Reţele neuronale convoluţionale, Big Data şi Deep Learning în analiza automată de imagini

Mihnea Horia VREJOIU
Institutul Naţional de Cercetare-Dezvoltare în Informatică – ICI Bucureşti
B-dul Mareşal Alexandru Averescu, Nr. 8-10, 011455, Bucureşti, România
mihnea.vrejoiu@ici.ro

Rezumat: În ultimii ani se vorbeşte din ce în ce mai mult despre inteligenţa artificială. Ceea ce stă de fapt în spatele inteligenţei artificiale astăzi poate fi rezumat pe scurt prin sintagma „reţele neuronale artificiale“, la care mai nou a fost adăugat adjectivul „adânci“. Aplicaţiile bazate pe acestea au ajuns să egaleze şi chiar să depăşească performanţa umană în multe domenii. Unul dintre primele domenii în care au fost dezvoltate şi în care au căpătat o largă răspândire este cel al vederii artificiale, respectiv al recunoaşterii / clasificării de imagini. Fără pretenţia de a acoperi complet subiectul, ne propunem în lucrarea de faţă o trecere în revistă, încercând să surprindem cât mai intuitiv posibil câteva elemente şi repere esenţiale asupra istoriei şi evoluţiei reţelelor neuronale artificiale, cu noua perspectivă oferită în ultima perioadă de disponibilitatea datelor masive (*Big Data*) utilizate în conjuncţie cu acestea, ca factor de importanţă majoră complementar, sinergic şi convergent alături de calitatea şi performanţele algoritmilor *deep learning* implicaţi. De asemenea, analizăm elementele şi mecanismele care definesc şi compun reţelele convoluţionale în general, funcţionarea şi specificul acestora cu aplicare în vederea artificială, precum şi două dintre primele astfel de arhitecturi de referinţă, AlexNet şi VGGNet, cu particularităţile acestora şi tehnici utilizate în procesele de antrenare, validare şi testare.

Cuvinte cheie: reţea neuronală (adâncă), reţea (neuronală) convoluţională, deep learning, big data, vedere artificială, recunoaşterea / 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 şi investiţiile în sisteme cognitive şi inteligenţă artificială au cunoscut o creştere explozivă, care continuă şi va continua, atât în mediul academic, de cercetare ştiinţifică şi dezvoltare tehnologică, precum şi în cel comercial şi de afaceri, de la cele mai mici până la marile corporaţii. Cele mai renumite nume din tehnologie (cum sunt: Google, Facebook, Microsoft, Yahoo,

Baidu, IBM, DropBox, etc.) investesc masiv în inteligenţa artificială pe care o înglobează în propriile modele de afaceri şi o utilizează din ce în ce mai mult în oferta lor: asistenţi virtuali (de ex. Siri), vedere artificială, recunoaşterea vorbirii, traducere automată şi o mulţime de alte aplicaţii. Mai mult, au apărut iniţiative precum consorţiul „Partnership on AI“ [49], care grupează ca parteneri fondatori Amazon, Apple, Deepmind, Facebook, Google, IBM, Microsoft, precum şi alţi aproximativ 80 parteneri din 13 ţări. Toate acestea se bazează astăzi practic pe tehnologia oferită de reţelele neuronale artificiale „adânci“ şi învăţarea profundă (*deep learning*). Unul dintre primele domenii în care acestea au fost dezvoltate şi în care au căpătat o largă răspândire este cel al vederii artificiale, respectiv al recunoaşterii / clasificării de imagini. Fără pretenţia de a acoperi complet subiectul, ne propunem în lucrarea de faţă o trecere în revistă, încercând să surprindem cât mai intuitiv posibil, câteva elemente şi repere esenţiale asupra istoriei şi evoluţiei reţelelor neuronale artificiale, cu noua perspectivă oferită în ultima perioadă de disponibilitatea datelor masive (Big Data) utilizate în conjuncţie cu acestea, ca factor de importanţă majoră complementar, sinergic şi convergent alături de calitatea şi performanţele algoritmilor deep learning implicaţi. În secţiunea următoare a lucrării de faţă ne propunem o trecere în revistă asupra reţelelor neuronale şi a evoluţiei lor de la modelul simplu al perceptronului la reţele de tip perceptron multistrat şi algoritmi de antrenare pentru acestea (*backpropagation* şi metoda gradientului). În secţiunea 3 abordăm noua perspectivă oferită de disponibilitatea datelor masive (*Big Data*), ca factor de importanţă majoră complementar, sinergic şi convergent alături de calitatea şi performanţele algoritmilor *deep learning* implicaţi astăzi în reţelele neuronale adânci, punctând importanţa crucială a creării setului de date ImageNet şi a competiţiei asociate ILSVRC (2010-2017) [48] pentru dezvoltarea acestora în domeniul vederii artificiale. Secţiunea 4 este dedicată reţelelor neuronale convoluţionale, cu prezentarea particularităţilor şi a elementelor componente, iar în secţiunea 5 sunt analizate câteva arhitecturi convoluţionale de referinţă. Secţiunea 6, care încheie lucrarea, sintetizează câteva concluzii.

2. Reţele neuronale artificiale. Perceptronul. Perceptronul multistrat.

Un neuron artificial modelează practic într-un mod simplificat, printr-o funcţie matematică, un neuron dintr-o reţea neuronală biologică (mai mulţi neuroni conectaţi) cum este, de exemplu, creierul uman. Neuronul artificial reprezintă unitatea elementară într-o reţea neuronală artificială. Aşa cum neuronul biologic are dendrite şi axoni, neuronul artificial are o structură arborescentă simplă, cu noduri de intrare şi un nod de ieşire conectat la toate nodurile de intrare. Neuronul artificial primeşte una, sau mai multe valori de intrare, analog potenţialelor excitatoare / inhibitoare postsinaptice aplicate dendritelor neuronului biologic. Acestea sunt sumate, producând la ieşire o valoare de activare, analog al potenţialului de acţiune 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 ieşire printr-o funcţie neliniară, denumită funcţie de activare.

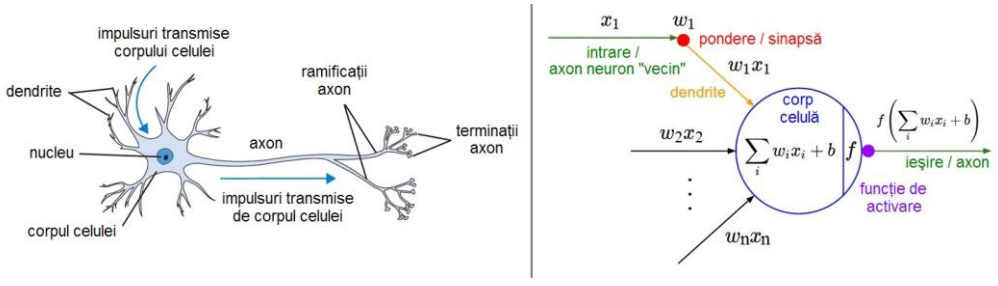


Figura 1. Ilustrarea modelării simplificate a neuronului biologic de către neuronul artificial (Adaptare după: Karpathy, 2015 [27])

Perceptronul

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

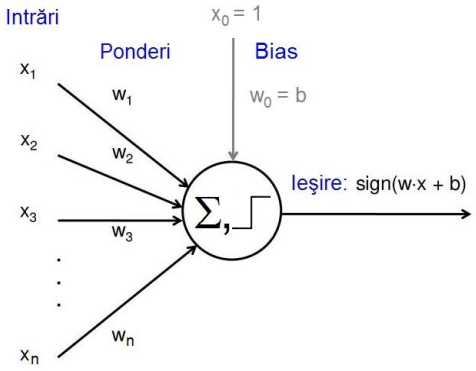


Figura 2. Perceptronul

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

$$y = w \cdot x + b = \Sigma w_i \cdot x_i + b; \quad i = 1 \div n \tag{1}$$

şi apoi, asupra rezultatului numeric astfel obţinut, aplică o funcţie neliniară, de ex. funcţia semn (signum):

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

Termenul liber *b* (*bias / offset*), reprezintă valoarea de prag de activare a perceptronului şi poate fi considerat ca o a *(n+1)*-a intrare, suplimentară, de valoare $x_0 = 1$ şi pondere $w_0 = b$ constante şi permite translatarea funcţiei 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 codificat de exemplu prin valorile 0 şi 1) în cazul datelor liniar separabile aplicate la intrare, prin separarea valorilor de ieşire peste sau sub un anumit prag. Acesta, precum şi 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 depăşit odată cu algoritmul iterativ de adaptare (antrenare) a perceptronului propus de Rosenblatt, 1962 [37] pentru stabilirea valorilor ponderilor. Algoritmul porneşte de la un set de ponderi aleatoare, nenule, mici, comparând în fiecare iteraţie ieşirea obţinută pentru fiecare vector de intrare de antrenare cu clasa reală / corectă căreia îi aparţine şi urmărind ca prin ajustarea ponderilor să se elimine fiecare eventuală eroare de clasificare. Astfel, dacă pentru exemplul curent de instruire ieşirea corectă trebuia să fie 0 dar s-a obţinut 1, atunci se diminuează toate ponderile corespunzătoare intrărilor având valoarea 1, iar pentru situaţia inversă acestea sunt crescute. Dacă la finalul unei iteraţii (după un număr de paşi egal cu numărul vectorilor de instruire) nu s-a (mai) produs nicio modificare a ponderilor, înseamnă că toţi vectorii de instruire respectivi au fost corect clasificaţi, iar vectorul ponderilor la care s-a ajuns reprezintă o soluţie a problemei de instruire respective şi algoritmul se termină cu succes. Dacă însă s-a modificat cel puţin o pondere, se trece la iteraţia următoare. În cazul în care într-un număr *k* prestabilit de iteraţii (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 şi algoritmul este oprit, problema neputând fi rezolvată cu ajutorul perceptronului. Tot Rosenblatt, 1962 [37] a enunţat şi demonstrat „Teorema de convergenţă 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* iteraţii, rezultând un vector al ponderilor $w(n+k) = w(n), \ \forall\, k > 0$, ca soluţie a problemei de instruire.

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

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

Perceptronul multistrat. Antrenarea cu backpropagation

O reţea de perceptroni conectaţi 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 aşa numite straturi ascunse (*hidden layer*) şi un strat de ieşire, conectate secvenţial între ele. Straturile se comportă fiecare analog neuronului biologic. Ieşirile unui strat reprezintă intrări pentru stratul următor. Straturile sunt (în general) complet conectate, fiecare perceptron de pe un strat fiind conectat cu toţi perceptronii de pe stratul anterior (complet conectat). Fiecare conexiune are o pondere individuală proprie. Astfel, vorbim în acest context de modele conecţioniste, ca o a doua paradigmă şi un al doilea val în domeniul reţelelor 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 „reţea neuronală adâncă“ („*deep neural network*“), în timp ce în cazul unui singur strat ascuns vorbim de „reţea neuronală superficială“ („*shallow neural network*“).

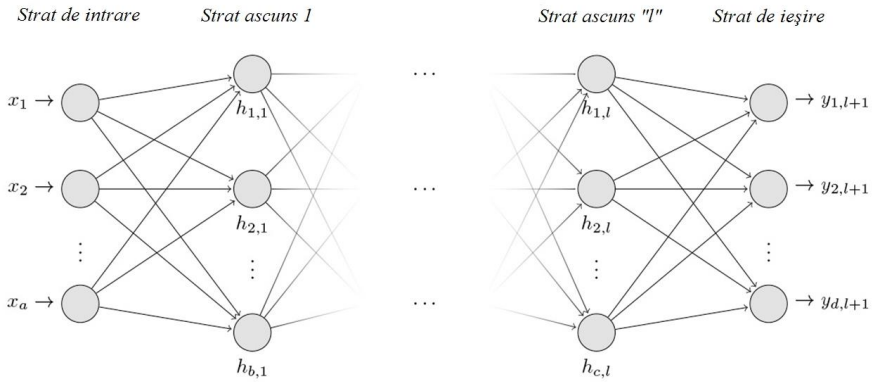


Figura 3. Arhitectura generală a perceptronului multistrat

Reţeaua de tip perceptron multistrat prezentată mai sus mai este de tip *feed-forward*, deoarece legăturile se fac de la intrare spre ieşire, strat după strat, exclusiv spre înainte. Valorile rezultate în straturi superioare nu influenţează în niciun fel valorile de ieşire ale straturilor inferioare. Dacă în cazul perceptronului intrările sunt utilizate pentru a calcula imediat rezultatul, în cazul reţelelor multistrat *feed-forward* se efectuează prelucrări ale datelor secvenţial pe fiecare strat unul după altul. Totodată, se folosesc alte funcţii 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 funcţie de tip „sigmoidă“, cum sunt funcţia logistică:

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

sau tangenta hiperbolică:

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

Reţeaua 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 conţină valori specifice care să „codifice“ caracteristicile relevante pentru identificarea şi 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 şi extragerea caracteristicilor (*feature extraction*) este în general o sarcină netrivială. Datele sunt propagate exclusiv înainte prin reţea, strat după strat, de la intrare spre ieşire. Numărul ieşirilor (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 învăţare supervizată, pe baza unor seturi de vectori de intrare de antrenare/instruire „etichetaţi“ (a căror apartenenţă la o anumită clasă dintre cele posibile este apriori cunoscută). Metoda cea mai utilizată este cea a retropropagării erorii (*backpropagation*), propusă iniţial de Paul Werbos, 1974 [47] în teza sa de doctorat la Harvard. Acesta a descris algoritmul de antrenare supervizată a reţelelor neuronale artificiale multistrat prin propagarea înapoi a erorilor dinspre straturile finale spre straturile inferioare şi 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“ şi utilizat cu succes algoritmul *backpropagation* la antrenarea reţelelor multistrat prin procesare paralelă distribuită (*parallel distributed processing*), făcând ca acesta să devină din acel moment foarte cunoscut şi utilizat. Metoda poate fi rezumată ca un proces continuu de antrenare / învăţare supervizată a reţelei, prin ajustarea şi „acordul fin“ la fiecare iteraţie al structurii acesteia (respectiv a valorilor ponderilor conexiunilor neuronilor săi, care au fost în primă instanţă iniţializate cu valori aleatorii, mici). Este practic vorba despre un proces iterativ de optimizare prin care se urmăreşte diminuarea la fiecare iteraţie a erorii, respectiv a diferenţei între ieşirea aşteptată şi cea obţinută în iteraţia respectivă, estimată utilizându-se o funcţie cost (sau *loss*). În acest mod se potenţează practic acele intrări şi caracteristici care sunt cele mai relevante pentru obţinerea ieşirii 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 şi cât să fie ajustate în sensul urmăririi gradientului descrescător al erorii.

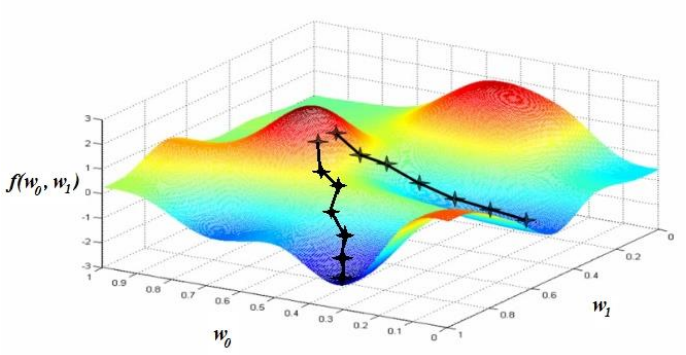


Figura 4. Metoda gradientului în backpropagation
(Adaptare după: Vryniotis, 2013 [44])

Pe măsură ce eroarea scade, fiecare nouă iteraţie devine mai rafinată, procesul putând necesita mii de iteraţii până când ieşirea calculată să se apropie suficient de mult de cea aşteptată, moment în care se poate considera că reţeaua este complet antrenată.

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

Astfel, cu algoritmul de învăţare supervizată utilizând *backpropagation* şi metoda gradientului, reţelele de tip MLP au însemnat un mare pas înainte. Totuşi, trebuie observat faptul că pentru cazul mai multor straturi ascunse, algoritmul de învăţare *backpropagation* bazat pe metoda gradientului are limitări serioase. Aceasta se datorează formei sigmoidale a funcţiei de activare a neuronilor care determină la derivare o saturare rapidă, ducând la efectul numit „dispariţia/ estomparea gradientului“ (*vanishing gradient*) de la strat la strat înapoi. Prin urmare, în cazul reţelelor de tip perceptron multistrat antrenate cu *backpropagation* şi metoda gradientului, numărul de straturi ascunse (adâncimea reţelei) 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 şi a numărului de neuroni din stratul/straturile ascuns/ascunse. Există estimări, de ex. Kolmogorov (1957) [28], care arată că pentru aproximarea unei funcţii de *n* variabile, sunt necesari *n(2n+1)* neuroni în stratul ascuns, iar în cazul utilizării a două straturi ascunse, *(2n+1)* neuroni. Totuşi aceste estimări nu conduc întotdeauna la o soluţie optimă. Mai trebuie făcute observaţiile că numărul de neuroni necesar pe stratul ascuns creşte practic exponenţial cu numărul de intrări, iar antrenarea reţelelor intră în categoria problemelor „NP-complete“ care nu pot fi rezolvate prin aproximări polinomiale în timp. Aceste limitări, precum şi existenţa unor algoritmi şi metode de clasificare cu performanţe suficient de bune mai rapide computaţional (arbori de decizie - 1985, *random forest* - 1995, SVM - 1996) au determinat ca evoluţia pe linia MLP să pară un drum înfundat, iar reţelele neuronale să fie practic lăsate de-o parte pentru o vreme de către cei mai mulţi dintre cercetătorii implicaţi.

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

Totuşi, în anul 2006 apare o schimbare importantă de paradigmă când Geoffrey Hinton [19; 20] introduce câteva idei revoluţionare pentru iniţializarea parametrilor reţelelor neuronale cu valori mai apropiate de valorile optimale utilizând maşini Boltzmann restricţionate (*Restricted Boltzmann Machines* – RBM). Se realizează mai întâi o antrenare nesupervizată strat cu strat a reţelei pentru ca ponderile să ajungă să modeleze structura intimă a datelor. Apoi, ponderile astfel iniţializate sunt ajustate utilizându-se *backpropagation*. Cu o astfel de reţea preantrenată cu RBM s-a obţinut o eroare de circa 1%. pe setul de date MNIST (70.000 de imagini conţinând fiecare câte o cifră, 60.000 de antrenare şi 10.000 de test). În 2007, Yoshua Bengio [1] a introdus în locul RBM *autoencoder*-e, dezvoltate ulterior cu diferite variaţii, cum ar fi *denoising* şi *sparse autoencoder*. În 2010, James Martens [32] a prezentat un alt algoritm pentru găsirea parametrilor, utilizând (şi) derivate de ordinul 2, fără învăţare nesupervizată prealabilă, care obţinea rezultate mai bune decât cele de până atunci. Mai mult, tot în acelaşi an, utilizând direct algoritmul clasic de *back-propagation*, cu o reţea adâncă şi lată, pe GPU, utilizând la antrenare (regiuni din) imagini uşor deformate elastic, fără niciun fel de algoritm ajutător pentru iniţializarea ponderilor, Dan Cireşan [2] a obţinut 0,35% eroare pe setul MNIST. Acelaşi grup, utilizând reţele neuronale convoluţionale şi *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 reţelelor neuronale artificiale, în ultimii ani acestea cunoscând o revigorare şi o dezvoltare impresionantă, potenţată şi de evoluţia tehnologică spectaculoasă în ceea ce priveşte capacităţile şi vitezele de calcul şi stocare, utilizarea procesoarelor grafice (*graphics processing unit* – GPU) pentru calcule paralele cu matrici, dispozitivele de achiziţie de date etc., precum şi disponibilitatea volumelor din ce în ce mai mari şi mai variate de date (*Big Data*) în tot mai multe domenii. Au fost dezvoltate noi modele neuronale şi arhitecturi de reţele neuronale adânci, inspirate fie din neuroştiinţe, fie din raţiuni de inginerie computaţională, tehnologia *deep learning* devenind la ora actuală practic sinonimă cu conceptele de inteligenţă artificială şi învăţare automată (*machine learning*).

Deşi evoluţia tehnologică în privinţa puterii şi capacităţii de calcul şi mai cu seamă apariţia şi dezvoltarea procesoarelor grafice (GPU) puternice a oferit o bază importantă pentru dezvoltarea arhitecturilor şi algoritmilor de reţele neuronale adânci şi obţinerea unor performanţe superioare, acestea nu au putut totuşi face un salt spectaculos până când nu s-a putut dispune şi de seturi de date reale pentru antrenare, validare şi testare suficient de mari, care să surprindă complexitatea şi varietatea din lumea reală. De exemplu, în domeniul vederii artificiale, un moment de referinţă l-a reprezentat crearea ImageNet [48], o colecţie de imagini de înaltă rezoluţie din lumea reală etichetate pe baza dicţionarului 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 sfârşitul anilor 1980, cu mai mult de 155.000 de cuvinte indexate. Această întreprindere a fost demarată în 2007 şi coordonată de Fei-Fei Li, iniţial la Princeton, astăzi conducător de cercetări la Google Cloud, profesor asociat şi director al laboratoarelor de inteligenţă artificială şi respectiv de vedere artificială la Universitatea Stanford. Pentru completarea setului de date iniţial au fost necesari doi ani şi jumătate. Au fost etichetate 3,2 milioane de imagini,

separate în 5.247 de categorii, ordonate în 12 subarbori ca: „mammal“, „vehicle“, „furniture“ ş.a.m.d., respectiv circa 500-600 imagini per nod. Setul de imagini ImageNet a fost făcut public iniţial la conferinţa 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.

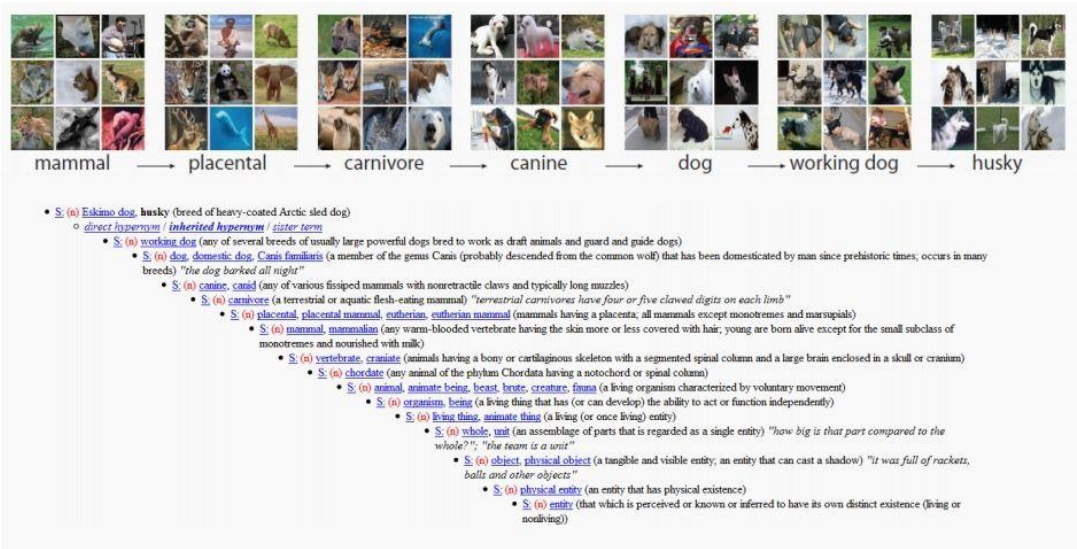


Figura 5. Structura ierarhică a ImageNet pliată pe cea a WordNet (Sursa: Gershgorn, 2017 [12])

ImageNet a devenit una din referinţele cele mai utilizate pe plan mondial pentru măsurarea performanţelor modelelor şi arhitecturilor neuronale convoluţionale în recunoaşterea şi localizarea obiectelor în imagini, precum şi un real catalizator pentru evoluţia explozivă a acestora, inclusiv prin organizarea competiţiei 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 şi 150.000 de imagini de testare, cu rezoluţii diferite, reprezentând aproximativ câte 1.000 de exemple din 1.000 de categorii/clase diferite. Competiţia din 2012 a fost câştigată pentru prima dată de o arhitectură neuronală convoluţională adâncă, denumită AlexNet [29] (SuperVision Group: Geoffrey Hinton, Ilya Sutskever şi Alex Krizhevsky – Universitatea din Toronto), cu 8 straturi, care a produs un salt spectaculos în calitatea clasificării/recunoaşterii cu o rată de eroare de 15,4%, mai bună cu 10,8% decât cea mai bună performanţă anterioară, şi cu 41% mai bună ca următorul clasat. A urmat o explozie a utilizării reţelelor convoluţionale în diferite scopuri (adnotarea automată a fotografiilor, conducerea automată a vehiculelor, recunoaşterea vorbirii, traducerea automată etc.) şi totodată o continuă îmbunătăţire a acestora, ajungându-se la obţinerea de performanţe comparabile cu – şi chiar peste – capabilitatea umană (eroare 5-10%). Matthew D. Zeiler (ulterior fondator al Clarifai) şi Rob Fergus au câştigat competiţia din 2013 cu ZFNet [45] tot cu 8 straturi dar cu o rată de eroare de 11,2%, iar în 2014 câştigători au fost GoogLeNet/Inception-v1 [43] (Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Drago Anguelov, Dumitru Erhan, Andrew Rabinovich – Google) şi VGGNet [41] (Visual Geometry Group – Karen Simonyan şi Andrew Zisserman – Universitatea Oxford, ulterior cooptaţi de Google în laboratorul DeepMind) cu 22, respectiv 16-19 straturi şi rate de eroare de 6,7% şi respectiv 7,3%. Dimensiunile arhitecturilor câştigătoare 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) şi la peste 200 de straturi în 2016 şi 2017 şi rate de eroare de 2,99% şi respectiv de 2,251% (DenseNet [25] şi SENet [24]).

În doar 7 ani, acurateţea câştigătoare în clasificarea obiectelor din setul de date ImageNet a crescut de la 71,8% la peste 97%, depăşind capabilităţile umane (cca. 90-95%) şi dovedind efectiv că date mai multe conduc la decizii mai bune. Este unanim recunoscut astăzi faptul că apariţia ImageNet a contribuit substanţial la schimbarea de paradigmă în abordarea cercetărilor în domeniul inteligenţei

artificiale care a adus ca importanţă datele la acelaşi nivel cu algoritmii. Între timp, odată ce s-a dovedit că algoritmii şi 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 şi Institutul Canadian pentru Cercetări Avansate, ulterior de către Facebook şi Amazon, care şi-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 (şi) faptului că în ultimii ani au devenit disponibile cantităţi uriaşe de date în mai toate domeniile, coroborat cu evoluţia tehnologică în ceea ce priveşte capacitatea de stocare şi gestionare a datelor şi viteza de accesare şi de procesare a acestora odată cu dezvoltarea memoriilor şi suporturilor de stocare rapide şi a noilor generaţii de procesoare multicore şi de procesoare grafice (GPU), care a permis şi dezvoltarea de modele mai mari, cu mai multe straturi şi mai mulţi 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 şi va egala sau surclasa performanţa umană dacă este antrenat pe un set de date de cel puţin 10 milioane de exemple etichetate (Ian Goodfellow, 2016 [15]). Pentru a avea acelaşi succes cu seturi de date mai restrânse, sunt efectuate astăzi cercetări prin care să se valorifice cantităţile mari de date neetichetate cu metode de învăţare nesupervizată sau semisupervizată.

4. Reţele neuronale convoluţionale (RNC)

Un precursor inspiraţional pentru reţelele neuronale convoluţionale 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 învăţare, 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*) şi straturi de celule complexe (C) care permit abateri în poziţionarea/localizarea caracteristicilor.

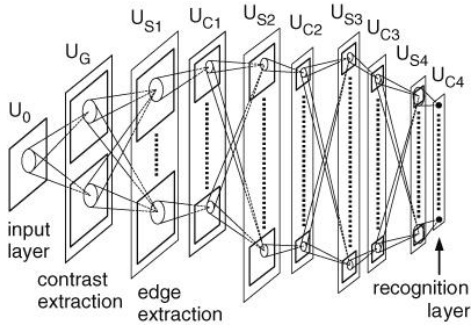


Figura 6. Arhitectura neocognitronului (Sursa: Fukushima, 2007 [11])

Fiecare strat C are ca intrări ieşirile stratului S anterior care sunt fixe, iar stratul S are intrări variabile care se modifică în cursul învăţării, fiecare celulă S ajungând să răspundă selectiv la o anumită caracteristică apărută în câmpul ei receptor asociat. În straturile iniţiale 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 învăţare). Conexiunile de intrare ale celulelor C provin de la ieşirile unui grup localizat de celule S din stratul anterior care extrag aceeaşi caracteristică dar în poziţii uşor modificate. Celule C răspund dacă măcar una dintre respectivele celulele S este activată la ieşire. Astfel se asigură o anumită invarianţă la mici translaţii, generându-se însă totodată şi un efect de înceţoşare la propagarea între straturile succesive. Fiecare strat S sau C este împărţit în substraturi denumite „plane de celule" (matrici 2-D de celule plasate retinotopic şi partajând acelaşi set de intrări şi oferind astfel o simetrie translaţională pentru acestea), corespunzător caracteristicilor la care respectivele celule răspund. De-a lungul timpului au fost propuse diferite modificări şi variante ale neocognitronului. De exemplu, prin adăugarea de conexiuni inverse (dinspre straturile superioare înapoi), s-a obţinut abilitatea recu-

noaşterii corecte chiar şi a unor tipare (parţial) acoperite/mascate şi pot fi reconstruite părţile ascunse ale acestora (Fukushima, 1987; 2005 [9; 10]). Chiar şi în cazurile în care apar simultan mai multe tipare, neocognitronul îşi focalizează atenţia asupra fiecăruia, unul câte unul şi le recunoaşte corect.

În fine, prima aplicaţie de succes a reţelelor neuronale convoluţionale (RNC) datează din anii ’90 şi este reprezentată de arhitectura LeNet-5 (Yann LeCun, 1989; 1998 [30; 31]).

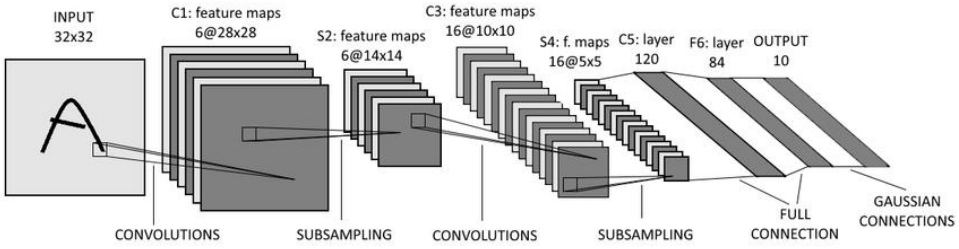


Figura 7. Arhitectura LeNet-5 (Adaptare după: LeCun, 1998 [31])

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

Reţelele neuronale convoluţionale (RNC) sunt astăzi cele mai utilizate pentru detecţia şi segmentarea obiectelor în imagini, precum şi pentru recunoaşterea / clasificarea obiectelor şi scenelor. Principalele avantaje ale RNC sunt:

- elimină necesitatea extragerii manuale a caracteristicilor, acestea fiind învăţate în mod automat direct din datele de antrenare;
- nivelul de performanţă la recunoaştere este extrem de ridicat (comparabil cu, sau peste cel uman);
- pot fi reînvăţate pentru noi sarcini de recunoaştere şi permit construcţii ulterioare pe baza unor (părţi iniţiale din) RNC preantrenate (*transfer learning*).

Foarte pe scurt, o reţea convoluţională este practic compusă din două părţi. Prima parte alternează straturi neuronale cu conexiuni locale de tip convoluţional şi respectiv de agregare pe bază de maxim sau medie (*max pooling / average pooling*) cu reducere dimensională (*subsampling*). Această parte „convoluţională” realizează extragerea automată a caracteristicilor specifice (*features extraction*) din datele de intrare. Partea finală a unei astfel de reţele 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 ieşirile părţii iniţiale convoluţionale (caracteristicile extrase automat de aceasta) obţinute pentru fiecare imagine aplicată intrării reţelei. Cu alte cuvinte, prima parte, cea convoluţională, 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 (şi care în particular poate fi şi un SVM sau orice altă variantă clasică utilizând ca intrare vectori de caracteristici). Vom detalia în cele ce urmează principalele componente şi mecanisme implicate în RNC.

Straturile convoluţionale

Convoluţia discretă a două funcţii 1-D *f* şi *g* definite pe mulţimea numerelor întregi *Z* este dată de:

$$(f * g)[j] = \Sigma f[i] \cdot g[j - i] = \Sigma f[j - i] \cdot g[i], \tag{5}$$

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

$$(a * b)[i, j] = \Sigma \Sigma a[m, n] \cdot b[i - m, j - n]. \tag{5'}$$

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

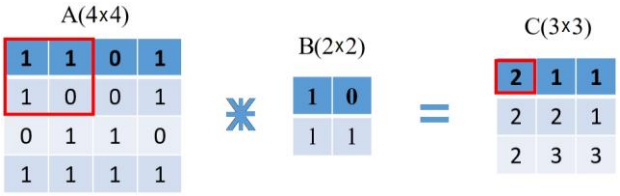


Figura 8. Exemplu de convoluţie a două matrici pătrate A (4×4) şi B (2×2), rezultând matricea pătrată C (3×3) (Adaptare după: Zhang, 2018 [46])

Fiecare element din matricea rezultat C este obţinut ca sumă a produselor elementelor corespunzătoare din aceleaşi poziţii din matricea (filtru, sau nucleu / *kernel*) de convoluţie B şi din regiunea din matricea iniţială A acoperită de B, prin glisarea matricii B în poziţii succesive peste toată matricea A pe orizontală şi pe verticală cu acelaşi pas (*stride*) *s = 1*.

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

$$c = a - b + 1. \tag{6}$$

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

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

$$c = a - b + 2p + 1, \tag{7}$$

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

$$p = (b - 1) / 2. \tag{8}$$

În cazurile prezentate mai sus, s-a considerat pasul de avansare pe orizontală şi verticală a „ferestrei" B peste matricea A, (*stride*), *s = 1*, acesta putând lua însă şi 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 relaţia:

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

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

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

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

lor de convoluţie (care sunt, aşa cum am arătat, matrici 3-D) sunt reprezentate de valorile ponderilor conexiunilor respective între câmpul receptor de pe stratul anterior, de dimensiuni spaţiale egale cu cele ale filtrului şi 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 convoluţie de dimensiuni $H_f \times W_f \times C$, se obţine o ieşire de dimensiuni $H_o \times W_o \times K$ (prin convoluţie, volume „trec" în alte volume), unde dimensiunile ieşirii sunt:

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

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

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

Mai facem următoarele observaţii: dimensiunile spaţiale ale filtrelor/nucleelor de convoluţie 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 aceeaşi dimensiune respectivă a hărţii de intrare pe care se aplică filtrul convoluţional respectiv.

Un strat convoluţional complet mai conţine şi o neliniaritate aplicată asupra rezultatului convoluţiei cu filtrele specifice stratului, funcţia 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]), ieşirea acestui strat fiind:

$$Y = g(f(X)) = max(0, f(X)). \tag{11}$$

Astfel, pentru un strat convoluţional *l*, cu intrarea X^{l-1} ($X^0 = (x^0_1, x^0_2, \ldots, x^0_n)$ reprezentând vectorul de intrare al reţelei), rezultatul aplicării convoluţiei cu fiecare filtru este o combinaţie liniară *f*:

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

iar ieşirea acestuia este dată de aplicarea neliniarităţii (ReLU), *g*:

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

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

Mai facem observaţiile că: „adâncimea" unei astfel de hărţi este egală cu numărul filtrelor de convoluţie aplicate în stratul respectiv (*K*), iar numărul total de parametri – ponderi w^l şi *bias*-uri b^l – care definesc un strat convoluţional (*n_param*) este determinat de dimensiunea şi numărul acestor filtre (*K*):

$$n\_param = (H_f \times W_f \times C + 1) \times K. \tag{14}$$

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

Trebuie remarcat faptul că, spre deosebire de cazul reţelelor 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 (40×40×3), tot 650 parametri sunt necesari în stratul convoluţional cu 50 de filtre exemplificat mai sus, în timp ce pentru situaţia complet conectată numărul parametrilor necesari ar fi crescut şi el de 100 ori, la 65.000 parametri.

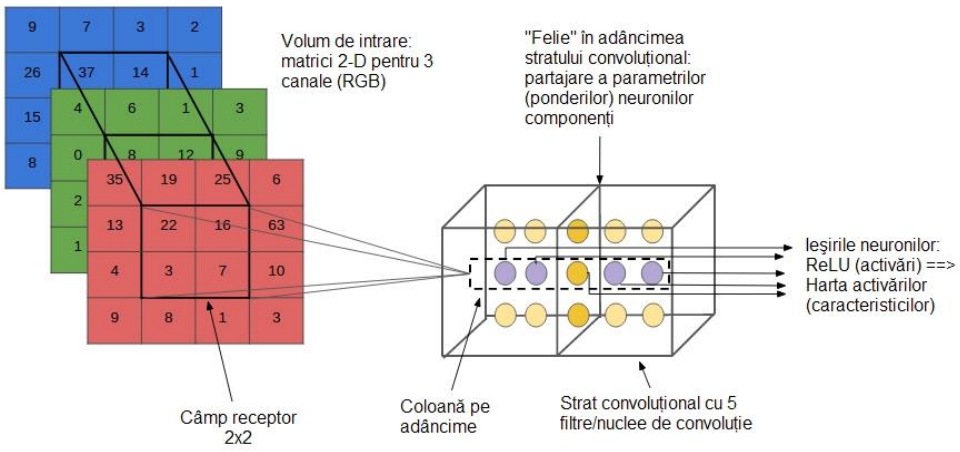


Figura 9. Straturile convoluţionale transformă volume în volume
(Adaptare după: Saxena, 2016 [39])

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

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

Aceste straturi au rolul de a înlocui ieşirile furnizate de aplicarea neliniarităţilor ReLU finale ale unui strat convoluţional precedent (harta activărilor rezultată din acesta) cu o statistică sumară locală a acestora şi de a furniza o reducere dimensională corespunzătoare totodată. Rezoluţia detaliilor este astfel diminuată, dar este propagată în straturile superioare informaţia esenţială integratoare, asigurându-se totodată evitarea exploziei numărului de parametri implicaţi 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 reţea. Pentru fiecare regiune din matricea de intrare acoperită de „filtrul de agregare” (câmp receptor – *receptive field*) este preluată din aceasta în matricea de ieşire corespunzătoare fie valoarea locală maximă, fie, respectiv, media valorilor locale din respectiva regiune.

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

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

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

O justificare intuitivă pentru aplicarea unui asemenea tip de agregare prin reţinerea valorilor locale maxime poate fi sugerată de faptul că valorile obţinute prin convoluţie dau o măsură a gradului de corelaţie între filtrul aplicat şi câmpul receptor asupra căruia acesta a fost aplicat, valorile maxime indicând corelaţie maximă. Deci acestea apar la identificarea caracteristicii specifice filtrului respectiv în câmpul receptor. Prin urmare, valoarea maximă furnizează informaţia cea mai semnificativă în ceea ce priveşte prezenţa sau absenţa caracteristicii respective în câmpul receptor.

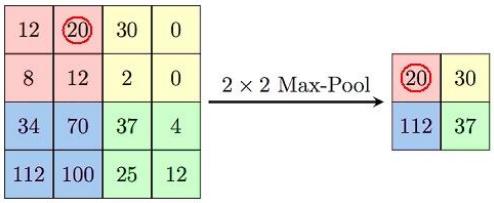


Figura 10. Exemplu de max pooling 2×2 (f = 2), fără padding, cu pas s = 2, fără suprapunere (s = f), rezultând o matrice pătrată (2×2) cu dimensiuni înjumătăţite (Adaptare după: Karpathy, 2018 [27])

În acest mod, un astfel de strat de agregare contribuie totodată şi la asigurarea unei anumite invarianţe/independenţe faţă de mici translaţii în poziţia obiectelor în imagini. Pe de altă parte, trebuie amintit faptul că aplicarea unui acelaşi filtru pe toate câmpurile receptoare localizate dar care pe ansamblu acoperă în întregime intrarea stratului convoluţional respectiv asigură independenţa totală faţă de poziţia în care este identificată caracteristica asociată acelui filtru.

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

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

$$d^{l} = s^{l} \cdot d^{l-1}, \tag{17}$$

unde f^i este dimensiunea filtrelor din stratul *i*, s^i este pasul de aplicare al acestora, iar d^i este distanţa între două câmpuri receptoare din imaginea iniţială 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 reţea (câte straturi să conţină), evident corelat şi cu dimensiunea filtrelor şi pasul utilizat în fiecare strat de convoluţie şi de agregare, este şi acela ca apertura (câmpul receptor iniţial al) unui neuron din harta de caracteristici finale să acopere (aproximativ) toată imaginea aplicată la intrarea reţelei convoluţionale.

Stratul/straturile finale complet conectate

Legătura între partea convoluţională a RNC şi clasificatorul final din aceasta se realizează printr-un aşa-numit „strat de aplatizare" (*flatten layer*), un pseudostrat care nu produce modificări de valori, ci are doar rolul de a adapta ieşirea 3-D rezultată la finalul porţiunii convoluţionale, prin translatare în 1-D, pentru utilizarea ca vector de caracteristici la intrarea clasificatorului final. Acesta din urmă este practic un perceptron multistrat cu funcţii de activare de tip softmax, furnizând la ieşire, pe baza vectorului de intrare, un număr de probabilităţi egale cu numărul de clase definite pentru spaţiul problemei, câte o probabilitate pentru fiecare dintre acestea. Antrenarea reţelelor neuronale convoluţionale 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 învăţare când este cazul, şi 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 şi convoluţionale). Pentru evitarea atenuării gradientului (*vanishing gradient*) prin mai multe straturi, se utilizează funcţii de activare a neuronilor neliniare de tip ReLU, care evită saturaţia celor de tip sigmoidă, asigurând totodată şi o reducere a numărului de ieşiri active (cele cu valori negative căpătând valoarea zero). Fiind implicat un număr foarte mare de parametri, pentru evitarea *overfitting*-ului şi asigurarea unei generalizări coerente şi consistente sunt utilizate diferite tehnici de regularizare precum normalizarea iniţială şi extinderea setului de date de intrare la antrenare (*data augmentation*), *dropout*-ul, normalizarea locală a valorilor de activare ale neuronilor la ieşirea unor straturi, normalizarea pe (mini)loturi (*(mini-)batch normalization*) ş.a.m.d. Vom încerca ilustrarea pe scurt a acestora în cele ce urmează, în contextul prezentării a două dintre arhitecturile neuronale convoluţionale de referinţă dintre cele mai utilizate: AlexNet şi VGG.

5. Arhitecturi RNC de referinţă

AlexNet (2012)

Arhitectura neuronală convoluţională (modelul) AlexNet (Krizhevsky, 2012 [29]) a câştigat competiţia ImageNet ILSVRC 2012 cu o îmbunătăţire semnificativă a celei mai bune performanţe privind rata de recunoaştere obţinută cu arhitecturi clasice. Cea mai mare reţea proiectată şi realizată până în acel moment, AlexNet a fost astfel prima reţea neuronală convoluţională care şi-a dovedit avantajele şi superioritatea netă faţă de cele mai performante abordări neconvoluţionale anterioare, devenind un important punct de referinţă şi catalizator pentru dezvoltările şi îmbunătăţirile ulterioare în domeniul reţelelor convoluţionale.

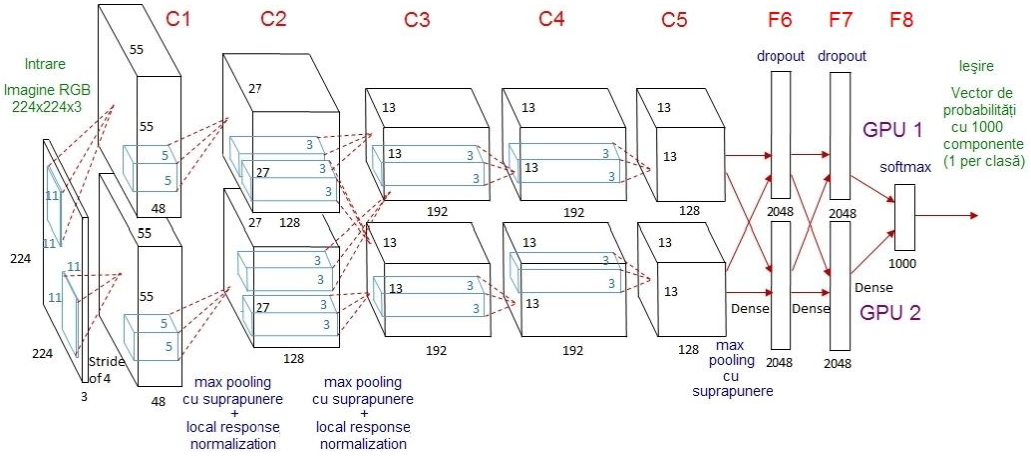


Figura 11. Arhitectura reţelei AlexNet (Adaptare după: Krizhevsky, 2012 [29])

AlexNet utilizează însă la intrare imagini RGB de dimensiune constantă, fixă, 224×224. De aceea imaginile, în general dreptunghiulare de dimensiuni diferite, au fost iniţial scalate astfel încât dimensiunea cea mai mică a acestora să ajungă 256 pixeli, apoi fiind decupate regiunile centrale după caz, astfel încât să se obţină 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 obţinut au fost practic utilizate fragmente 224×224 aşa cum se va arăta mai jos.

Reţeaua AlexNet este compusă din opt straturi: primele cinci convoluţionale şi ultimele trei complet conectate. Ieşirea ultimului strat complet conectat produce o distribuţie de probabilitate peste cele 1.000 de (etichete de) clase utilizând funcţia softmax. Adâncimea reţelei s-a dovedit a fi importantă. Astfel, dacă se elimină un strat convoluţional (care conţine cel mult 1% din parametrii modelului), nivelul de performanţă scade cu 1-2%.

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

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

Al treilea strat convoluţional aplică 384 de nuclee de dimensiune 3×3×256 pe ieşirea normalizată şi agregată a celui de-al doilea strat, cu suprapunere, cu pas (*stride*) de 1 pixel şi *padding* 1, rezultând la ieşirea stratului o hartă de caracteristici de dimensiuni 13×13×384.

Al patrulea strat convoluţional are 384 nuclee de dimensiune 3×3×192, care sunt aplicate ieşirii stratului al treilea, cu pas (*stride*) de 1 pixel şi *padding* 1, rezultând la ieşirea stratului o hartă de caracteristici tot de dimensiuni 13×13×384.

În fine, al cincilea strat convoluţional are 256 nuclee de dimensiune tot 3×3×192, care sunt aplicate ieşirii stratului al patrulea, cu pas (stride) de 1 pixel şi padding 1. Rezultă o hartă de caracteristici de dimensiune 13×13×256, asupra căreia este aplicată o agregare *max pooling* 3×3 finală, cu suprapunere, cu pas (*stride*) de 2, rezultând la ieşirea stratului (şi al părţii convoluţionale) o hartă de caracteristici de dimensiuni 6×6×256.

Straturile şase şi şapte, complet conectate (dense), au fiecare câte 4096 neuroni.

Stratul complet conectat opt (final) furnizează 1.000 de neuroni de ieşire (câte unul pentru fiecare clasă) şi utilizează funcţia softmax pentru calcularea distribuţiei de probabilitate peste cele 1.000 de clase.

În total, întreaga arhitectură a reţelei AlexNet are aproximativ 659.000 de neuroni şi 60 milioane de parametri care trebuie antrenaţi.

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 împărţite în două grupuri de câte 48 fiecare şi în mod similar toate filtrele aplicate mai departe în partea convoluţională au fost împărţite în grupe egale pe cele două „fire”. Cu excepţia celui de-al treilea strat convoluţional care este conectat la ambele „fire”, conexiunile între straturile convoluţionale păstrează „firul” de execuţie. Straturile finale complet conectate au (şi ele) conexiuni de la ambele „fire”. Antrenarea utilizând cele două GPU a durat 5-6 zile (2012).

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

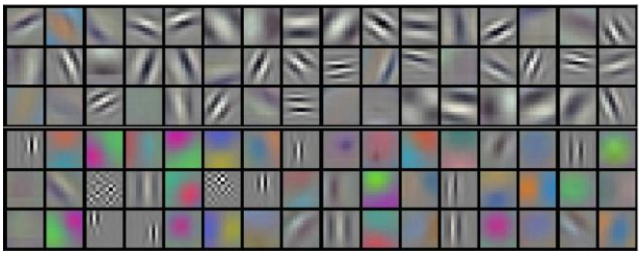


Figura 12. Nucleele/filtrele convoluţionale învăţate în primul strat convoluţional pe cele două „fire” de procesare paralelă pe câte unul dintre cele două GPU (Sursa: Krizhevsky, 2012 [29])

Astfel, în primul strat convoluţional de exemplu se observă formarea unei varietăţi de filtre elementare, selective după frecvenţă şi respectiv după orientare, precum şi a unei varietăţi de filtre bazate pe culoare.

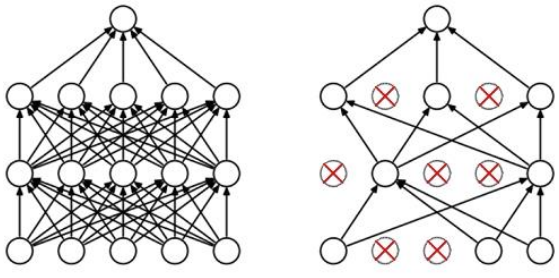
Demnă de remarcat este „specializarea” pe fiecare dintre cele două GPU, rezultat al conectivităţii restricţionate pe cele două fire ale arhitecturii. Nucleele corespunzătoare GPU 1 sunt în majoritate independente de culoare, pe când cele corespunzătoare GPU 2 sunt majoritar cu specific de culoare. Acest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

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

Un alt aspect specific important este utilizarea normalizării locale a răspunsului (*local response normalization*), respectiv a valorilor de activare de la ieşirea primelor două straturi convoluţionale. Este vorba de o normalizare statistică locală a acestora prin diminuarea lor la o valoare în jurul jumătăţii, ţinându-se cont şi de valorile de activare ale neuronilor din aceeşi poziţie spaţială (din aceeaşi coloană pe adâncime) din două „felii" vecine anterioare şi ulterioare „feliei" curente. În acest mod, conform autorilor, s-a creat „o competiţie între activările mari de la ieşirile 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 îmbunătăţire 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 şi 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*) şi 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 obţinute prin translaţii şi reflexii orizontale ale unor regiuni aleatoare din imaginile din setul de antrenare. Aceasta se realizează prin extragerea aleatoare a unor porţiuni 224×224 (şi generarea şi a reflexiei orizontale a acestora) din imaginile 256×256 din setul de antrenare, etichetate la fel cu imaginea din care au provenit şi învăţarea reţelei utilizându-se aceste fragmente. Se obţine în acest mod o creştere a setului de date de antrenare cu un factor de 2048, chiar dacă exemplele de antrenare astfel obţinute sunt în mare parte foarte interdependente. Totodată, la testare, reţeaua face predicţii prin extragerea a cinci fragmente 224×224 (cele patru colţuri şi zona centrală) cu generarea şi a reflexiilor orizontale corespunzătoare (deci un total de zece fragmente) şi medierea predicţiilor stratului softmax al reţelei pe cele zece fragmente. A doua formă de extindere a setului de date constă în modificarea intensităţilor în canalele RGB ale imaginilor de antrenare utilizându-se analiza componentelor principale (*Principal Components Analysis* - PCA). Astfel este simulată invarianţa identităţii obiectelor în imaginile naturale la schimbări de culoare şi intensitate a iluminării acestora. Prin extinderea setului de date (*data augmentation*), s-a obţinut o reducere a ratei de eroare cu peste 1%.

Noua tehnică "*dropout*" utilizată de AlexNet constă în setarea aleatoare la zero a ieşirilor unor neuroni de pe straturile ascunse cu o probabilitate de 0,5 (respectiv jumătate dintre aceştia). Neuronii care sunt astfel "decuplaţi" nu contribuie la propagarea înainte a semnalului şi nici nu participă la ajustarea celorlalte ponderi la antrenarea cu *backpropagation*. În acest mod, pentru fiecare imagine de intrare prezentată, reţeaua oferă o arhitectură diferită, dar toate aceste arhitecturi partajează acelaşi set de ponderi. Această tehnică reduce coadaptările complexe ale neuronilor, din moment ce un neuron nu se poate baza pe prezenţa altor neuroni particulari, fiind astfel forţat să înveţe caracteristici mai robuste care sunt utile în conjuncţie cu multe subseturi aleatoare de alţi neuroni.



a) b)

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

La testare, sunt utilizaţi toţi neuronii arhitecturii complete, fără *dropout*, dar ieşirile acestora sunt multiplicate cu 0,5, ceea ce constituie o aproximare rezonabilă a preluării mediei geometrice a distribuţiilor predictive produse de numărul exponenţial mare de reţele *dropout*.

VGGNet (2014)

Pe măsură ce reţelele neuronale convoluţionale au evoluat devenind un instrument comun pentru domeniul vederii artificiale, au apărut mai multe încercări de îmbunătăţire a arhitecturii AlexNet originale (Krizhevsky, 2012 [29], cu scopul obţinerii unei acurateţi (chiar) mai bune. De exemplu, câştigătorii competiţiei ILSVRC-2013 (Zeiler, 2014 [45]; Sermanet, 2014 [40]) au utilizat câmpuri receptoare de dimensiune mai mică, precum şi paşi mai mici în primul strat convoluţional. O altă direcţie de îmbunătăţire a fost aceea de a se antrena şi testa reţelele dens pe întreaga imagine şi la mai multe scale (Sermanet, 2014[40]; Howard, 2014 [23]).

Una dintre arhitecturile câştigătoare ale competiţiei ILSVRC 2014 (Simonyan, 2015 [41]), ulterior referită ca VGGNet, a adresat un alt aspect important şi anume adâncimea reţelei convoluţionale, prin adăugarea succesivă de noi straturi în timp ce ceilalţi parametri ai acesteia au fost menţinuţi. Aceasta s-a putut realiza prin utilizarea de filtre de convoluţie de dimensiune foarte mică (3×3) în toate straturile. Reţeaua VGGNet de referinţă constă în 16 (VGG16) sau respectiv 19 (VGG19) straturi, dintre care primele 13, respectiv 16, sunt convoluţionale (grupate în blocuri succesive, fiecare terminat cu un strat de agregare de tip *max pooling*) şi ultimele 3 sunt complet conectate, fiind este extrem de atractivă datorită arhitecturii sale foarte uniforme, cu puţini hiperparametri. În mare măsură VGGNet poate fi considerată similară cu AlexNet pe un singur „fir", cu convoluţii exclusiv 3×3 structurate în straturi cu secvenţe de mai multe astfel de filtre consecutive per strat convoluţional.

Se poate uşor observa că unei secvenţe de două straturi cu filtre convoluţionale 3×3 consecutive, fiecare cu pas 1, fără agregare între ele, îi corespunde un câmp receptor 5×5, iar unei secvenţe compuse din trei astfel de convoluţii îi corespunde un câmp receptor efectiv 7×7. Prin utilizarea unei astfel de secvenţe de două sau trei convoluţii 3×3 în locul uneia singure 5×5 sau respectiv 7×7, pe de o parte sunt introduse două, respectiv trei neliniarităţi ReLU în locul uneia singure, ceea ce conduce la o funcţie de decizie mai discriminativă, iar pe de altă parte se obţine totodată şi o scădere importantă a numărului de parametri. De exemplu, pentru C canale de culoare, un filtru/nucleu format din trei straturi de convoluţie 3×3 conţine un total de $3\times3^2\times C^2 = 27C^2$ ponderi, în timp ce un singur strat de convoluţie 7×7 conţine $1\times7^2\times C^2 = 49C^2$ ponderi, adică cu 81% mai multe. Aceasta poate fi privită ca o regularizare pe filtrele convoluţionale 7×7, prin descompunerea acestora în secvenţa de trei filtre 3×3 cu neliniarităţile inserate între ele.

Astfel, în locul utilizării unor câmpuri receptoare relativ mari în straturile convoluţionale (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 reţeaua, convoluţiile fiind aplicate asupra intrărilor cu pas (*stride*) 1.

Ca şi AlexNet, VGGNet utilizează la intrare imagini RGB de dimensiune constantă, fixă, 224×224, dar diferă modul de obţinere a acestora din imagini cu diferite rezoluţii. De asemenea, ca şi la Alexnet, singura preprocesare aplicată este normalizarea prin scăderea valorilor medii obţinute pe întreg setul de antrenare din valorile tuturor pixelilor pentru fiecare canal. Imaginea 224×224 aplicată la intrare este trecută iniţial printr-o succesiune de straturi convoluţionale în care sunt utilizate filtre cu câmpuri receptoare foarte mici 3×3 (dimensiunea minimă pentru care se poate vorbi despre un centru şi vecinătăţi stânga-dreapta şi sus-jos)). Ca o paranteză, menţionăm că a fost realizată şi testată inclusiv o arhitectură în care au fost utilizate de asemenea filtre de convoluţie 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 spaţial este 1, astfel încât să se păstreze rezoluţia spaţială (dimensiunile spaţiale) în urma aplicării convoluţiei.

Straturile convoluţionale sunt grupate – atât pentru VGG16 cât şi 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 şi reducerea dimensională prin înjumătăţirea fiecărei dimensiuni spaţiale, de la 224×224 iniţial la 7×7 la ieşirea ultimului strat de max pooling din ultimul bloc. Trebuie subliniat faptul că nu există astfel de straturi de agregare între straturile convoluţionale din cadrul fiecărui bloc. Numărul de straturi convoluţionale consecutive în fiecare din primele două blocuri/module este acelaşi (două), în timp ce în ultimele trei blocuri/module există secvenţe de câte 3, respectiv 4 straturi convoluţionale în cazul VGG16, respectiv VGG19. Numărul de filtre din fiecare strat convoluţional creşte (se dublează) de la bloc la bloc până la cel de-al patrulea inclusiv, de la 64 la 128, 256 şi respectiv 512, în cel de-al cincilea rămânând constant (512).

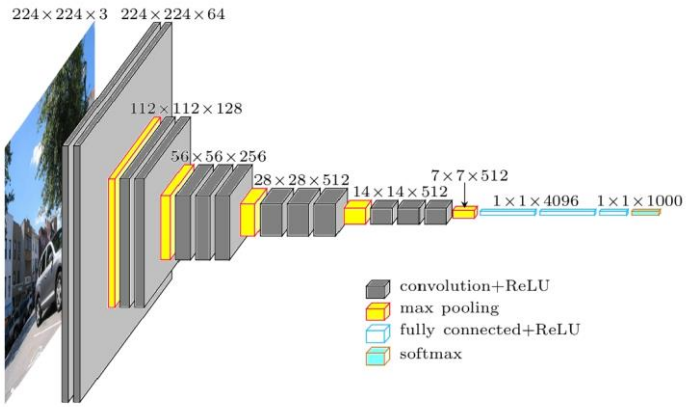


Figura 14. Macroarhitectura VGG16 (Adaptare după: Cord, 2016 [4])

Ieşirea succesiunii de straturi convoluţionale astfel organizate este aplicată ca intrare părţii de clasificare finale, 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 conţine 1.000 canale şi realizează efectiv clasificarea în cele 1000 clase ILSVRC. Ieşirea finală este calculată ca probabilităţi peste acestea utilizând funcţia softmax. Această configuraţie a straturilor complet conectate este aceeaşi în cazul ambelor reţele (VGG16 şi VGG19).

Spre deosebire de AlexNet, funcţia neliniară de activare ReLU (*Rectified Linear Unit*) este utilizată în toate straturile inclusiv în cele complet conectate, nu numai în cele convoluţionale. În acelşi 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 îmbunătăţire a performanţei, producând însă creşterea consumului de memorie şi a duratei de calcul.

Se poate observa că numărul de parametri per strat creşte continuu spre straturile finale (unde sunt „concentraţi" aproximativ 90% dintre aceştia), cu un maxim în primul dintre acestea, în timp ce memoria necesară per strat este maximă în primele straturi convoluţionale, scăzând continuu spre straturile finale complet conectate. În total, arhitectura VGGNet conţine aproximativ 138,3, respectiv 143,6 millioane de parametri şi 13,5, respectiv 14,8 milioane de neuroni (VGG16 vs. VGG19).

Procedura de antrenare a VGG Net urmăreşte în mare cea utilizată pentru AlexNet, cu diferenţa că fragmentele de imagini utilizate la intrare sunt decupate din imagini de antrenare multiscală. Învăţarea este realizată prin optimizarea funcţiei 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 şi moment 0,9. Antrenarea este regularizată cu *weight decay* şi *dropout* cu probabilitate 0,5 în primele două straturi complet conectate. Rata de învăţare este iniţializată la 10^{-2}, apoi redusă de 10 ori când acurateţea pe setul de validare nu se mai îmbunătăţeşte. În total, rata de învăţare a fost redusă astfel de 3 ori, iar învăţarea

a fost oprită după 370.000 de iteraţii (74 de epoci). Cu toate că are un număr mai mare de parametri şi o adâncime mai mare (mai multe straturi) în comparaţie cu AlexNet, VGGNet necesită mai puţine epoci pentru a converge datorită regularizării implicite produse de adâncimea mai mare şi dimensiunea mai mică a filtrelor de convoluţie, precum şi datorită preiniţializării unora dintre straturi – primele 4 convoluţionale şi ultimele 3 complet conectate – cu valori generate anterior la antrenarea unei variante de reţea VGG mai puţin adâncă (11 straturi). Ulterior, s-a demonstrat că ponderile pot fi iniţializate şi fără preantrenare, utilizându-se o procedură de iniţializare aleatoare (Glorot, 2010 [13]). Iniţial (2014), reţeaua VGGNet a fost antrenată pe 4 GPU nVIDIA GTX Titan Black cu câte 6GB RAM fiecare, timp de 2-3 săptămâni, funcţie 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 iteraţie 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 şi multi-scală, cu latura stabilită aleator într-un anumit interval (de exemplu [256÷512]). În cazul al doilea, s-a utilizat preiniţializarea 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, bucăţile decupate sunt supuse aleator la oglindire orizontală şi 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, reţeaua este aplicată dens pe imaginea de test rescalată, similar cu Sermanet, 2014 [40]. Astfel, straturile complet conectate sunt mai întâi convertite în straturi convoluţionale (primul cu filtre 7×7, iar ultimele două cu filtre 1×1). Apoi, reţeaua complet convoluţională astfel obţinută este aplicată pe întreaga imagine nedecupată. Rezultatul este o hartă de scoruri per clasă cu un număr de canale egal cu numărul de clase şi rezoluţie spaţială variabilă, dependentă de dimensiunea imaginii de intrare.

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

Astfel, aplicând reţeaua complet convoluţională 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 puţin 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 obţine o acurateţe îmbunătăţită datorită eşantionării mai fine decât în cazul reţelei complet convoluţionale.

De asemenea, evaluarea pe fragmente multiple este complementară evaluării dense (şi) datorită condiţiilor de frontieră pentru convoluţii: în cazul fragmentelor, *padding*-ul se face cu valori de zero, în timp ce în cazul evaluării dense *padding*-ul corespunzător unui acelaşi fragment apare natural din vecinătăţile fragmentului din imaginea respectivă, ceea ce măreşte substanţial câmpul receptor al întregii reţele, astfel încât este captat mai mult context. Totuşi, în practică, timpul de calcul suplimentar în cazul fragmentelor multiple nu justifică potenţialul câştig în acurateţe.

S-a constatat că, din punct de vedere al acurateţii, cele două arhitecturi cu 16 şi respectiv 19 straturi sunt comparabile, creşterea numărului de straturi ne(mai)aducând o creştere a acesteia.

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

Alte arhitecturi neuronale convoluţionale de referinţă, mai complexe/sofisticate, cu (şi) mai multe straturi, dar totuşi cu un număr de parametri mult redus ca ordin de mărime şi complexitate de calcul abordabilă, cu performanţe chiar mai bune (menţionate aici mai sus, dar şi în secţiunea 3):

- GoogLeNet – Szegedy, 2014 [43], cu 22 straturi şi aprox. 4 milioane de parametri; câştigătoare ILSVRC 2014; utilizează NIN (*Network in Network*); secvenţă de module Inception (fiecare cu căi paralele cu filtre având câmpuri receptoare de dimensiuni diferite, concatenate la ieşire); sunt utilizate convoluţii 1×1 pentru reducerea dimensionalităţii înaintea convoluţiilor costisitoare;
- ResNet (*Residual Network*) – He, 2015 [18], cu 152 straturi şi sub 4 milioane de parametri; câştigătoare ILSVRC 2015; secvenţă de module reziduale (fiecare cu minim două straturi convoluţionale şi conexiune suplimentară a intrării stratului iniţial direct la ieşirea stratului final al modulului unde se combină respectiva intrare cu ieşirea convoluţională), fără straturi finale complet conectate;
- DenseNet (*Densely connected convolutional network*) – Huang, 2016 [25], variantă la ResNet cu o uşoară îmbunătăţire;
- SENet (*Squeeze-and-Excitation Networks*) – Hu, 2017 [24], *wrapper* peste alte arhitecturi (ResNet, Inception etc.); câştigătoare ILSVRC 2017.

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

6. Concluzii

Reţelele neuronale convoluţionale (RNC) au ajuns astăzi practic standardul în aplicaţiile bazate pe analiza de imagini, performanţele lor egalând şi chiar depăşind performanţa umană în domeniu. Acest fapt se datorează pe de o parte inovării şi îmbunătăţirilor în zona modelelor (de ex. utilizarea ReLU) şi algoritmilor *deep learning* şi evoluţiei tehnologice în ceea ce priveşte puterea şi capacitatea de stocare şi procesare (inclusiv apariţia şi dezvoltarea procesoarelor grafice GPU), iar pe de altă parte, disponibilităţii de volume uriaşe de date de imagini (*big data*), capabile să surprindă complexitatea lumii înconjurătoare. În acest sens, crearea setului de imagini ImageNet (2009) şi competiţia asociată ILSVRC (2010-2017) a avut un important rol de catalizator. RNC elimină necesitatea extragerii manuale a caracteristicilor, acestea fiind învăţate în mod automat direct din datele de antrenare. Ele pot fi reînvăţate pentru noi sarcini de recunoaştere şi permit construcţii ulterioare pe baza unor (părţi iniţiale 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 convoluţionale să utilizeze filtre de dimensiuni mici (3×3 sau cel mult 5×5) şi pas (*stride*) 1, precum şi *padding* cu zero al volumelor de intrare astfel încât stratul convoluţional să nu altereze dimensiunile spaţiale ale intrării. În cazul necesităţii utilizării (şi a) unor filtre de dimensiune mai mare (7×7 sau mai mult), acestea sunt aplicate în general în primul strat convoluţional al RNC care are ca intrare imaginea originală.

Straturile de agregare sunt răspunzătoare de reducerea dimensională spaţială a intrării. Cel mai adesea se utilizează *max pooling* cu câmpuri receptoare 2×2 şi pas (*stride*) 2, astfel eliminându-se 75% din activările din volumul de intrare (datorită înjumătăţirii atât a lăţimii cât şi a înălţimii). Mai puţin utilizate sunt câmpurile receptoare 3×3 cu pas (*stride*) 2, dimensiuni mai mari nefiind practic utilizate datorită pierderii prea abrupte şi timpurii de rezoluţie ceea ce conduce la performanţe mai slabe.

Un argument în plus pentru utilizarea pasului 1 în straturile convoluţionale este acela că în practică s-a constatat că utilizându-se paşi mici se obţin rezultate mai bune, pe lângă faptul că – aşa cum am mai menţionat – pasul 1 lasă problema reducerii dimensiunilor spaţiale exclusiv straturilor de agregare, cele convoluţionale realizând doar transformarea adâncimii volumului de intrare (funcţie de numărul filtrelor aplicate). De asemenea, utilizarea *padding*-ului cu zero, pe lângă beneficiul păstrării dimensiunilor spaţiale după convoluţii, îmbunătăţeşte performanţa. În cazul neaplicării *padding*-ului cu zero a intrării şi efectuării convoluţiilor posibile, volumele sunt reduse puţin după fiecare strat convoluţional, informaţia de la margini fiind pierdută prea rapid.

În unele situaţii, datorită constrângerilor legate de memorie, este posibil să fie necesare compromisuri în ceea ce priveşte cele expuse mai sus. Astfel, mai ales în cazul primelor arhitecturi RNC, necesarul de memorie poate creşte foarte repede. De exemplu, dacă se aplică 3 straturi convoluţionale cu câte 64 de filtre 3×3 şi pas 1 fiecare pe o imagine 224×224×3 se generează 3 volume de activări de dimensiune 224×224×64 fiecare. Aceasta înseamnă un total de circa 10 milioane de activări, sau 72MB de memorie (per imagine, pentru activări şi gradienţi). 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 convoluţional al RNC. De exemplu, în cazul ZFNet [45] sunt utilizate în primul strat convoluţional filtre 7×7 cu pas 2, iar în cazul AlexNet [29] sunt utilizate filtre 11×11 cu pas 4.

În locul aplicării unui strat convoluţional cu filtre de dimensiune mai mare se poate aplica succesiv, cu acelaşi rezultat, o succesiune de straturi convoluţionale cu filtre mai mici, dar cu un număr total mai mic de parametri (de ex. 1 strat 5x5, respectiv 7x7 ≡ 2, respectiv 3 straturi 3x3).

Stratul/straturile finale complet conectate ale unei RNC pot fi înlocuite la clasificare, cu acelaşi rezultat, tot cu straturi convoluţionale (cu filtre 1x1), urmate de un strat de deconvoluţie, sau convoluţie transpusă (*fully convolutional network – FCN*) şi utilizându-se metode de tip *skip*, prin care valori de activare din straturi intermediare cu rezoluţie 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 şi metode de optimizare a performanţelor RNC, cum sunt:

- utilizarea convoluţiilor 1×1 pentru reducerea şi expandarea judicioasă a numărului hărţilor de caracteristici (*feature maps*);
- utilizarea unor tehnici de evitare a unor conexiuni (*skip*) şi/sau de creare a unor căi multiple de-a lungul reţelei;
- utilizarea unor metode eficiente de iniţializare a ponderilor, de regularizare şi/sau normalizare (de ex. *dropout, batch normalization*), respectiv de extindere a seturilor de date de antrenare / învăţare (*data augmentation*);
- medierea ieşirilor furnizate de clasificator pentru mai multe fragmente decupate / oglindite din imaginea originală;
- utilizarea ansamblurilor de reţele (*ensembles of networks*).

Menţiuni

Prezenta lucrare are la bază parte din activităţile şi rezultatele fazei I a proiectului PN 1819-0201 derulat la ICI Bucureşti (2018) în cadrul Programului naţional nucleu „RESINFO-TD” finanţat de Ministerul Cercetării şi 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/ (accesat 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 Revie*w 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 ştiinţific gradul III în Departamentul *„Sisteme inteligente distribuite intensive ca date“* din ICI București. Domeniile și subiectele sale de expertiză şi interes cuprind: vedere artificială (prelucrare şi analiză de imagini, recunoaşterea formelor, recunoaşterea optică de caractere – OCR, recunoaşterea numerelor de înmatriculare – LPR) şi învăţare 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).