culoare, de dimensiuni $H_i \times W_i \times C$, sunt aplicate K filtre de convoluție de dimensiuni $H_f \times W_f \times C$, se obține o ieșire de dimensiuni $H_o \times W_o \times K$ (prin convoluție, volume „trec” în alte volume), unde dimensiunile ieșirii sunt:

$$H_o = (H_i - H_f + 2p) / s + 1, \quad (10)$$

$$W_o = (W_i - W_f + 2p) / s + 1 \quad (10')$$

Dimensiunile volumului de intrare $H_i \times W_i \times C$, dimensiunea filtrului/nucleului de convoluție (sau a câmpului receptor) $H_f \times W_f \times C$, pasul acestuia (*stride*) s și „grosimea” de *padding* p la capetele dimensiunilor spațiale ale intrării, sunt referite ca **hiperparametrii stratului convoluțional**.

Mai facem următoarele observații: dimensiunile spațiale ale filtrelor/nucleelor de convoluție sunt (în general) numere impare, pentru a exista un element central căruia să-i corespundă rezultatul aplicării respectivului filtru; a treia dimensiune, cea care indică numărul de canale de culoare, C , este întotdeauna egală cu aceeași dimensiune respectivă a hărții de intrare pe care se aplică filtrul convoluțional respectiv.

Un strat convoluțional complet mai conține și o neliniaritate aplicată asupra rezultatului convoluției cu filtrele specifice stratului, funcția de activare a neuronului, care este cel mai adesea de tip ReLU – *Rectified Linear Unit* (Hahnloser 2000; 2001 [16; 17]; Nair, 2010 [35]; Glorot, 2011 [14]), ieșirea acestui strat fiind:

$$Y = g(f(X)) = \max(0, f(X)). \quad (11)$$

Astfel, pentru un strat convoluțional l , cu intrarea X^{l-1} ($X^0 = (x^0_1, x^0_2, \dots, x^0_n)$ reprezentând vectorul de intrare al rețelei), rezultatul aplicării convoluției cu fiecare filtru este o combinație liniară f :

$$Y^l = f(X^{l-1}) = X^{l-1} \cdot W^l + b^l = \sum x^{l-1}_i \cdot w^l_i + b^l, \quad (12)$$

iar ieșirea acestuia este dată de aplicarea neliniarității (ReLU), g :

$$X^l = g(Y^l) = g(f(X^{l-1})) = \max(0, f(X^{l-1})) \quad (13)$$

și furnizează „harta activărilor” (*activation map*), sau „harta caracteristicilor” (*feature map*) stratului curent l .

Mai facem observațiile că: „adâncimea” unei astfel de hărți este egală cu numărul filtrelor de convoluție aplicate în stratul respectiv (K), iar numărul total de parametri – ponderi w^l și *bias*-uri b^l – care definesc un strat convoluțional (n_param) este determinat de dimensiunea și numărul acestor filtre (K):

$$n_param = (H_f \times W_f \times C + 1) \times K. \quad (14)$$

De exemplu, în cazul a $K = 5$ filtre $2 \times 2 \times 3$ aplicate fără *padding* asupra unei imagini de intrare RGB $4 \times 4 \times 3$, cu pas $s = 1$, numărul de parametri este $(2 \times 2 \times 3 + 1) \times 5 = 65$, iar harta de activări care se obține are $3 \times 3 \times 5 = 45$ neuroni. În cazul a $K = 50$ de filtre de exemplu, numărul de parametri este de 650, iar harta de activări conține $3 \times 3 \times 50 = 450$ neuroni.

Trebuie remarcat faptul că, spre deosebire de cazul rețelelor neuronale complet conectate de tip MLP, în cazul RNC numărul de parametri nu depinde de dimensiunea imaginii de intrare. Astfel, pentru o imagine de intrare de 100 de ori mai mare ($40 \times 40 \times 3$), tot 650 parametri sunt necesari în stratul convoluțional cu 50 de filtre exemplificat mai sus, în timp ce pentru situația complet conectată numărul parametrilor necesari ar fi crescut și el de 100 ori, la 65.000 parametri.

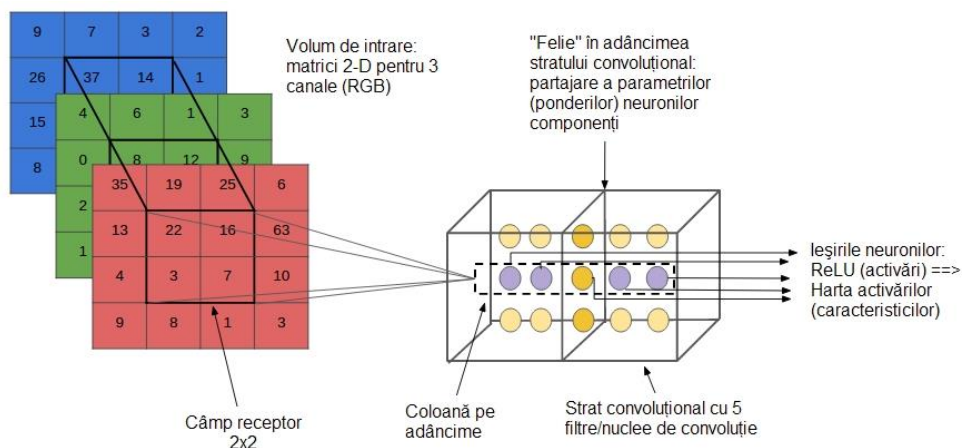


Figura 9. Straturile convoluționale transformă volume în volume
(Adaptare după: Saxena, 2016 [39])

Mai trebuie subliniat faptul că fiecare neuron dintr-o „felie” (*slice*) a unei hărți a activărilor dintr-un strat convoluțional – „felie” reprezentând harta unei anumite caracteristici obținută prin aplicarea filtrului/nucleului de convoluție (*kernel map*) corespunzător „feliiei” respective – partajează un același set de ponderi, specific astfel pentru întreaga „felie” respectivă, ceea ce contribuie la asigurarea invarianței la translații (caracteristica respectivă poate apărea în orice poziție). Numărul acestor ponderi este dat de dimensiunea filtrului respectiv, $f \times f$. Toate „feliile” corespunzând aplicării tuturor filtrelor din stratul respectiv partajează fiecare un același *bias* unic, b , pentru generarea hărții activărilor stratului respectiv.

Straturile de agregare (pooling) și reducere dimensională (down sampling)

Aceste straturi au rolul de a înlocui ieșirile furnizate de aplicarea neliniarităților ReLU finale ale unui strat convoluțional precedent (harta activărilor rezultată din acesta) cu o statistică sumară locală a acestora și de a furniza o reducere dimensională corespunzătoare totodată. Rezoluția detaliilor este astfel diminuată, dar este propagată în straturile superioare informația esențială integratoare, asigurându-se totodată evitarea exploziei numărului de parametri implicați în acestea. Putem vorbi de straturi de *max pooling* care extrag valori locale maxime, sau de *average pooling*, care extrag valori locale medii care sunt propagate mai departe în rețea. Pentru fiecare regiune din matricea de intrare acoperită de „filtrul de agregare” (câmp receptor – *receptive field*) este preluată din aceasta în matricea de ieșire corespunzătoare fie valoarea locală maximă, fie, respectiv, media valorilor locale din respectiva regiune.

Ca și în cazul straturilor convoluționale, dimensiunea „filtrului de agregare” f_p , pasul s_p de glisare a acestuia peste matricea de intrare (ieșirea – harta caracteristicilor – stratului convoluțional anterior) de dimensiune n_c și mărimea *padding*-ului p_p dacă este cazul, determină dimensiunea ieșirii n_p :

$$n_p = (n - f_p + 2p_p) / s_p + 1. \quad (15)$$

În general, RNC utilizează straturi de agregare de tip *max pooling*. De obicei, acestea nu implică *padding* ($p = 0$) și adesea $s = f$, astfel încât, practic $n_p = n / f$, iar pentru $f = 2$, $n_p = n / 2$, adică se obține o înjumătățire a fiecărei dimensiuni spațiale.

O justificare intuitivă pentru aplicarea unui asemenea tip de agregare prin reținerea valorilor locale maxime poate fi sugerată de faptul că valorile obținute prin convoluție dau o măsură a gradului de corelație între filtrul aplicat și câmpul receptor asupra căruia acesta a fost aplicat, valorile maxime indicând corelație maximă. Deci acestea apar la identificarea caracteristicii specifice filtrului respectiv în câmpul receptor. Prin urmare, valoarea maximă furnizează informația cea mai semnificativă în ceea ce privește prezența sau absența caracteristicii respective în câmpul receptor.

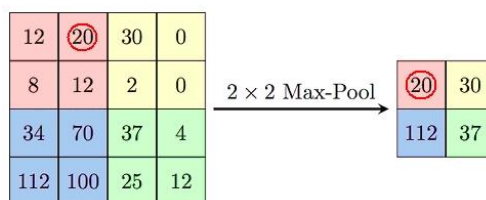


Figura 10. Exemplu de max pooling 2×2 ($f = 2$), fără padding, cu pas $s = 2$, fără suprapunere ($s = f$), rezultând o matrice pătrată (2×2) cu dimensiuni înjumătățite (Adaptare după: Karpathy, 2018 [27])

În acest mod, un astfel de strat de agregare contribuie totodată și la asigurarea unei anumite invarianțe/independențe față de mici translații în poziția obiectelor în imagini. Pe de altă parte, trebuie amintit faptul că aplicarea unui același filtru pe toate câmpurile receptoare localizate dar care pe ansamblu acoperă în întregime intrarea stratului convoluțional respectiv asigură independența totală față de poziția în care este identificată caracteristica asociată aceluia filtru.

Mai precizăm că pe măsură ce se avansează spre straturile convoluționale (și de agregare) superioare, dimensiunea câmpului receptor din imaginea inițială „acoperit” de un neuron dintr-o hartă de caracteristici (*feature map*) corespunzătoare acestuia, respectiv „apertura” respectivului strat, r , crește strat după strat, astfel:

$$r^l = r^{l-1} + (f^l - 1) \cdot d^{l-1}, \quad (16)$$

$$d^l = s^l \cdot d^{l-1}, \quad (17)$$

unde f^i este dimensiunea filtrelor din stratul i , s^i este pasul de aplicare al acestora, iar d^i este distanța între două câmpuri receptoare din imaginea inițială corespunzătoare la două caracteristici adiacente; pentru stratul (imaginea) de intrare, $d^0 = 1$.

Un criteriu pentru cât de adâncă trebuie să fie o anumită arhitectură de rețea (câte straturi să conțină), evident corelat și cu dimensiunea filtrelor și pasul utilizat în fiecare strat de convoluție și de agregare, este și acela ca apertura (câmpul receptor inițial al) unui neuron din harta de caracteristici finale să acopere (aproximativ) toată imaginea aplicată la intrarea rețelei convoluționale.

Stratul/straturile finale complet conectate

Legătura între partea convoluțională a RNC și clasificatorul final din aceasta se realizează printr-un așa-numit „strat de aplatizare” (*flatten layer*), un pseudostrat care nu produce modificări de valori, ci are doar rolul de a adapta ieșirea 3-D rezultată la finalul porțiunii convoluționale, prin translatare în 1-D, pentru utilizarea ca vector de caracteristici la intrarea clasificatorului final. Acesta din urmă este practic un perceptron multistrat cu funcții de activare de tip softmax, furnizând la ieșire, pe baza vectorului de intrare, un număr de probabilități egale cu numărul de clase definite pentru spațiul problemei, câte o probabilitate pentru fiecare dintre acestea. Antrenarea rețelelor neuronale convoluționale se face pe datele de antrenare prin grupare aleatoare a acestora în (mini)loturi a căror dimensiune este limitată de disponibilul de memorie al GPU, pe epoci cu ajustarea parametrilor de învățare când este cazul, și validare pe datele de validare după fiecare epocă. În general se utilizează tehnica retropropagării erorilor (*backpropagation*) cu metoda gradientului (*gradient descent*) în toate straturile (complet conectate și convoluționale). Pentru evitarea atenuării gradientului (*vanishing gradient*) prin mai multe straturi, se utilizează funcții de activare a neuronilor neliniare de tip ReLU, care evită saturația celor de tip sigmoidă, asigurând totodată și o reducere a numărului de ieșiri active (cele cu valori negative căpătând valoarea zero). Fiind implicat un număr foarte mare de parametri, pentru evitarea *overfitting*-ului și asigurarea unei generalizări coerente și consistente sunt utilizate diferite tehnici de regularizare precum normalizarea inițială și extinderea setului de date de intrare la antrenare (*data augmentation*), *dropout*-ul, normalizarea locală a valorilor de activare ale neuronilor la ieșirea unor straturi, normalizarea pe (mini)loturi (*(mini-)batch normalization*) ș.a.m.d. Vom încerca ilustrarea pe scurt a acestora în cele ce urmează, în contextul prezentării a două dintre arhitecturile neuronale convoluționale de referință dintre cele mai utilizate: AlexNet și VGG.

5. Arhitecturi RNC de referință

AlexNet (2012)

Arhitectura neuronală convoluțională (modelul) AlexNet (Krizhevsky, 2012 [29]) a câștigat competiția ImageNet ILSVRC 2012 cu o îmbunătățire semnificativă a celei mai bune performanțe privind rata de recunoaștere obținută cu arhitecturi clasice. Cea mai mare rețea proiectată și realizată până în acel moment, AlexNet a fost astfel prima rețea neuronală convoluțională care și-a dovedit avantajele și superioritatea netă față de cele mai performante abordări neconvoluționale anterioare, devenind un important punct de referință și catalizator pentru dezvoltările și îmbunătățirile ulterioare în domeniul rețelelor convoluționale.

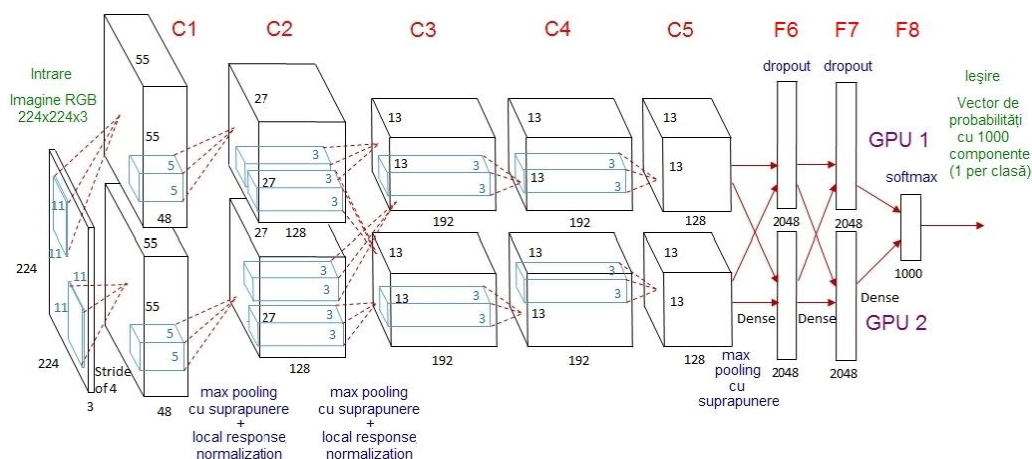


Figura 11. Arhitectura rețelei AlexNet (Adaptare după: Krizhevsky, 2012 [29])

AlexNet utilizează însă la intrare imagini RGB de dimensiune constantă, fixă, 224×224 . De aceea imaginile, în general dreptunghiulare de dimensiuni diferite, au fost inițial scalate astfel încât dimensiunea cea mai mică a acestora să ajungă 256 pixeli, apoi fiind decupate regiunile centrale după caz, astfel încât să se obțină imagini de dimensiune constantă 256×256 . Totodată, s-a procedat la o normalizare a setului de imagini de antrenare pe baza valorii medii pentru fiecare canal de culoare RGB. Din setul astfel obținut au fost practic utilizate fragmente 224×224 așa cum se va arăta mai jos.

Rețeaua AlexNet este compusă din opt straturi: primele cinci convoluționale și ultimele trei complet conectate. Ieșirea ultimului strat complet conectat produce o distribuție de probabilitate peste cele 1.000 de (etichete de) clase utilizând funcția softmax. Adâncimea rețelei s-a dovedit a fi importantă. Astfel, dacă se elimină un strat convoluțional (care conține cel mult 1% din parametrii modelului), nivelul de performanță scade cu 1-2%.

În primul strat convoluțional sunt aplicate asupra imaginilor de intrare RGB normalizate de dimensiune $224 \times 224 \times 3$ un număr de 96 de filtre/nuclee de dimensiune $11 \times 11 \times 3$, cu suprapunere, cu pas (stride) de 4 pixeli (acesta reprezentând distanța între centrele a două câmpuri receptoare corespunzând la doi neuroni alăturați) și fără padding. Pe harta de caracteristici $55 \times 55 \times 96$ rezultată se aplică apoi o agregare *max pooling* 3×3 , cu suprapunere, cu pas (stride) de 2 și cu normalizarea locală a răspunsului (*local response normalization*), rezultând la ieșirea stratului o hartă de caracteristici de dimensiuni $27 \times 27 \times 96$.

Al doilea strat convoluțional se aplică acestei ieșiri a celui dintâi pe care o filtrează cu 256 de nuclee de dimensiune $5 \times 5 \times 48$, cu suprapunere, cu pas (stride) de 1 pixel și *padding* 2. Și pe harta de caracteristici $27 \times 27 \times 256$ rezultată aici se aplică o agregare *max pooling* 3×3 , cu suprapunere, cu pas (stride) de 2 și cu normalizarea răspunsului local, rezultând la ieșirea stratului o hartă de caracteristici de dimensiuni $13 \times 13 \times 256$. Straturile convoluționale trei-patru și patru-cinci sunt conectate între ele fără straturi de agregare sau normalizare intermediare.

Al treilea strat convoluțional aplică 384 de nuclee de dimensiune $3 \times 3 \times 256$ pe ieșirea normalizată și agregată a celui de-al doilea strat, cu suprapunere, cu pas (*stride*) de 1 pixel și *padding* 1, rezultând la ieșirea stratului o hartă de caracteristici de dimensiuni $13 \times 13 \times 384$.

Al patrulea strat convoluțional are 384 nuclee de dimensiune $3 \times 3 \times 192$, care sunt aplicate ieșirii stratului al treilea, cu pas (*stride*) de 1 pixel și *padding* 1, rezultând la ieșirea stratului o hartă de caracteristici tot de dimensiuni $13 \times 13 \times 384$.

În fine, al cincilea strat convoluțional are 256 nuclee de dimensiune tot $3 \times 3 \times 192$, care sunt aplicate ieșirii stratului al patrulea, cu pas (*stride*) de 1 pixel și *padding* 1. Rezultă o hartă de caracteristici de dimensiune $13 \times 13 \times 256$, asupra căreia este aplicată o agregare *max pooling* 3×3 finală, cu suprapunere, cu pas (*stride*) de 2, rezultând la ieșirea stratului (și al părții convoluționale) o hartă de caracteristici de dimensiuni $6 \times 6 \times 256$.

Straturile șase și șapte, complet conectate (*dense*), au fiecare câte 4096 neuroni.

Stratul complet conectat opt (final) furnizează 1.000 de neuroni de ieșire (câte unul pentru fiecare clasă) și utilizează funcția softmax pentru calcularea distribuției de probabilitate peste cele 1.000 de clase.

În total, întreaga arhitectură a rețelei AlexNet are aproximativ 659.000 de neuroni și 60 milioane de parametri care trebuie antrenați.

O particularitate a arhitecturii AlexNet, dictată exclusiv de resursele hardware disponibile la momentul respectiv (2012), este separarea pe două „fire” de procesare paralelă pe două GPU nVIDIA GTX 580 cu câte 3GB RAM fiecare. Astfel, începând cu primele 96 de filtre, acestea au fost împărțite în două grupuri de câte 48 fiecare și în mod similar toate filtrele aplicate mai departe în partea convoluțională au fost împărțite în grupe egale pe cele două „fire”. Cu excepția celui de-al treilea strat convoluțional care este conectat la ambele „fire”, conexiunile între straturile convoluționale păstrează „firul” de execuție. Straturile finale complet conectate au (și ele) conexiuni de la ambele „fire”. Antrenarea utilizând cele două GPU a durat 5-6 zile (2012).

Filtrele de convoluție odată „învățate” (prin ajustarea ponderilor) asigură extragerea automată a caracteristicilor specifice, începând cu unele elementare, extrem de simple în primul strat convoluțional și ajungând, prin agregarea acestora strat după strat, la caracteristici complexe reprezentând părți din, sau chiar obiecte întregi.



Figura 12. Nucleele/filtrele convoluționale învățate în primul strat convoluțional pe cele două „fire” de procesare paralelă pe câte unul dintre cele două GPU (Sursa: Krizhevsky, 2012 [29])

Astfel, în primul strat convoluțional de exemplu se observă formarea unei varietăți de filtre elementare, selective după frecvență și respectiv după orientare, precum și a unei varietăți de filtre bazate pe culoare.

Demnă de remarcat este „specializarea” pe fiecare dintre cele două GPU, rezultat al conectivității restricționate pe cele două fire ale arhitecturii. Nucleele corespunzătoare GPU 1 sunt în majoritate independente de culoare, pe când cele corespunzătoare GPU 2 sunt majoritar cu specific de culoare. Acest tip de specializare apare la fiecare rulare și este independent de o inițializare aleatoare a ponderilor sau numerotare a GPU.

Unul dintre principalele elemente de noutate introduse de AlexNet a fost utilizarea funcției ReLU (*Rectified Linear Unit*) de activare a neuronilor din straturile convoluționale. Aceasta a contribuit substanțial la asigurarea evitării efectului de estompare a gradientului la antrenarea cu *backpropagation* de-a lungul a mai multe straturi, precum și a accelerării convergenței și a vitezei în procesul de antrenare. În straturile complet conectate este utilizată funcția tangentă hiperbolică (*tanh*).

Un alt aspect specific important este utilizarea normalizării locale a răspunsului (*local response normalization*), respectiv a valorilor de activare de la ieșirea primelor două straturi convoluționale. Este vorba de o normalizare statistică locală a acestora prin diminuarea lor la o valoare în jurul jumătății, ținându-se cont și de valorile de activare ale neuronilor din aceeași poziție spațială (din aceeași coloană pe adâncime) din două „felii” vecine anterioare și ulterioare „feliei” curente. În acest mod, conform autorilor, s-a creat „o competiție între activările mari de la ieșirile neuronilor calculate pentru diferite filtre/nuclee”. Utilizarea acestui tip de normalizare locală (înlocuită în abordările actuale cu normalizarea pe loturi – *batch normalization*) a adus la acel moment o îmbunătățire a ratei de eroare cu 1,2-1,4%. Problema *overfitting*-ului, inevitabilă la antrenare în cazul unui număr atât de mare de parametri, chiar și cu un set de date de intrare compus din cele circa 1,2 milioane de imagini ImageNet, este rezolvată cu succes prin utilizarea unei metode duale de extindere a setului de date de intrare (*data augmentation*) și de asemenea, prin utilizarea unei tehnici noi, denumită *dropout* (Hinton, 2012 [21]; Srivastava, 2014 [42]) în primele două straturi finale complet conectate.

Prima formă de extindere a setului de date constă în generarea de noi imagini obținute prin translații și reflexii orizontale ale unor regiuni aleatoare din imaginile din setul de antrenare. Aceasta se realizează prin extragerea aleatoare a unor porțiuni 224×224 (și generarea și a reflexiei orizontale a acestora) din imaginile 256×256 din setul de antrenare, etichetate la fel cu imaginea din care au provenit și învățarea rețelei utilizându-se aceste fragmente. Se obține în acest mod o creștere a setului de date de antrenare cu un factor de 2048, chiar dacă exemplele de antrenare astfel obținute sunt în mare parte foarte interdependente. Totodată, la testare, rețeaua face predicții prin extragerea a cinci fragmente 224×224 (cele patru colțuri și zona centrală) cu generarea și a reflexiilor orizontale corespunzătoare (deci un total de zece fragmente) și medierea predicțiilor stratului softmax al rețelei pe cele zece fragmente. A doua formă de extindere a setului de date constă în modificarea intensităților în canalele RGB ale imaginilor de antrenare utilizându-se analiza componentelor principale (*Principal Components Analysis* - PCA). Astfel este simulată invarianța identității obiectelor în imaginile naturale la schimbări de culoare și intensitate a iluminării acestora. Prin extinderea setului de date (*data augmentation*), s-a obținut o reducere a ratei de eroare cu peste 1%.

Noua tehnică „*dropout*” utilizată de AlexNet constă în setarea aleatoare la zero a ieșirilor unor neuroni de pe straturile ascunse cu o probabilitate de 0,5 (respectiv jumătate dintre aceștia). Neuronii care sunt astfel „decuplați” nu contribuie la propagarea înainte a semnalului și nici nu participă la ajustarea celorlalte ponderi la antrenarea cu *backpropagation*. În acest mod, pentru fiecare imagine de intrare prezentată, rețeaua oferă o arhitectură diferită, dar toate aceste arhitecturi partajează același set de ponderi. Această tehnică reduce coadaptările complexe ale neuronilor, din moment ce un neuron nu se poate baza pe prezența altor neuroni particulari, fiind astfel forțat să învețe caracteristici mai robuste care sunt utile în conjuncție cu multe subseturi aleatoare de alți neuroni.

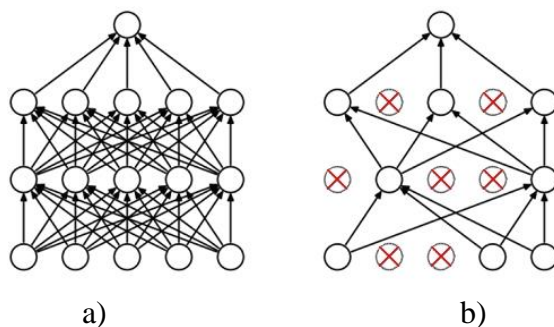


Figura 13. Rețea neuronală complet conectată a) standard; b) cu aplicare „dropout”
(Adaptare după: Srivastava, 2014 [42])

La testare, sunt utilizați toți neuronii arhitecturii complete, fără *dropout*, dar ieșirile acestora sunt multiplicare cu 0,5, ceea ce constituie o aproximare rezonabilă a preluării mediei geometrice a distribuțiilor predictive produse de numărul exponențial mare de rețele *dropout*.

VGGNet (2014)

Pe măsură ce rețelele neuronale convoluționale au evoluat devenind un instrument comun pentru domeniul vederii artificiale, au apărut mai multe încercări de îmbunătățire a arhitecturii AlexNet originale (Krizhevsky, 2012 [29], cu scopul obținerii unei acurateți (chiar) mai bune. De exemplu, câștigătorii competiției ILSVRC-2013 (Zeiler, 2014 [45]; Sermanet, 2014 [40]) au utilizat câmpuri receptoare de dimensiune mai mică, precum și pași mai mici în primul strat convoluțional. O altă direcție de îmbunătățire a fost aceea de a se antrena și testa rețelele dens pe întreaga imagine și la mai multe scale (Sermanet, 2014[40]; Howard, 2014 [23]).

Una dintre arhitecturile câștigătoare ale competiției ILSVRC 2014 (Simonyan, 2015 [41]), ulterior referită ca VGGNet, a adresat un alt aspect important și anume adâncimea rețelei convoluționale, prin adăugarea succesivă de noi straturi în timp ce ceilalți parametri ai acesteia au fost menținuți. Aceasta s-a putut realiza prin utilizarea de filtre de convoluție de dimensiune foarte mică (3×3) în toate straturile. Rețeaua VGGNet de referință constă în 16 (VGG16) sau respectiv 19 (VGG19) straturi, dintre care primele 13, respectiv 16, sunt convoluționale (grupate în blocuri succesive, fiecare terminat cu un strat de agregare de tip *max pooling*) și ultimele 3 sunt complet conectate, fiind este extrem de atractivă datorită arhitecturii sale foarte uniforme, cu puțini hiperparametri. În mare măsură VGGNet poate fi considerată similară cu AlexNet pe un singur „fir”, cu convoluții exclusiv 3×3 structurate în straturi cu secvențe de mai multe astfel de filtre consecutive per strat convoluțional.

Se poate ușor observa că unei secvențe de două straturi cu filtre convoluționale 3×3 consecutive, fiecare cu pas 1, fără agregare între ele, îi corespunde un câmp receptor 5×5 , iar unei secvențe compuse din trei astfel de convoluții îi corespunde un câmp receptor efectiv 7×7 . Prin utilizarea unei astfel de secvențe de două sau trei convoluții 3×3 în locul uneia singure 5×5 sau respectiv 7×7 , pe de o parte sunt introduse două, respectiv trei neliniarități ReLU în locul uneia singure, ceea ce conduce la o funcție de decizie mai discriminativă, iar pe de altă parte se obține totodată și o scădere importantă a numărului de parametri. De exemplu, pentru C canale de culoare, un filtru/nucleu format din trei straturi de convoluție 3×3 conține un total de $3 \times 3^2 \times C^2 = 27C^2$ ponderi, în timp ce un singur strat de convoluție 7×7 conține $1 \times 7^2 \times C^2 = 49C^2$ ponderi, adică cu 81% mai multe. Aceasta poate fi privită ca o regularizare pe filtrele convoluționale 7×7 , prin descompunerea acestora în secvența de trei filtre 3×3 cu neliniaritățile inserate între ele.

Astfel, în locul utilizării unor câmpuri receptoare relativ mari în straturile convoluționale (de exemplu 11×11 cu pas 4 în Krizhevsky, 2012 [29], sau 7×7 cu pas 2 în Zeiler, 2014 [45]; Sermanet, 2014 [40]), au fost utilizate câmpuri receptoare (filtre) foarte mici (3×3) în toată rețeaua, convoluțiile fiind aplicate asupra intrărilor cu pas (*stride*) 1.

Ca și AlexNet, VGGNet utilizează la intrare imagini RGB de dimensiune constantă, fixă, 224×224 , dar diferă modul de obținere a acestora din imagini cu diferite rezoluții. De asemenea, ca și la Alexnet, singura preprocesare aplicată este normalizarea prin scăderea valorilor medii obținute pe întreg setul de antrenare din valorile tuturor pixelilor pentru fiecare canal. Imaginea 224×224 aplicată la intrare este trecută inițial printr-o succesiune de straturi convoluționale în care sunt utilizate filtre cu câmpuri receptoare foarte mici 3×3 (dimensiunea minimă pentru care se poate vorbi despre un centru și vecinătăți stânga-dreapta și sus-jos)). Ca o paranteză, menționăm că a fost realizată și testată inclusiv o arhitectură în care au fost utilizate de asemenea filtre de convoluție 1×1 , care pot fi privite ca o transformare liniară a canalelor de intrare (urmată de neliniaritate). Pasul utilizat este întotdeauna 1, iar mărimea *padding*-ului spațial este 1, astfel încât să se păstreze rezoluția spațială (dimensiunile spațiale) în urma aplicării convoluției.

Straturile convoluționale sunt grupate – atât pentru VGG16 cât și pentru VGG19 – în cinci blocuri/module succesive. Fiecare dintre acestea din urmă se termină cu câte un strat de agregare de tip *max pooling* utilizând o fereastră 2×2 cu pas (stride) 2, fără suprapunere deci, care realizează și reducerea dimensională prin înjumătățirea fiecărei dimensiuni spațiale, de la 224×224 inițial la 7×7 la ieșirea ultimului strat de max pooling din ultimul bloc. Trebuie subliniat faptul că nu există astfel de straturi de agregare între straturile convoluționale din cadrul fiecărui bloc. Numărul de straturi convoluționale consecutive în fiecare din primele două blocuri/module este același (două), în timp ce în ultimele trei blocuri/module există secvențe de câte 3, respectiv 4 straturi convoluționale în cazul VGG16, respectiv VGG19. Numărul de filtre din fiecare strat convoluțional crește (se dublează) de la bloc la bloc până la cel de-al patrulea inclusiv, de la 64 la 128, 256 și respectiv 512, în cel de-al cincilea rămânând constant (512).

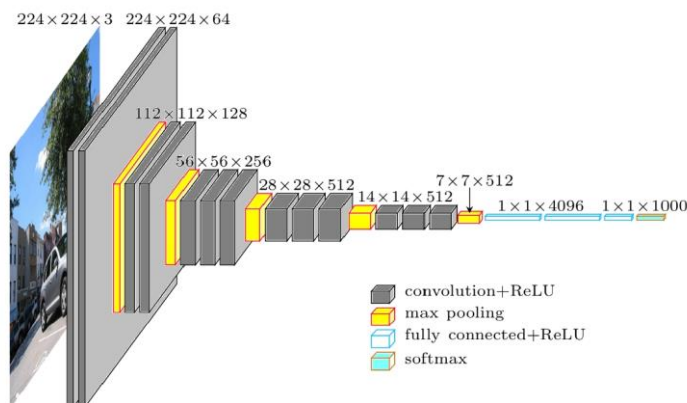


Figura 14. Macroarhitectura VGG16 (Adaptare după: Cord, 2016 [4])

Ieșirea succesiunii de straturi convoluționale astfel organizate este aplicată ca intrare părții de clasificare finală, formată din trei straturi complet conectate similar cu cele din arhitectura AlexNet. Primele două dintre acestea au câte 4.096 canale fiecare, iar cel de-al treilea conține 1.000 canale și realizează efectiv clasificarea în cele 1000 clase ILSVRC. Ieșirea finală este calculată ca probabilități peste acestea utilizând funcția softmax. Această configurație a straturilor complet conectate este aceeași în cazul ambelor rețele (VGG16 și VGG19).

Spre deosebire de AlexNet, funcția neliniară de activare ReLU (*Rectified Linear Unit*) este utilizată în toate straturile inclusiv în cele complet conectate, nu numai în cele convoluționale. În același timp, pentru niciunul dintre acestea din urmă nu se efectuează nicio normalizare locală (*Local Response Normalization*), deoarece s-a constatat că nu aduce niciun fel de îmbunătățire a performanței, producând însă creșterea consumului de memorie și a duratei de calcul.

Se poate observa că numărul de parametri per strat crește continuu spre straturile finale (unde sunt „concentrați” aproximativ 90% dintre aceștia), cu un maxim în primul dintre acestea, în timp ce memoria necesară per strat este maximă în primele straturi convoluționale, scăzând continuu spre straturile finale complet conectate. În total, arhitectura VGGNet conține aproximativ 138,3, respectiv 143,6 milioane de parametri și 13,5, respectiv 14,8 milioane de neuroni (VGG16 vs. VGG19).

Procedura de antrenare a VGG Net urmărește în mare cea utilizată pentru AlexNet, cu diferența că fragmentele de imagini utilizate la intrare sunt decupate din imagini de antrenare multiscală. Învățarea este realizată prin optimizarea funcției obiectiv de regresie logistică polinomială utilizându-se metoda gradientului pe mini loturi (bazată pe *backpropagation* (LeCun, 1998 [31]) cu momente. Sunt utilizate loturi de 256 de imagini de intrare și moment 0,9. Antrenarea este regularizată cu *weight decay* și *dropout* cu probabilitate 0,5 în primele două straturi complet conectate. Rata de învățare este inițializată la 10^{-2} , apoi redusă de 10 ori când acuratețea pe setul de validare nu se mai îmbunătățește. În total, rata de învățare a fost redusă astfel de 3 ori, iar învățarea

a fost oprită după 370.000 de iterații (74 de epoci). Cu toate că are un număr mai mare de parametri și o adâncime mai mare (mai multe straturi) în comparație cu AlexNet, VGGNet necesită mai puține epoci pentru a converge datorită regularizării implicite produse de adâncimea mai mare și dimensiunea mai mică a filtrelor de convoluție, precum și datorită preinițializării unora dintre straturi – primele 4 convoluționale și ultimele 3 complet conectate – cu valori generate anterior la antrenarea unei variante de rețea VGG mai puțin adâncă (11 straturi). Ulterior, s-a demonstrat că ponderile pot fi inițializate și fără preantrenare, utilizându-se o procedură de inițializare aleatoare (Glorot, 2010 [13]). Inițial (2014), rețeaua VGGNet a fost antrenată pe 4 GPU nVIDIA GTX Titan Black cu câte 6GB RAM fiecare, timp de 2-3 săptămâni, funcție de varianta de arhitectură.

Fragmentele de dimensiune fixă 224×224 de imagini utilizate ca intrări sunt decupate aleator din imaginile de antrenare rescalate (o astfel de decupare per imagine per iterație cu metoda gradientului). Se poate utiliza antrenarea mono-scală (cu latura mică a imaginii scalate de 256 de pixeli, analog cu Krizhevsky, 2012 [29]; Zeiler, 2014 [45]; Sermanet, 2014 [40], dar și multi-scală, cu latura stabilită aleator într-un anumit interval (de exemplu [256÷512]). În cazul al doilea, s-a utilizat preinițializarea ponderilor cu valori precalculate mono-scală pentru o latură de 384 de pixeli. Toate acestea pot fi considerate ca o extindere a setului de date de antrenare (*data augmentation*). De asemenea, (tot) pentru extinderea setului de date de intrare, bucățile decupate sunt supuse aleator la oglindire orizontală și modificare a culorii, analog cu Krizhevsky, 2012 [29].

Pentru testare se utilizează de asemenea rescalarea imaginilor utilizate (la o scală predefinită, care poate fi diferită de cea utilizată la antrenare). Apoi, rețeaua este aplicată dens pe imaginea de test rescalată, similar cu Sermanet, 2014 [40]. Astfel, straturile complet conectate sunt mai întâi convertite în straturi convoluționale (primul cu filtre 7×7 , iar ultimele două cu filtre 1×1). Apoi, rețeaua complet convoluțională astfel obținută este aplicată pe întreaga imagine nedecupată. Rezultatul este o hartă de scoruri per clasă cu un număr de canale egal cu numărul de clase și rezoluție spațială variabilă, dependentă de dimensiunea imaginii de intrare.

În final, pentru a se obține un vector de dimensiune fixă cu scoruri per clasă pentru imagine, harta respectivă este mediată spațial (agregată sumativ). De asemenea, setul de imagini de test este (și el) extins prin oglindirea pe orizontală a imaginilor. Scorurile finale pentru imagine sunt obținute prin medierea rezultatelor aplicării funcției softmax în cele două situații (imagine originală și respectiv oglindită).

Astfel, aplicând rețeaua complet convoluțională pe întreaga imagine nu mai este necesară decuparea aleatoare a mai multor fragmente din aceasta la testare (ca în cazul AlexNet – Krizhevsky, 2012 [29]), care este mai puțin eficientă, necesitând recalculări pentru fiecare fragment decupat. Pe de altă parte, în cazul utilizării unui set mare de fragmente decupate (ca în Szegedy, 2014 [43]) se poate obține o acuratețe îmbunătățită datorită eșantionării mai fine decât în cazul rețelei complet convoluționale.

De asemenea, evaluarea pe fragmente multiple este complementară evaluării dense (și) datorită condițiilor de frontieră pentru convoluții: în cazul fragmentelor, *padding*-ul se face cu valori de zero, în timp ce în cazul evaluării dense *padding*-ul corespunzător unui același fragment apare natural din vecinătățile fragmentului din imaginea respectivă, ceea ce mărește substanțial câmpul receptor al întregii rețele, astfel încât este captat mai mult context. Totuși, în practică, timpul de calcul suplimentar în cazul fragmentelor multiple nu justifică potențialul câștig în acuratețe.

S-a constatat că, din punct de vedere al acurateții, cele două arhitecturi cu 16 și respectiv 19 straturi sunt comparabile, creșterea numărului de straturi ne(mai)aducând o creștere a acesteia.

Ca și AlexNet, VGGNet este una dintre cele mai utilizate variante pentru extragerea automată a caracteristicilor din imagini. Configurația ponderilor atât pentru VGG16 cât și pentru VGG19 este disponibilă public, fiind astfel adesea utilizată în multiple alte aplicații pentru extragerea primară a caracteristicilor și/sau *transfer learning*.

Alte arhitecturi neuronale convoluționale de referință, mai complexe/sofisticate, cu (și) mai multe straturi, dar totuși cu un număr de parametri mult redus ca ordin de mărime și complexitate de calcul abordabilă, cu performanțe chiar mai bune (menționate aici mai sus, dar și în secțiunea 3):

- GoogLeNet – Szegedy, 2014 [43], cu 22 straturi și aprox. 4 milioane de parametri; câștigătoare ILSVRC 2014; utilizează NIN (*Network in Network*); secvență de module Inception (fiecare cu căi paralele cu filtre având câmpuri receptoare de dimensiuni diferite, concatenate la ieșire); sunt utilizate convoluții 1×1 pentru reducerea dimensionalității înaintea convoluțiilor costisitoare;
- ResNet (*Residual Network*) – He, 2015 [18], cu 152 straturi și sub 4 milioane de parametri; câștigătoare ILSVRC 2015; secvență de module reziduale (fiecare cu minim două straturi convoluționale și conexiune suplimentară a intrării stratului inițial direct la ieșirea stratului final al modulului unde se combină respectiva intrare cu ieșirea convoluțională), fără straturi finale complet conectate;
- DenseNet (*Densely connected convolutional network*) – Huang, 2016 [25], variantă la ResNet cu o ușoară îmbunătățire;
- SENet (*Squeeze-and-Excitation Networks*) – Hu, 2017 [24], *wrapper* peste alte arhitecturi (ResNet, Inception etc.); câștigătoare ILSVRC 2017.

Adesea în practică, pentru obținerea unor rezultate superioare în clasificare este utilizat un ansamblu de mai multe arhitecturi (*ensemble methods*) – variante ale aceluiași model RNC sau chiar modele diferite – în paralel, combinându-se răspunsul acestora prin mediere, ceea ce permite valorificarea principului că modele diferite au sensibilități și cauze potențial generatoare de eroare diferite.

6. Concluzii

Rețelele neuronale convoluționale (RNC) au ajuns astăzi practic standardul în aplicațiile bazate pe analiza de imagini, performanțele lor egalând și chiar depășind performanța umană în domeniu. Acest fapt se datorează pe de o parte inovării și îmbunătățirilor în zona modelelor (de ex. utilizarea ReLU) și algoritmilor *deep learning* și evoluției tehnologice în ceea ce privește puterea și capacitatea de stocare și procesare (inclusiv apariția și dezvoltarea procesoarelor grafice GPU), iar pe de altă parte, disponibilității de volume uriașe de date de imagini (*big data*), capabile să surprindă complexitatea lumii înconjurătoare. În acest sens, crearea setului de imagini ImageNet (2009) și competiția asociată ILSVRC (2010-2017) a avut un important rol de catalizator. RNC elimină necesitatea extragerii manuale a caracteristicilor, acestea fiind învățate în mod automat direct din datele de antrenare. Ele pot fi reînvățate pentru noi sarcini de recunoaștere și permit construcții ulterioare pe baza unor (părți inițiale din) RNC preantrenate (*transfer learning*).

În general dimensiunea imaginilor aplicate la intrarea RNC trebuie să fie multiplu de puteri (suficient de mari) ale lui 2, pentru a permite reducerile dimensionale succesive necesare de-a lungul straturilor RNC. Valorile cele mai utilizate sunt: 32 (CIFAR-10), 64, 96 (STL-10), sau 224 (ImageNet), 384, sau 512.

Este bine ca straturile convoluționale să utilizeze filtre de dimensiuni mici (3×3 sau cel mult 5×5) și pas (*stride*) 1, precum și *padding* cu zero al volumelor de intrare astfel încât stratul convoluțional să nu altereze dimensiunile spațiale ale intrării. În cazul necesității utilizării (și a) unor filtre de dimensiune mai mare (7×7 sau mai mult), acestea sunt aplicate în general în primul strat convoluțional al RNC care are ca intrare imaginea originală.

Straturile de agregare sunt răspunzătoare de reducerea dimensională spațială a intrării. Cel mai adesea se utilizează *max pooling* cu câmpuri receptoare 2×2 și pas (*stride*) 2, astfel eliminându-se 75% din activările din volumul de intrare (datorită înjumătățirii atât a lățimii cât și a înălțimii). Mai puțin utilizate sunt câmpurile receptoare 3×3 cu pas (*stride*) 2, dimensiuni mai mari nefiind practic utilizate datorită pierderii prea abrupte și timpurii de rezoluție ceea ce conduce la performanțe mai slabe.

Un argument în plus pentru utilizarea pasului 1 în straturile convoluționale este acela că în practică s-a constatat că utilizându-se pași mici se obțin rezultate mai bune, pe lângă faptul că – așa cum am mai menționat – pasul 1 lasă problema reducerii dimensiunilor spațiale exclusiv straturilor de agregare, cele convoluționale realizând doar transformarea adâncimii volumului de intrare (funcție de numărul filtrelor aplicate). De asemenea, utilizarea *padding*-ului cu zero, pe lângă beneficiul păstrării dimensiunilor spațiale după convoluții, îmbunătățește performanța. În cazul neaplicării *padding*-ului cu zero a intrării și efectuării convoluțiilor posibile, volumele sunt reduse puțin după fiecare strat convoluțional, informația de la margini fiind pierdută prea rapid.

În unele situații, datorită constrângerilor legate de memorie, este posibil să fie necesare compromisuri în ceea ce privește cele expuse mai sus. Astfel, mai ales în cazul primelor arhitecturi RNC, necesarul de memorie poate crește foarte repede. De exemplu, dacă se aplică 3 straturi convoluționale cu câte 64 de filtre 3×3 și pas 1 fiecare pe o imagine $224 \times 224 \times 3$ se generează 3 volume de activări de dimensiune $224 \times 224 \times 64$ fiecare. Aceasta înseamnă un total de circa 10 milioane de activări, sau 72MB de memorie (per imagine, pentru activări și gradienti). Cum plăcile GPU sunt în general destul de limitate ca memorie, poate apărea necesitatea unor compromisuri. În practică, este preferat ca acestea să vizeze (exclusiv) primul strat convoluțional al RNC. De exemplu, în cazul ZFNet [45] sunt utilizate în primul strat convoluțional filtre 7×7 cu pas 2, iar în cazul AlexNet [29] sunt utilizate filtre 11×11 cu pas 4.

În locul aplicării unui strat convoluțional cu filtre de dimensiune mai mare se poate aplica succesiv, cu același rezultat, o succesiune de straturi convoluționale cu filtre mai mici, dar cu un număr total mai mic de parametri (de ex. 1 strat 5×5 , respectiv $7 \times 7 \equiv 2$, respectiv 3 straturi 3×3).

Stratul/straturile finale complet conectate ale unei RNC pot fi înlocuite la clasificare, cu același rezultat, tot cu straturi convoluționale (cu filtre 1×1), urmate de un strat de deconvoluție, sau convoluție transpusă (*fully convolutional network – FCN*) și utilizându-se metode de tip *skip*, prin care valori de activare din straturi intermediare cu rezoluție mai mare sunt combinate cu cele din ultimul strat. Astfel pot fi suportate orice dimensiuni ale imaginii de intrare a RNC.

În practică sunt aplicate o serie de tehnici și metode de optimizare a performanțelor RNC, cum sunt:

- utilizarea convoluțiilor 1×1 pentru reducerea și expandarea judicioasă a numărului hărților de caracteristici (*feature maps*);
- utilizarea unor tehnici de evitare a unor conexiuni (*skip*) și/sau de creare a unor căi multiple de-a lungul rețelei;
- utilizarea unor metode eficiente de inițializare a ponderilor, de regularizare și/sau normalizare (de ex. *dropout*, *batch normalization*), respectiv de extindere a seturilor de date de antrenare / învățare (*data augmentation*);
- medierea ieșirilor furnizate de clasificator pentru mai multe fragmente decupate / oglindite din imaginea originală;
- utilizarea ansamblurilor de rețele (*ensembles of networks*).

Mențiuni

Prezenta lucrare are la bază parte din activitățile și rezultatele fazei I a proiectului PN 1819-0201 derulat la ICI București (2018) în cadrul Programului național nucleu „RESINFO-TD” finanțat de Ministerul Cercetării și Inovării.

BIBLIOGRAFIE

1. Bengio, Y. and LeCun, Y. (2007). Scaling learning algorithms towards AI. *Large Scale Kernel Machines*;
2. Ciresan, D. C.; Meier, U.; Gambardella, L. M. and Schmidhuber, J. (2010). Deep, big, simple neural nets for handwritten digit recognition. *Neural Computation*, 22(12):3207–3220;
3. Ciresan, D. C.; Meier, U.; Masci, J.; Gambardella, L. M. and Schmidhuber, J. (2011). Flexible, high performance convolutional neural networks for image classification. Proc. International Joint Conference on Artificial Intelligence - IJCAI'2011, 1237–1242;
4. Cord, M. (2016). Deep CNN and Weak Supervision Learning for visual recognition. *Report of the Heuritech Deep Learning Meetup #5*, 26 feb. <https://blog.heuritech.com/2016/02/29/a-brief-report-of-the-heuritech-deep-learning-meetup-5/> (accesat 2018);
5. Cybenko, G. (1989). Approximations by superpositions of sigmoidal functions, *Mathematics of Control, Signals, and Systems*, 2(4), 303-314. doi:10.1007/BF02551274;
6. Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K. and Li, F.-F. (2009). ImageNet: A Large-Scale Hierarchical Image Database. *Proc. of IEEE Conference on Computer Vision and Pattern Recognition - CVPR 2009*. doi: 248-255. 10.1109/CVPR.2009.5206848;
7. Fukushima, K. (1975). Cognitron: A self-organizing multilayered neural network. *Biological Cybernetics*, 20(3-4), 121-136 (Sept.);
8. Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4), 193-202 (April);
9. Fukushima, K. (1987). Neural network model for selective attention in visual pattern recognition and associative recall. *Applied Optics*, 26(23), pp. 4985-4992 (Dec.);
10. Fukushima, K. (2005). Restoring partly occluded patterns: a neural network model. *Neural Networks*, 18(1), 33-43;
11. Fukushima, K. (2007). Neocognitron. *Scholarpedia* (online) 2(1):1717. doi:10.4249/scholarpedia.1717. <http://www.scholarpedia.org/article/Neocognitron> (accesat 2018);
12. Gershgorn, D. (2017). It's not about the algorithm. The data that transformed AI research – and possibly the world. *Quartz* (online), July 26. <https://qz.com/1034972/the-data-that-changed-the-direction-of-ai-research-and-possibly-the-world/> (accesat 2018);
13. Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *Proc. of 13th International Conference on Artificial Intelligence and Statistics – AISTATS 2010. JMLR Vol.9: PMLR 9: 249–256*;
14. Glorot, X.; Bordes, A. and Bengio, Y. (2011). Deep sparse rectifier neural networks. *Proc. of 14th International Conference on Artificial Intelligence and Statistics – AISTATS 2011. JMLR Vol.15: PMLR 15: 315-323*;
15. Goodfellow, I.; Bengio, Y. and Courville, A. (2016). Deep Learning. *MIT Press*;
16. Hahnloser, R.; Sarpeshkar, R.; Mahowald, M. A.; Douglas, R. J. and Seung, H. S. (2000). Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*. 405: 947–951. doi:10.1038/35016072;
17. Hahnloser, R and Seung, H.S. (2001). Permitted and Forbidden Sets in Symmetric Threshold-Linear Networks. *NIPS 2001*;
18. He, K.; Zhang, X.; Ren, S. and Sun, J. (2015). Deep Residual Learning for Image Recognition. *Preprint arXiv:1512.03385*;

19. Hinton, G. E. and Salakhutdinov, R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786), 504–507;
20. Hinton, G. E., Osindero, S., and Teh, Y. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18, 1527–1554;
21. Hinton, G.E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I. and Salakhutdinov, R.R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *Preprint arXiv:1207.0580*;
22. Hornik, K. (1991). Approximation Capabilities of Multilayer Feedforward Networks. *Neural Networks*, 4(2), 251–257. doi:10.1016/0893-6080(91)90009-T;
23. Howard, A. G. (2013). Some improvements on deep convolutional neural network based image classification. (*Preprint ArXiv: 1312.5402*). *Proc. ICLR 2014*;
24. Hu, J.; Shen, L.; Albanie, S.; Sun, G. and Wu, E. (2017). Squeeze-and-Excitation Networks. *arXiv: 1709.01507, Sep.*;
25. Huang, G.; Zhuang, L.; Van der Maaten, L. and Weinberger, K. Q. (2016). Densely Connected Convolutional Networks. *arXiv: 1608.06993*, Aug.;
26. Hubel, D. H.; Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. *J. Physiol.* 148(3):574–91, Oct. doi:10.1113/jphysiol.1959.sp006308;
27. Karpathy, A. and Li, F.–F. (2015). *Convolutional Neural Networks for Visual Recognition*. Stanford CS Class (CS231n): <http://cs231n.github.io/convolutional-networks/> (accessed 2018);
28. Kolmogorov, A. N. (1957). On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. *Dokl. Akad. Nauk SSSR*, 114(5), 953–956;
29. Krizhevsky, A.; Sutskever, I. and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *NIPS 2012*, pp. 1106–1114;
30. LeCun, Y.; Boser, B.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W. and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551;
31. LeCun, Z.; Bottou, L.; Bengio, Y. and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proc. of the IEEE*, 86(11):2278–2324. doi:10.1109/5.726791;
32. Martens, J. (2010). Deep learning via Hessian-free optimization. *Proc. of the 27th International Conference on Machine Learning, ICML-10*, Haifa, Israel, Jun 21-24, 735-742;
33. McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, Vol. 5(4), 115-133;
34. Miller, G. (1995). WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11), Nov., 39-41;
35. Nair, V. and Hinton, G. (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. *Proc. of the 27th International Conference on Machine Learning, ICML-10*, Haifa, Israel, Jun 21-24, 807-814;
36. Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review* 65: 386-408;
37. Rosenblatt, F. (1962). *Principles of neurodynamics; perceptrons and the theory of brain mechanisms*. Washington D.C., Spartan Books;
38. Rumelhart, D. E.; McClelland, J. L. & Group, P. R. (1987). *Parallel distributed processing* (Vol. 1, p. 184). Cambridge, MA: MIT press;

39. Saxena, A. (2016). Convolutional Neural Networks (CNNs): An Illustrated Explanation. *XRDS Crossroads - The ACM Magazine for Students*. BLOG (Jun.). <https://blog.xrds.acm.org/2016/06/convolutional-neural-networks-cnns-illustrated-explanation/> (accesat 2018);
40. Sermanet, P.; Eigen, D.; Zhang, X.; Mathieu, M.; Fergus, R. and LeCun, Y. (2014). OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. *Proc. ICLR 2014*;
41. Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition, *Proc. ICLR 2015*;
42. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I. and Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research (JMLR)*, 15(Jun): 1929–1958;
43. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V. and Rabinovich, A. (2014). Going deeper with convolutions. *CoRR*, abs/1409.4842;
44. Vryniotis, V. (2013). Tuning the learning rate in Gradient Descent. DatumBox. BLOG (Oct.). <http://blog.datumbox.com/tuning-the-learning-rate-in-gradient-descent/> (accesat 2018);
45. Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013. Published in *Proc. ECCV 2014*;
46. Zhang, Y. (2018). Lecture notes slide, DeepLearn2018 Summer School, Genoa, July 23-27. <http://grammars.grlmc.com/DeepLearn2018/> (accesat 2018);
47. Werbos, P. B. R. (1974). *New Tools for Prediction and Analysis in the Behavioral Sciences*. (PhD Thesis), Harvard University;
48. *** - ImageNet, 2018: <http://image-net.org/> (accesat 2018);
49. *** - Partnership on AI, 2018: <http://www.partnershiponai.org> (accesat 2018).



Mihnea Horia VREJOIU este cercetător științific gradul III în Departamentul „*Sisteme inteligente distribuite intensive ca date*” din ICI București. Domeniile și subiectele sale de expertiză și interes cuprind: vedere artificială (prelucrare și analiză de imagini, recunoașterea formelor, recunoașterea optică de caractere – OCR, recunoașterea numerelor de înmatriculare – LPR) și învățare automată (clasificatoare, memorii asociative).

Mihnea Horia VREJOIU is a 3rd degree scientific researcher in the „*Distributed and Data Intensive Intelligent Systems*” Department at ICI Bucharest. His main areas and topics of expertise and interest cover: Artificial Vision (Image Processing and Analysis, Pattern Recognition, Optical Character Recognition – OCR, License Plate Recognition – LPR) and Machine Learning (classifiers, associative memories).