Explorand Frontierele Cunoasterii: Inteligenta Artificiala si Invatarea automata

Elevi: Duca Elena Adelina

An II, informatica ID

Cuprins

[1. Introducere 3](#_Toc153094088)

[2. Introducere in Teoria retelelor neuronale. 4](#_Toc153094089)

[SLP si MLP 4](#_Toc153094090)

[Functia de activare 4](#_Toc153094091)

[Bias in Retele Neuronale 6](#_Toc153094092)

[Backpropagation 7](#_Toc153094093)

[Rata de eroare 8](#_Toc153094094)

[3. **Exemplu practic** - construirea unui perceptron cu un singur strat 8](#_Toc153094095)

[4. Notiuni avansate 10](#_Toc153094096)

[Retele neuronale convolutionale 10](#_Toc153094097)

[Evitarea supraantrenarii 15](#_Toc153094098)

[Pruning 15](#_Toc153094099)

[Transferul de invatare 16](#_Toc153094100)

[Invatarea supervizata 17](#_Toc153094101)

[Invatarea nesupervizata 19](#_Toc153094102)

[Invatarea online 19](#_Toc153094103)

[5. Alte tipuri de retele 19](#_Toc153094104)

[Retele neuronale recurente (Recurrent Neural Network) 19](#_Toc153094105)

[Biblioteci dedicate pentru realizarea retelelor neuronale 21](#_Toc153094106)

[Acceleratoare de hardware 21](#_Toc153094107)

[6. Utilizari ale Inteligentei Artificiale 22](#_Toc153094108)

[Medicina 22](#_Toc153094109)

[Securitatea Cibernetica 22](#_Toc153094110)

[Social Media 22](#_Toc153094111)

[Va deveni Inteligenta Artificiala mai inteligenta decat oamenii? 23](#_Toc153094112)

[Bibliografie 24](#_Toc153094113)

# Introducere

Într-o eră în care tehnologia evoluează exponențial, Inteligența Artificială (IA) și Învățarea Automată reprezintă coloanele vertebrale ale transformărilor care definesc prezentul și viitorul nostru digital. Aceste domenii nu sunt doar concepte tehnologice, ci promisiuni fascinante de a aduce mașinile la un nivel de înțelegere și adaptabilitate comparabil cu cel uman.

De la algoritmi avansați până la rețele neurale complexe, ne propunem să pătrundem în culisele tehnologiei care stă la baza Inteligenței Artificiale. Vom explora cum mașinile nu doar procesează informații, ci învață din ele, anticipând nevoile și adaptându-se la schimbările într-un mod care pară a depăși limitele programelor convenționale.

Prin acest proiect, vom arunca o privire retrospectivă asupra evoluției IA, de la rădăcinile sale în logica computațională până la apogeul său în rețelele neurale complexe. Vom dezvălui secretele tehnologiilor precum învățarea supervizată și nesupervizată, transferul de învățare și alte concepte, ilustrând cum acestea colaborează pentru a construi mașini care pot recunoaște modele, înțelege limbajul natural și chiar prevede comportamentul uman.

O incursiune în aplicațiile practice ale Inteligenței Artificiale ne va dezvălui cum aceasta transformă domenii variate precum medicina, finanțele, securitatea cibernetică și multe altele. Vom explora cum IA nu doar eficientizează procesele existente, ci și deschide uși către descoperiri inovatoare și soluții personalizate.

**Dar, in primul rand, ce este Inteligenta Artificiala?**

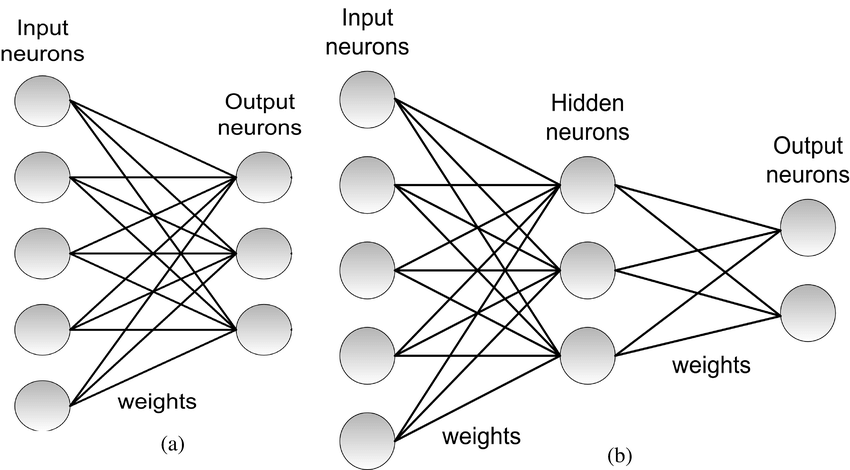
Inteligenta artificiala se concentreaza pe dezvoltarea si implementarea sistemelor care pot sa invete, sa rezolve probleme, incercand sa replice intr-o anumita masura capabilitatile cognitive umane. Aceste sisteme sunt bazate pe algoritmi matematici si sunt capabile sa analizele seturi gigantice de date, intr-un timp cu mult mai redus fata de o persoana, sa recunoasca modele si sa evolueze in timp.

Problema cea mai mare cu astfel de sisteme este faptul ca au nevoie de cantitati enorme de informatii pentru a fi in stare sa diferentieze tiparele din acele date, spre deosebire de oameni, care au nevoie doar de cateva exemple pentru a diferentia mediul inconjurator. Acesti algoritmi consuma multa energie, deoarece se folosesc multe exemple pentru a putea face recunoasterea, de multe ori, in diverse ipostaze, cu mici diferente - de exemplu, in cazul unor imagini, reteaua ar putea functiona mai bine daca ii oferim aceleasi obiecte din imagini in diverse unghiuri, luminozitati sau folosind rotatii, multiplicand semnificativ puterea de procesare. Pentru o simpla clasificare, astfel de retele au nevoie de cateva mii, chiar zeci de mii de exemple.

# Introducere in Teoria retelelor neuronale.

## SLP si MLP

Cel mai simplu exemplu este o retea de tip perceptron cu un singur strat (Single Layer Perceptron), precum in Figura 1(a). Aceasta este o forma de retea neuronala fundamentala si se foloseste drept un bloc de constructie pentru retelele mai complexe. O astfel de retea are nevoie de un set de neuroni de intrare (input), un set de neuroni de iesire (output) si conexiunile dintre acesti neuroni numite greutati (weighturi). O forma extinsa a acestui concept este un perceptron cu multiple straturi (Multi-Layer Perceptron), si consta in plus in adaugarea unor straturi “ascunse”, intre neuronii de intrare si cei de iesire. Poate fi un singur strat ascuns precum in Figura 1(b), sau mai multe, in functie de caz.



*Figura 1 SLP si MLP*

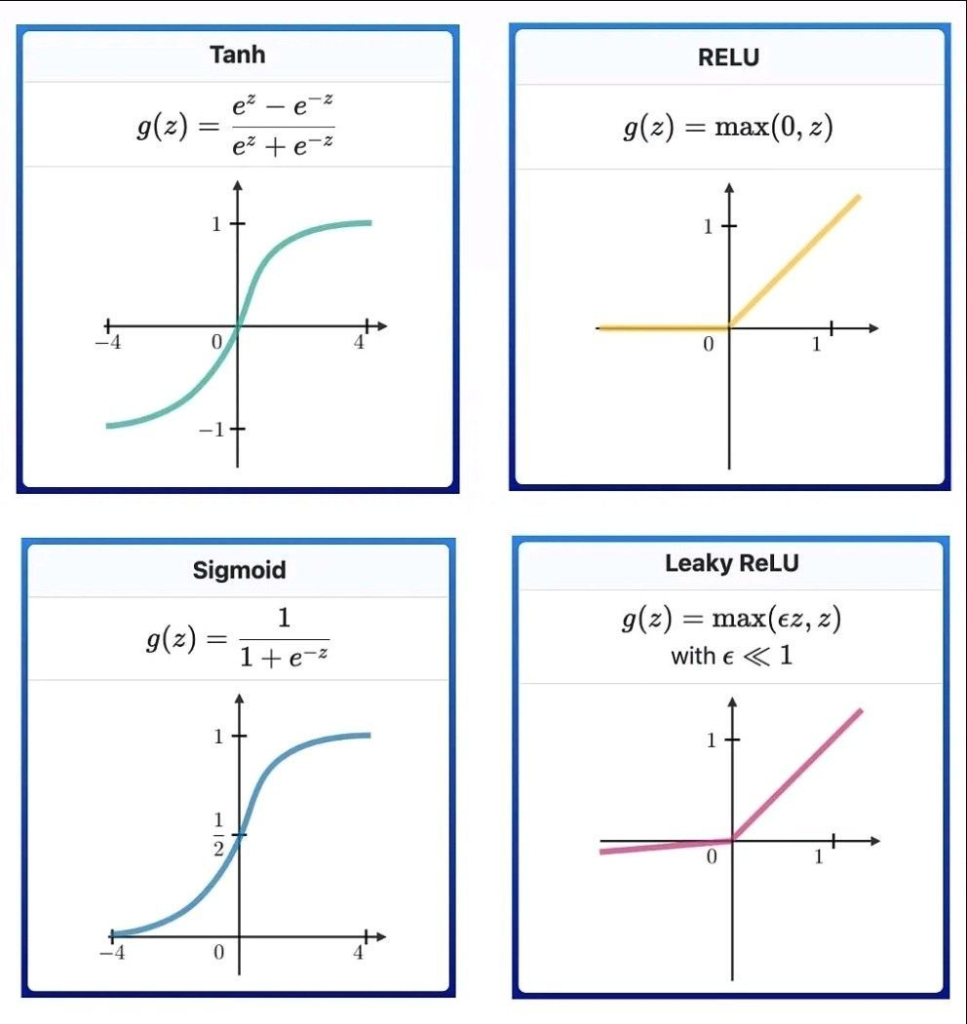
Neuronii de intrare transmit retelei neuronale semnalele initiale de la datele ce trebuie procesate. Neuronii de iesire clasifica informatiile introduse si este caracterizat de o eticheta atribuita fiecarui set de date. Neuronii ascunsi propaga informatia dintr-un capat in altul, adaugand complexitate retelei. Greutatea reprezinta o conexiune între doi neuroni. Această greutate indică importanța relativă a semnalului transmis de la neuronul sursă (neuronul de intrare) la neuronul destinație (neuronul de ieșire). În timpul antrenamentului rețelei, greutățile sunt ajustate pentru a optimiza performanța rețelei în realizarea sarcinilor specifice. De obicei, fiecare neuron dintr-un strat este conectat cu fiecare neuron din urmatorul strat printr-o greutate, dar exista si excepti, pe care le vom aborda mai tarziu.

## Functia de activare

O funcție de activare este esențială în procesul de funcționare a neuronilor într-o rețea neuronala. Această funcție preia valoarea de ieșire a unui neuron și o transformă într-un anumit mod. Importanța funcțiilor de activare sta în capacitatea lor de a normaliza și regla valorile neuronilor într-un interval specific. Acest proces de normalizare este benefic în construirea rețelelor neuronale, deoarece ajută la restrângerea intervalului de valori, facilitând astfel prelucrarea eficientă a datelor.

Există diverse tipuri de funcții de activare, iar alegerea acestora depinde de scopul specific al rețelei neuronale. Iată câteva exemple comune:

* Funcția Tangentă Hiperbolică (tanh): Similară cu funcția sigmoid, dar cu valori în intervalul [-1, 1]. Este folosită în diverse contexte, inclusiv în straturile ascunse ale rețelelor.
* Funcția Rectified Linear Unit (ReLU): Este egala cu zero pentru toate valorile negative și egala cu valorile positive in rest. Este una dintre cele mai populare funcții de activare, adesea utilizată în straturile ascunse.
* Funcția Sigmoid: Transformă valoarea într-un interval între 0 și 1, fiind adesea folosită în straturile de ieșire pentru probleme de clasificare binară.
* Funcția Leaky ReLU: Similară cu ReLU, dar permite un flux mic de valoare pentru valorile negative, evitând problema "neuronilor morți".



*Figura 2 Tipuri de functii de activare*

Alegerea funcției de activare poate influența performanța rețelei, iar experimentarea cu diferite funcții în funcție de natura problemei poate duce la rezultate optime. Astfel, funcțiile de activare reprezintă o componentă crucială în arhitectura și eficacitatea rețelelor neuronale

## Bias in Retele Neuronale

Pentru a conferi rețelelor neuronale o mai mare adaptabilitate și dinamism, s-a introdus un parametru esențial în construcția lor: bias-ul. Bias-ul joacă un rol crucial în procesul de învățare și ajustare a ieșirilor neuronilor, oferindu-le un grad suplimentar de libertate în procesul de luare a deciziilor.

Includerea bias-ului este ca un element cheie pentru a face ca rețelele să fie capabile să învețe și să se adapteze la modele care nu sunt centrate obligatoriu în jurul originii într-un sistem de coordinate, vom arata cum arata intr-un grafic intr-o alta sectiune. Practic, bias-ul este o valoare constantă adăugată ponderilor (greutăților) asociate intrărilor unui neuron înainte de a aplica funcția de activare. Acest pas adițional permite neuronilor să fie mai flexibili în interpretarea și procesarea datelor de intrare.

În esență, bias-ul contribuie la decalarea și ajustarea funcționării neuronilor, permițându-le să se potrivească mai bine cu variațiile din datele de intrare. De exemplu, într-o rețea neurală care învață să recunoască imagini, bias-ul poate ajuta la gestionarea diferențelor în luminozitate sau contrast, permițând rețelei să răspundă eficient la variabilitatea condițiilor de iluminare. Astfel, bias-ul este o componentă esențială în arsenalul rețelelor neuronale, facilitând adaptarea lor la complexitatea și diversitatea informațiilor cu care lucrează.

**Formula matematică pentru un neuron într-o rețea neurală:**

Pentru un neuron într-o rețea neurală de tip Single Layer Perceptron (SLP) sau Multilayer Perceptron (MLP), putem exprima valoarea acestuia printr-o ecuație matematică. Fie *x1​,x2​,...,xn*​ valorile neuronilor de intrare și *w1​,w2​,...,wn*​ greutățile asociate acestor conexiuni, iar b să fie bias-ul. Notând cu *f* funcția de activare, ecuația pentru valoarea neuronului este:

**Valoare neuron= f(w1​⋅x1​+w2​⋅x2​+...+wn​⋅xn​+b)** (vezi Figura 3)

Această formulă reprezintă suma ponderată a intrărilor (înmulțirea fiecărui input cu greutatea corespunzătoare), adăugând bias-ul, și aplicând apoi funcția de activare. Funcția de activare introduce non-linearități în rețea, permitând acesteia să învețe relații complexe în datele de intrare.

Pentru o rețea cu mai multe straturi (MLP), această formulă este aplicată recursiv pentru fiecare neuron din fiecare strat, rezultând într-o rețea complexă și adaptabilă la diverse tipuri de date.

O imagine care conține diagramă, linie, text, Desen tehnic

Descriere generată automat

*Figura 3 Formula matematica a unui neuron*

**Cum se ajusteaza greutatile?**

## Backpropagation

Backpropagation este o metodă esențială în ajustarea parametrilor (greutăți și bias-uri) unei rețele neurale. Pentru a înțelege acest proces, luăm în considerare un exemplu simplu, folosind un set de date compus doar din imagini. Desigur, această tehnică poate fi aplicată și altor tipuri de date, precum cuvinte sau semnale audio, dar pentru claritate, ne vom concentra pe imagini.

Sa presupunem că introducem o imagine în rețea, reprezentând pixelii acelei imagini. Rețeaua face o predicție, etichetând imaginea cu o anumită clasă. Comparăm această etichetă cu cea reală și calculăm o "rată de eroare". Acesta este pasul în care backpropagation devine crucial.

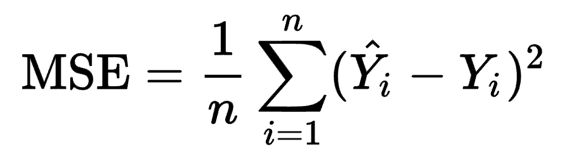
Pentru fiecare greutate și bias în rețea, ajustăm valoarea lor prin scăderea unui produs între greutate, valoarea pixelului corespunzător și o "rată de învățare". Alegerea ratei de învățare este importantă, deoarece controlează cât de mult impact are fiecare imagine asupra ajustărilor rețelei.

Procesul continuă propagându-se înapoi prin rețea. Pentru fiecare neuron, calculăm rata de eroare și efectuăm ajustările corespunzătoare. De exemplu, când trecem printr-o funcție de activare, calculăm inversul acelei funcții pentru a obține ajustările necesare. Toate aceste operații se desfășoară la nivel matricial, facilitând manipularea eficientă a datelor.

Este crucial să înțelegem că la început, toate greutățile și bias-urile sunt inițializate aleator. Aceasta este o parte esențială a procesului, deoarece rețeaua începe să învețe de la zero. Ajustările ulterioare se bazează pe performanța rețelei și sunt realizate în faza de învățare. În final, există și o fază de testare, în care evaluăm cât de bine se descurcă rețeaua în a face predicții pe date noi**.**

## **Rata de eroare**

Dupa cum am precizat mai devreme, rata de eroare este o comparatie dintre eticheta reala unei imagini si eticheta prezisa de catre retea si se calculeaza in functie de tipul acesteia. Una dintre cele mai cunoscute functii de calculare a erorii este eroarea medie a patratelor (Medium Square Error – MSE). Aceasta se calculează drept media pătratelor diferențelor dintre predicțiile modelului și valorile reale ale setului de date de testare, vezi Figura 4. Cu cât valoarea MSE este mai mică, cu atât modelul este considerat mai precis.



*Figura 4 Formula de calcul al erorii medie a patratelor*

# **Exemplu practic** - construirea unui perceptron cu un singur strat

In cele ce urmeaza, vom implementa in python notiune teoretice prezentate. Pentru acest exemplu, am folosit setul de cifre de la MNIST, mai exact cifra 0 si 1.

Vom avea nevoie de urmatoarele biblioteci: numpy – pentru a lucra cu datele sub forma matriciala mult mai efficient si mai rapid, folosindu-ne de functiilor predefinite ale acestei librarii; pandas – pentru a citi datele dintr-un csv. Acest pas depinde de forma de stocare a datelor, acestea pot fi citite si imagine cu imagine dintr-un fisier cu cv2 de exemplu, dar am observant ca dureaza mai mult timp sa se citeasca; keras.utils pentru a folosi functia np\_utils de transformare a etichetelor intr-o forma mai usor de folosit in conextul retelelor; sklearn.model\_selection pentru a ne folosi de functia train\_test\_split de despartire a unui set de date in doua seturi de date.

Asigură-te că aceste biblioteci sunt instalate în mediul tău de lucru. Dacă nu sunt deja instalate, poți utiliza comanda **pip install numpy pandas scikit-learn keras, tqdm**.

import numpy as np

import pandas as pd

from keras.utils import np\_utils

from sklearn.model\_selection import train\_test\_split

from tqdm import tqdm

def activare(neuron):

    #Implementarea functiei sigmoid

    return 1 / (1 + np.exp(-neuron))

def calculare\_neuron(input, weights, bias):

    return np.dot(weights, input) + bias

def eroarea\_medie\_a\_patratelor(y\_prezis, y\_adevarat):

    return 2 \* (y\_prezis - y\_adevarat) / np.size(y\_adevarat)

def ajustare\_greutati(input, eroare, rata\_de\_invatare):

     return weights - rata\_de\_invatare \* np.dot(eroare, input.T)

#Citirea setului de date si al etichetelor

testData = pd.read\_csv('D:\\Projects\\04\_Pattern Recognition\\Resources\\mnist\_numbers\_8x8\_0\_and\_1.csv')

#In acest caz, imaginile sunt stocate intr-un csv, fiecare imagine pe cate un rand, avand eticheta pe prima coloana si apoi pe rand fiecare pixel pe cate o alta coloana

X = testData.iloc[:, 1:].values / 255

X = X.reshape(X.shape[0],  8 \* 8, 1)

Y = testData.iloc[:, 0].values

Y = np\_utils.to\_categorical(Y) # aceasta functie transforma un vector de forma [2, 3, 1]  intr-o matrice de forma [[0,0,1,0],[0,0,0,1],[0,1,0,0]]

#Aceasta este o practica comuna in construirea retelelor neuronale pentru calcularea erorii fiecarei etichete.

Y = Y.reshape(Y.shape[0], 2, 1)

# Impartirea setului de date in doua parti - una de invatare si una de test

x\_invatare, x\_test, y\_invatare, y\_test = train\_test\_split(X, Y, test\_size=0.3)

input\_size = len(x\_invatare[0])

output\_size = 2 # Doar doua etichete, una pentru 0 si una pentru cifra 1

# Initializarea cu valori aleatorii a greutatilor

weights = np.random.randn(output\_size, input\_size)

bias = np.random.randn(output\_size, 1)

#Initializare parametrii retea

rata\_de\_invatare = 0.1

epochs = 100 # De cate ori sa se analizeze datele pentru ca reteaua sa le invete

for e in tqdm(range(epochs)):

    for input, eticheta in zip(x\_invatare, y\_invatare):

        #Calculare valoare neuroni

        neuron = calculare\_neuron(input,weights,bias)

        neuron\_activat = activare(neuron)

        #Calculare eroare

        eroare = np.array(eroarea\_medie\_a\_patratelor(neuron\_activat, eticheta))

        #Ajustare greutati

        weights = ajustare\_greutati(input, eroare, rata\_de\_invatare)

#Testarea retelei

correct = 0

for input, eticheta in tqdm(zip(x\_test, y\_test)):

        #Calculare valoare neuroni

        neuron = calculare\_neuron(input,weights,bias)

        neuron\_activat = activare(neuron)

        #Verificare daca reteaua a identificat corect reteaua

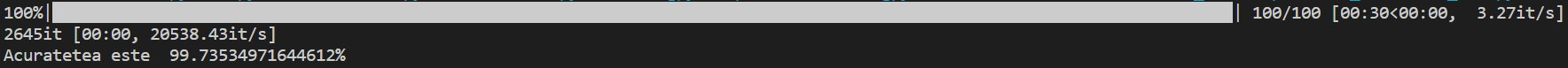
        if np.argmax(neuron\_activat) == np.argmax(eticheta):

            correct += 1

#Calcularea acuratetii retelei ca fiind numarul de etichete identificate corect impartit la numarul total de imagini

print("Acuratetea este ", str(correct / len(x\_test)\*100)+"%")

Dupa rularea acestui program, am obtinut o acuratete de 99,7%, deci reteaua noastra identifica foarte bine cifrele de 0 si 1.



# Notiuni avansate

Acum ca am explorat cum functioneaza retelele neuronale la baza, atat din punct de vedere al teoriei cat si prin ilustrarea unui exemplu practic, ramane totusi intrebarea cum reusesc ele sa analizeze date mult mai complexe.

## Retele neuronale convolutionale

Într-un efort de a analiza și înțelege date complexe, rețelele neuronale convolutionale (CNN sau ConvNets) deschid noi orizonturi în lumea inteligenței artificiale. Aceste rețele revoluționare utilizează o matrice specială numită kernel pentru a extrage caracteristici semnificative din imagini, facilitând astfel identificarea acestora cu mai puține resurse.

O imagine care conține artă, proiectare, ilustrație

Descrierea a fost generată automat, cu un grad mediu de încredere*Figura 5 Diferenta dintre un strat de tip perceptron si un strat convolutional*

**Cum funcționează:**

* Convoluție cu Kernel: Un kernel este o matrice mică care se deplasează peste întreaga imagine, vezi figura 6. În timpul acestei operații, kernel-ul examinează bucatele mici din imagine, extrăgând trăsături specifice. De exemplu, un kernel poate identifica marginile, altul culorile, iar un altul textura. Aceste trăsături sunt esențiale pentru identificarea și înțelegerea imaginilor.

O imagine care conține pătrat, captură de ecran, text, Dreptunghi

Descriere generată automat

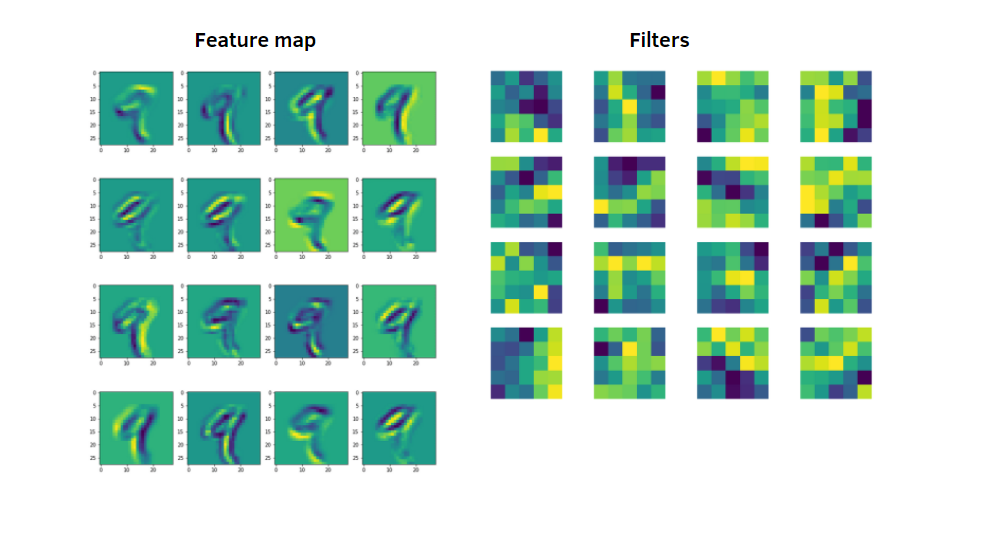
*Figura 6 Modul de functionare al unui kernel*

* Straturi de Pooling: După operația de convoluție, se aplică adesea straturi de "pooling" pentru a reduce dimensiunea datelor. Această etapă minimizează numărul de intrări pentru straturile ulterioare, facilitând procesarea. Pooling-ul poate fi de tip max pooling – care extrage maximum dintr-un spatiu definit sau average pooling – care face media valorilor din spatiul definit. Acestea de asemenea, ca si kernelul, se plimba pe intreaga imagine, vezi figura 7.



*Figura 7 Max Pooling si Average Pooling*

* Hărți de Trăsături: Rezultatul aplicării mai multor kerneluri asupra imaginii este o "hartă de trăsături". Aceasta reprezintă o imagine abstractă în care trăsăturile semnificative sunt evidențiate. Un exemplu de “feature map” este la figura 8 – in partea dreapta sunt kernelurile folosite, iar in stanga imaginile rezultate de trecerea acelor kerneluri.



*Figura 8 Feature map*

Pentru a ilustra acest concept, luăm în considerare o imagine cu flori. Un kernel ar putea identifica forma trandafirului, altul culoarea roșie, iar rețeaua, bazându-se pe aceste trăsături, poate recunoaște că în imagine este prezent un trandafir. Vezi figura 9 pentru un alt exemplu ilustrativ cu masini. O imagine care conține text, Vehicul de teren, vehicul, mașină

Descriere generată automat

*Figura 9 Exemplu de functionare a unui kernel care identifica masini intr-o imagine*

Aceste straturi convolutionale sunt eficiente deoarece analizează porțiuni mai mici ale imaginii, construind forme complexe treptat. De asemenea, s-a constatat științific că ochiul uman are tendința să se concentreze mai întâi pe forme simple, precum linii, care sunt apoi combinate în creier pentru a forma texturi complexe și, în cele din urmă, obiecte. Această abordare a straturilor convolutionale reflectă modul de funcționare a creierului uman.

**Strat convolutional pe text**

Când aplicăm un strat convolutional asupra datelor de tip text, procesul este similar cu cel utilizat pentru imagini, deși conceptul poate părea mai abstract. Cu toate acestea, metoda de funcționare rămâne identică. De exemplu, în cazul datelor textuale, fiecare literă a unui cuvânt este reprezentată într-o matrice de cuvinte. Această matrice trebuie să aibă un număr de coloane egal cu numărul maxim de litere posibile într-un cuvânt din text.

Procesul de Convoluție, vezi figura 10:

* Reprezentare Matriceală a Textului: Textul este transformat într-o matrice de caractere, unde fiecare caracter este reprezentat de o valoare specifică.
* Kernel-uri în Contextul Textului: Similar cu imagini, se folosesc kernel-uri care pot reprezenta secvențe de cuvinte. De exemplu, un kernel ar putea fi format din cuvinte consecutive, cum ar fi "nu fi".
* Aplicarea Convoluției: Kernel-ul se deplasează peste matricea textului, evaluând secvențele de cuvinte. Dacă există o corespondență între kernel și un fragment al textului, rezultatul convoluției va evidenția acea secvență.
* Detectarea Structurilor în Text: În contextul unei propoziții precum "A fi sau a nu fi", un kernel care caută secvența "nu fi" poate furniza o valoare mare la sfârșitul propoziției, indicând o corespondență. În schimb, într-o propoziție precum "a fi sau a fi nu", kernel-ul nu poate să recunoască nicio structură, deoarece secvența nu corespunde.

O imagine care conține diagramă, Plan, hartă, text

Descriere generată automat

*Figura 10 Schema de arhitectura conventionala pe text – in partea stanga este matricea de intrare, urmeaza kernelurile, rezultatele ce se formeaza in urma mutarea kernelelor, valorile dupa stratul de pooling, combinarea tuturor rezultatelor intr-un vector, apoi etichetarea datelor.*

Așa cum în cazul imaginilor se caută trăsături vizuale, în cazul textului se caută secvențe semnificative de cuvinte. Această abordare oferă rețelelor neuronale posibilitatea de a detecta și înțelege structurile și relațiile în datele text, contribuind astfel la analiza și procesarea lor într-un mod semnificativ.

## Evitarea supraantrenarii

**Permutarea datelor**

O retea neuronala are nevoie de o distributie aleatorie a datelor pentru a evita invatarea unui singur tip de eticheta. Cu cat datele sunt mai aleatorii invatate, cu atat reteaua poate generaliza mai bine. Un mod de a face reteaua sa generalizeze mai bine datele este permutarea datelor in mod aleatoriu la fiecare epoch.

...

for e in range(epoch):

        random\_indices = np.random.choice(len(x\_train), size=1000, replace=False)

        x\_learning = [x\_train[i] for i in random\_indices]

        y\_learning = [y\_train[i] for i in random\_indices]

        for x, y in tqdm(zip(x\_learning,y\_learning)):

...

**Dropout**

Dropout este o tehnică de regularizare utilizată în rețelele neurale pentru a preveni supraantrenarea (overfitting). La fiecare pas din timpul antrenamentului, se dezactiveaza in mod aleatoriu cativa neuroni. In felul acesta, neuronii invata caracteristici independente si reteaua poate generaliza mai bine.

## Pruning

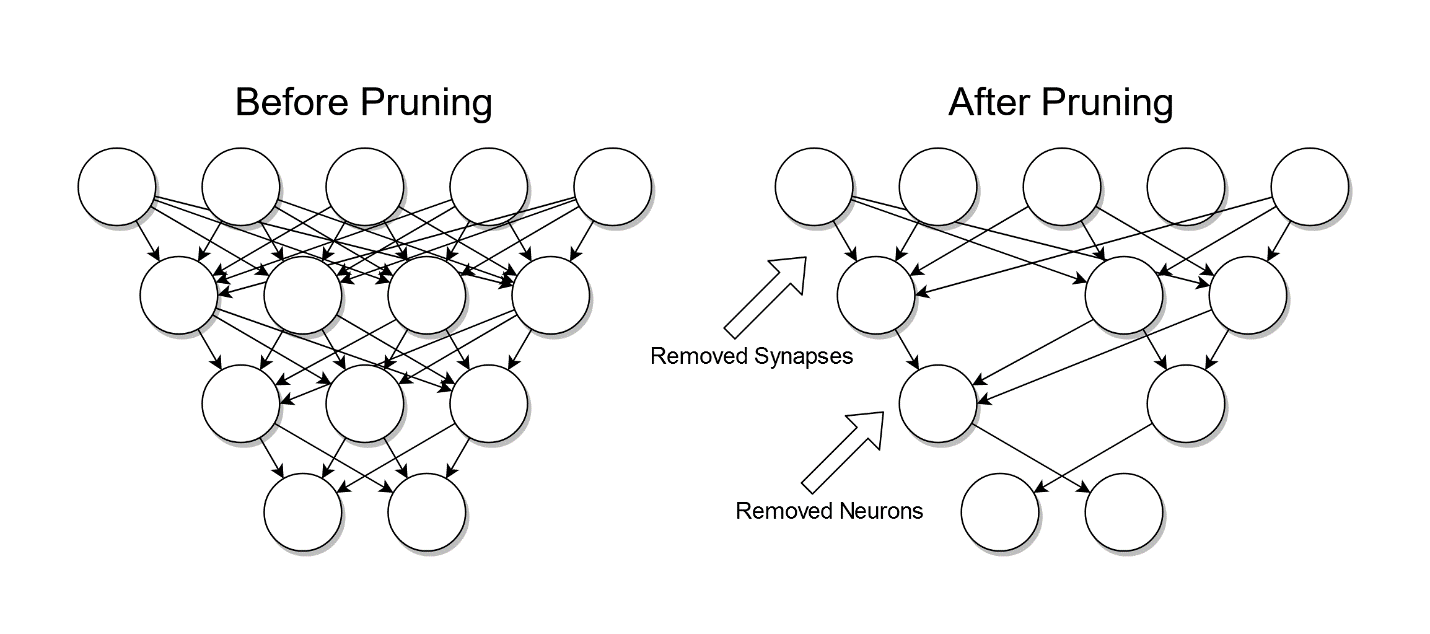
Pruning, în contextul rețelelor neurale, se referă la tehnica de eliminare a anumitor neuroni sau ponderi dintr-o rețea neurală pentru a obține o versiune mai mică și mai eficientă a modelului, fără a sacrifica semnificativ performanța. Această tehnică este utilizată pentru a reduce dimensiunea modelului și a eficientiza utilizarea resurselor computaționale.

Există două tipuri principale de pruning în rețelele neurale:

* Pruning al greutatilor (Weight Pruning): Această metodă implică eliminarea sau setarea la zero a unor greutati specifice ale modelului. Pruning-ul greutatilor se bazează pe ideea că multe ponderi ale rețelei pot avea valori foarte mici, iar eliminarea acestora nu afectează semnificativ performanța. Prin urmare, se elimina greutatile care au o contribuție relativ mică la predicție.
* Pruning al neuronilor (Neuron Pruning): În acest caz, întreaga conexiune a unui neuron este eliminată, inclusiv ponderile asociate cu acel neuron. Dacă un neuron pare să aibă o contribuție mică la performanța generală a rețelei, acesta poate fi eliminat în întregime.

Pentru a implementa procesul de pruning, mai intai trebuie antrenata reteaua, in mod normal, apoi se evalueaza importanta fiecărui neuron sau greutati la performanță. Aceste informații sunt utilizate pentru a decide ce neuroni sau greutati să fie eliminate. După eliminare, modelul este ajustat și, în unele cazuri, poate necesita încă o perioadă scurtă de antrenament pentru a compensa pierderea de informații.

Pruningul este avantajos, deoarece reduce dimensiunea modelului care necesită mai puține resurse computaționale, ocupa mai putin spatiu in memorie si este mai rapid. De asemenea poate fi o cale de a nu mai antrena o retea noua pentru un set de date – prin luarea unei retele mai complexe deja antrenare si implementarea pruning-ului pe neuronii/greutatile care nu contribuie la setul de date dorit, vezi figura 11.



*Figura 11 Pruning*

Totusi, prezinta si dezavantaje, deaorece necesită o gestionare atentă pentru a asigura că modelul rămâne funcțional și precis si se pot pierde informatii care pot afecta performanta modelului.

## Transferul de invatare

Reantrenarea unei rețele neurale deja existente, cum ar fi GoogleNet, se referă la procesul de ajustare sau extindere a modelului pre-antrenat pentru o nouă sarcină sau un nou set de date. Acest proces este util atunci când aveți un model pre-antrenat pe o sarcină similară sau set de date și doriți să îl adaptați pentru a rezolva o problemă specifică.

Iată pașii obișnuiți pentru reantrenarea unui model precum GoogleNet:

* Selectarea Modelului Pre-antrenat: Alegeți un model pre-antrenat care are o arhitectură potrivită pentru problema dvs. și care a fost antrenat pe un set de date mare și relevant. În cazul GoogleNet, acesta este un model complex de rețea neurală convoluțională
* Modificarea Straturilor de Ieșire: Deoarece veți utiliza modelul pentru o altă sarcină sau un alt set de date, va trebui să modificați stratul de ieșire al modelului pentru a se potrivi cu numărul de clase din noul set de date. Înlocuiți stratul de ieșire și stratul de clasificare cu noile straturi specifice sarcinii dvs.
* Antrenarea cu Noul Set de Date: Antrenați modelul modificat cu noul set de date. Această etapă va actualiza ponderile straturilor pe care le-ați adăugat sau înlocuit, permițând modelului să învețe caracteristicile specifice setului de date noi.
* Evaluarea Performanței: După antrenament, evaluați performanța modelului pe un set de date de validare sau testare pentru a evalua eficacitatea adaptării la noua sarcină.

Reantrenarea modelelor precum GoogleNet este o practică obișnuită și este utilă atunci când este nevoie să realizați o sarcină specifică, dar nu aveți suficiente date pentru a antrena un model de la zero.

## Invatarea supervizata

Supervised learning este un tip de paradigmă în machine learning în care un model este antrenat pe un set de date etichetat. În acest context, "supervizat" înseamnă că algoritmul de învățare primește un set de date care conține exemple de intrare împreună cu etichetele corespunzătoare, iar obiectivul este să învețe o funcție de asociere între intrările și ieșirile dorite si este si modelul explicat deja.

Probleme rezolvate prin învățare supervizată includ clasificarea - modelul trebuie să învețe să atribuie fiecărei intrări una dintre mai multe clase predefinite. De exemplu, recunoașterea imaginilor cu câini sau pisici. și regresia - modelul trebuie să prezică o valoare numerică sau continuă. De exemplu, estimarea prețului unei case în funcție de caracteristicile sale.

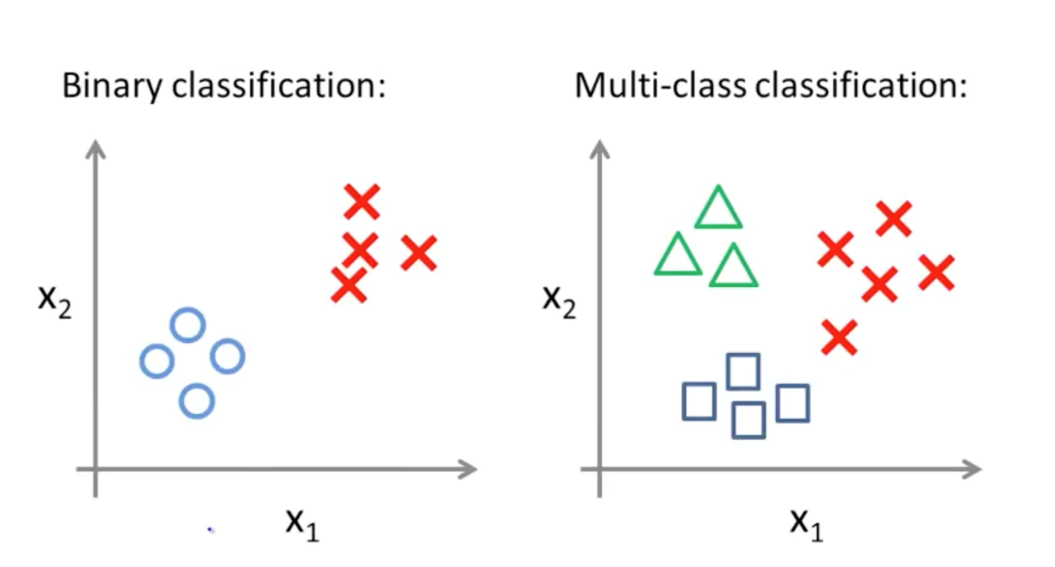
În învățarea supervizată, există un accent puternic pe utilizarea datelor etichetate pentru a învăța modelele, și acest tip de învățare este larg răspândit în diverse domenii și aplicații datorită abilității sale de a face predicții precise pe baza exemplelor de antrenament. Un mare dezavantaj al acestor tipuri de retele este faptul ca au nevoie ca setul de date sa fie etichetat, iar aceasta sarcina trebuie sa fie realizata de cineva. De asemenea, etichetarea trebuie sa se realizeze corespunzator, altfel reteaua va invata eronat.

**Reprezentarea grafica a retelelor neuronale supervizate**

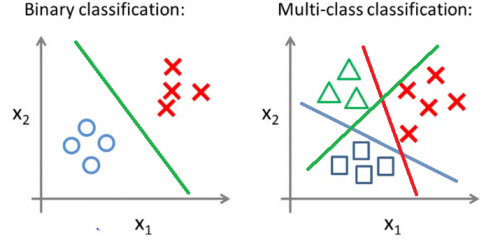
Relele neuronale pot fi reprezentate si grafic, de fapt ele se bazeaza pe principiul ca doua sau mai multe grupuri se vor situa in anumite zone pe un grafic in functie de o trasatura, vezi figura 12.

De altfel, weighturile si bias-ul pot fi traduse pe grafic. Acestea contribuie la despartirea claselor printr-o linie. Daca este un caz simplu, linia poate fi dreapta ca in figura 13. Fiecare greutate reprezinta un mic segment din graficul final, si valoarea greutatii reprezinta de fapt inclinatia segmentului. Bias-utranslateaza mai sus sau mai jos graficul.

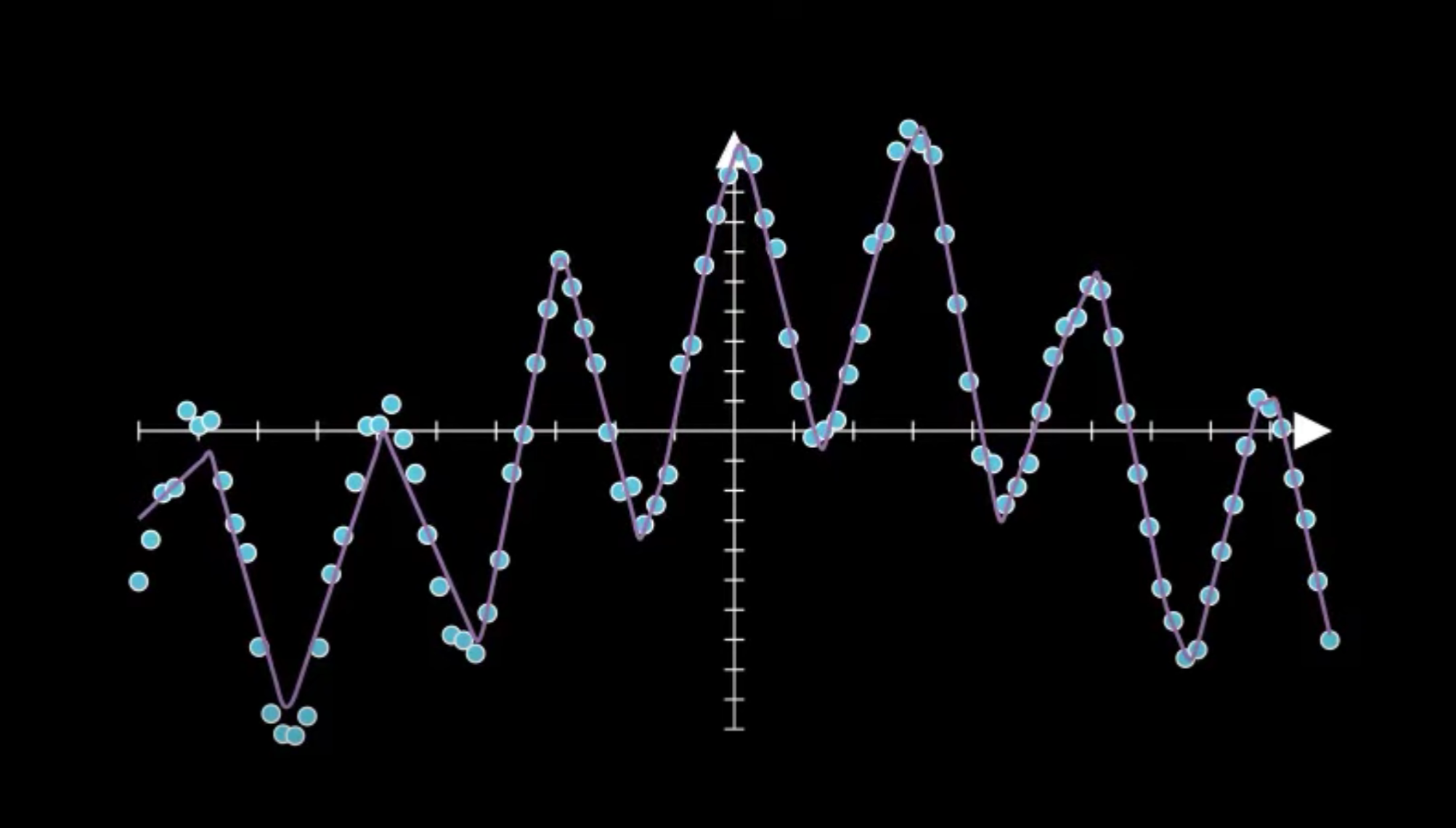
Pentru exemple complexe, se formeaza curbe complicate care reusesc sa desparta datele de intrare, vezi Figura 14. Este important ca graficul sa nu fie fix pe puncta, astfel incat reteaua sa poate generaliza si pe alt set de date.



*Figura 12 Clasificarea unui set de date*



*Figura 13. Clasificare unui set de date printr-o linie*

**

*Figura 14. Grafic complex al unui set de date*

## Invatarea nesupervizata

Unsupervised learning se concentreaza pe identificarea tiparelor dintr-un anumit set de date. Inputurile sunt clasificate in grupuri in functie de similitudinile dintre acestea, fara a fi etichetate. Aceasta metoda seamana mai bine cu modelul de recunoastere al creierului care formeaza pretutindeni conexiuni si asocieri intre date diferite. Acesti algoritmi pot identifica tipare ascunse pe care oamenii nu le pot observa in mod direct, contribuind la înțelegerea mai profundă a datelor. De exemplu, În domenii precum marketingul, învățarea nesupervizată poate fi utilizată pentru a segmenta publicul țintă în grupuri omogene.

De asemenea, aceasta tehnica poate fi folosita pentru a pregăti seturile de date pentru a le face mai ușor de prelucrat de catre alți algoritmi sau chiar pentru a crea noi seturi de date, care seamana cu cele deja existente – aceasta metoda este benefica mai ales in cazul in care nu se dispune de un set de date suficient de mare pentru antrenarea unei retele neuronale de la zero. De exemplu, sa presupunem ca vrem sa identificam daca pacientii au o boala sau nu in functie de o ecografie. Daca avem un set limitat cu pacienti sanatosi si un alt set de pacienti bolnavi, ne putem folosi de o astfel de tehnica de invatare nesupervizata numinca GAN (generative adversial networks) pentru a genera date asemanatoare din cele doua grupuri. Problema cu aceste date este ca e posibil sa iasa eronate si astfel si reteaua neoronala antrenata ulterior sa nu identifice cum trebuie seturi noi de date reale.

## Invatarea online

In mod obisnuit, se antreneaza o retea pe un set de date apoi se foloseste (inference). Se pune problema atunci cand apar noi date ce trebuie sa fie antrenate. De obicei, la o data stabilita, se da jos reteaua de pe server pentru a fi antrenata cu noile date apoi este verificata si urcata din nou.

O alta metoda ar fi ca reteaua sa invete tot timpul – online learning. Astfel, de fiecare data cand se introduce o noua informatie, reteaua isi si ajusteaza ponderile pentru noua data. In felul acesta se economiseste atat timp cand si spatiu, deoarece imaginea nu mai trebuie stocata si dureaza mult mai putin sa se ajusteze ponderile pentru un singur input.

Totusi, ca orice alt concept, are si dezavantaje, deoarece imaginile trebuie etichetate corespunzator imediat pentru a evita invatarea informatiei eronat. De asemenea, este posibil ca modelul să se îndepărteze de obiectivele inițiale de învățare dacă nu este gestionat corespunzător, mai ales dacă există o schimbare semnificativă în distribuția datelor.

# Alte tipuri de retele

## Retele neuronale recurente (Recurrent Neural Network)

Rețelele neuronale recurente (RNN) sunt un tip de arhitectură de rețea neurală specializată pentru lucrul cu date secvențiale sau de lungime variabilă. Această arhitectură permite modelelor să păstreze informații anterioare și să le utilizeze în momentul actual al procesării. RNN-urile sunt folosite într-o varietate de domenii, precum procesarea limbajului natural, recunoașterea vocală, traducerea automată, generarea de text și multe altele.

Principalele componente ale unei rețele neuronale recurente sunt, vezi figura 15:

* Celulă Recurentă (Recurrent Cell): Acesta este blocul de bază al RNN-urilor. O celulă recurentă primește o intrare la un moment dat și utilizează informațiile anterioare stocate în memorie pentru a genera o ieșire și pentru a actualiza starea sa internă.
* Secvențe de Date: RNN-urile lucrează cu secvențe de date, unde fiecare element din secvență este prelucrat în ordine. Aceasta poate fi o serie temporală, un șir de cuvinte într-o propoziție sau orice altă secvență de date.
* Stare Ascunsă (Hidden State): La fiecare pas în procesarea secvențială, RNN-ul menține o stare ascunsă care capturează informațiile anterioare din secvență. Această stare ascunsă este actualizată la fiecare pas și servește ca memorie a rețelei.

O imagine care conține text, diagramă, captură de ecran, Font

Descriere generată automat

*Figura 15. O retea recurenta care face o prezicere pe baza datelor anterioare*

Problema fundamentală a RNN-urilor clasice este cunoscută sub numele de "problema dispariției/explodarii gradientului" (vanishing/exploding gradient problem). Aceasta apare atunci când modelul trebuie să păstreze informații pentru un interval lung de timp. Pe măsură ce rețeaua propaghează gradientul în timp, acesta tinde să devină foarte mic, să dispară sau sa creasca exponential mai mare in functie de greutatea care este intre celule, ceea ce face dificilă actualizarea eficientă a parametrilor rețelei.

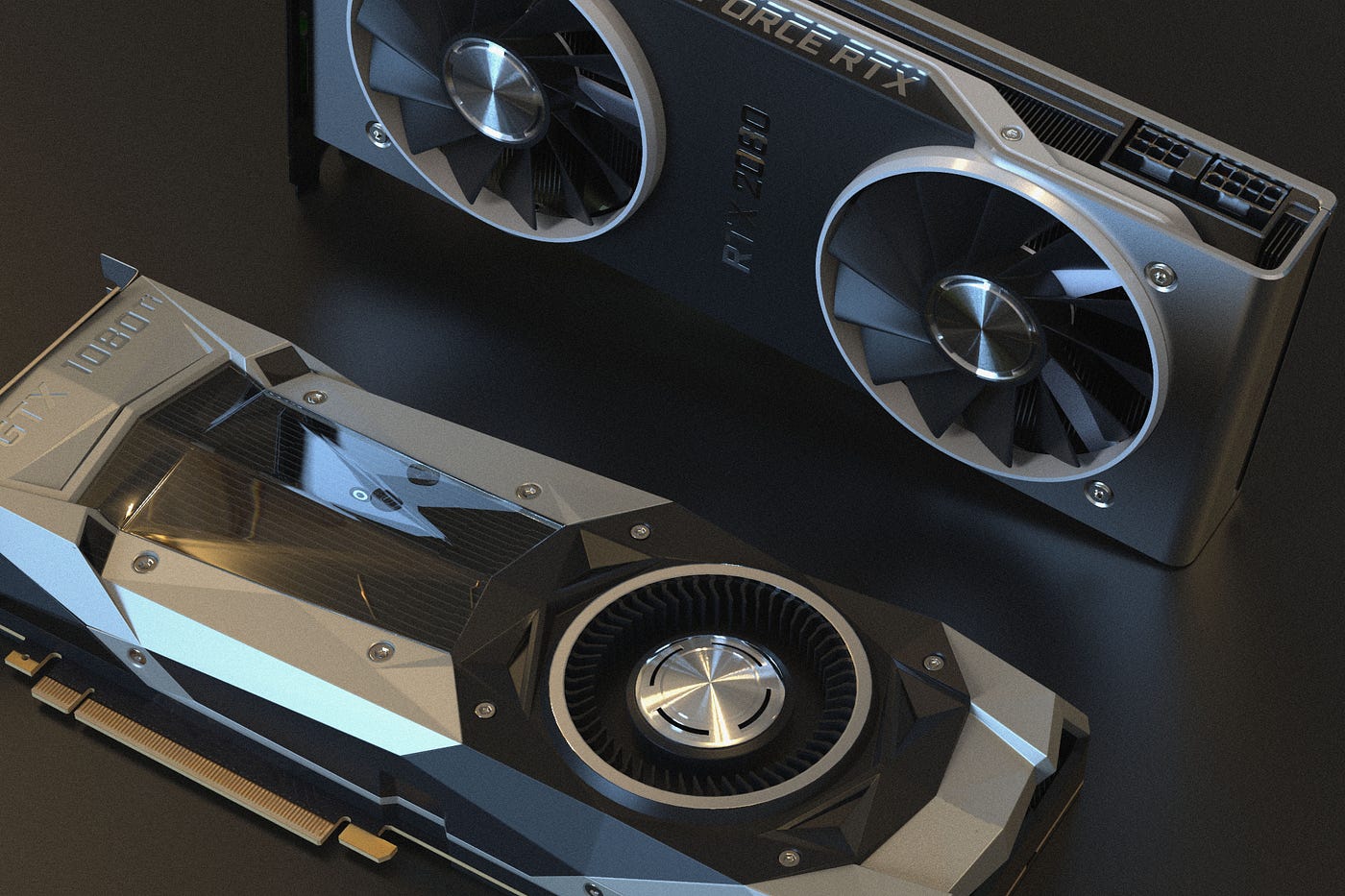
## Biblioteci dedicate pentru realizarea retelelor neuronale

In python exista cateva biblioteci care pot fi folosite in implementarea retelelor neuronale. Acestea au functii predefinite de calculare a straturilor de tip perceptron, convolutionale, pooling si multe altele. Faciliteaza crearea unor asfel de programe. Printre acestea se numara :

* TensorFlow
* PyTorch
* Keras
* Scikit-learn

## Acceleratoare de hardware

Datorită nevoii de a gestiona volumul mare de date și de a realiza calcule intensive asociate cu antrenarea și inferența modelelor s-au dezvoltat dispositive speciale operatiunilor din IA. Aceleratorii de hardware aduc beneficii semnificative în termeni de viteză și eficiență energetică, deoarece sunt specializați pentru anumite operațiuni, în loc să utilizeze arhitectura generală a unui CPU (Central Processing Unit).



*Figura 16. Unitati de procesare grafica*

Cele mai cunoscute acceleratoare pentru IA sunt unitatile de procesare grafica (GPU) – vezi figura 16, deoarece au capacitate mare de efectuarea operatilor in paralel. De asemenea, limbajul de programare CUDA de la NVIDIA facilitateaza accelerea calculelor acestor dispozitive, facand favorabile GPU-rile pentru aceste cerinte.

Se mai pot folosi si ASIC-uri (circuite integrate specific aplicatiei) care pot fi optimizate pentru sarcini precum inferenta sau FPGA-uri (Field-Programmable Gate Array) care sunt dispositive configurabile programabile, astfel fiind foarte versatile.

# Utilizari ale Inteligentei Artificiale

## Medicina

Medicina este un domeniu interesant cand vine vorba de implementarea Inteligentei Artificiale. Desi este mai complicat de implementat, deoarece nu sunt seturi de date prestabilite, iar folosirea datelor pacientilor trebuie facuta cu grija pentru a nu incalca niciun drept sau regula, beneficile sunt extraordinare. De exemplu, cercetatorii de la Royal Marsden Hospital and Institute of Cancer Research au implementat o retea care identifica de doua ori mai bine decat analizele de laborator o forma de cancer pe creier rara. In alte cazuri, astfel de algoritmi au redus identificarea unei boli de la 4 saptamani la doar cateva ore. Acest lucru sporeste semnificativ identificarea pacientilor bolnavi, chiar inca dinainte de aparitia semnelor externe in cazul unor boli, precum infectia cu Covid-19.

## Securitatea Cibernetica

Inteligenta artificiala ar putea fi folosita in identificarea amenintarilor sistemelor de calculatoare. Aceastea ar putea fi folosite de exemplu in firewalls sau pentru identificarea email-urilor spam, cu continut periculos. In cazul aplicatiilor pentru useri, IA ar putea analiza daca respective aplicatie dispune de puncte slabe de penetrare inainte ca aceasta sa ajunga pe internet.

## Social Media

Pe social media se va folosi inteligenta artficiala pentru crearea de continut, sub orice forma. Exista deja aplicatii care pot crea poze noi in diverse circumstante – eroi, creature mistice, birou – pe baza unor selfie-uri sau aplicatii care pot altera vocea unei personae, pentru a avea o anumite tonalitate sau a suna mai aproape de un personaj dintr-un film. De asemenea, exista un influencer spaniol (Aitana Lopez) care este generat de IA si se castiga 10000 de euro pe baza contentului acesta.

Acest subiect poate fi foarte delicat, deoarece in viitor s-ar putea sa nu se mai poata face diferenta dintre oamenii generati si cei reali pe social media.

## Va deveni Inteligenta Artificiala mai inteligenta decat oamenii?

In opinia noastra, Inteligenta Artificiala nu va depasi oamenii, cel putin nu in forma actuala. Fiintele au nevoie doar de cateva exemple pentru a putea clasifica diverse obiecte, fac mult mai rapid conexiuni si asocieri si au o capacitate mare de a generaliza – daca un copil vede o pisica maro si apoi una gri si este invatat ca aceastea sunt pisici, nu o sa ii fie greu sa isi de-a seama ca si o pisica neagra este tot pisica. In schimb, retelele neuronale au nevoie de cantitati gigantice de informatii si de multe ori nu ajung la performantele noastre de asocieri, asocieri care cateodata nici nu par legate intre ele, dar care formeaza inteligenta umana.

Aceasta este o unealta benefica in dezvoltarea tehnologiilor actuale, deoarece poate analiza multe date intr-un timp foarte scurt fata de oameni. Totusi, trebuie avut in vedere faptul ca aceste programe nu tin cont de context, spre deosebire de oamenii care înțeleg subtilități, nuanțe și aspecte emoționale ale comunicării care pot fi dificil de capturat de către sistemele de inteligență artificială actuale.

De asemenea, o problema mare cu acesti algoritmi este faptul ca ei consuma foarte multa energie. De exemplu costul antrenarii ChatGPT a fost cat energia anuala unui cartier (1000 de case in America). Si totusi, acest chatbot nu este atat de precis, da de multe ori rateuri (mai ales cand vine vorba de matematica) si pare ca “halucineaza”. Cine stie ce cantitati de resurse se vor consuma pentru modele mult mai avansate.

De aceea, consideram ca aceste unelte ar trebui folosite doar cand beneficiul utilizarii unui astfel de model este mai mare decat resurse si eforturile consumate. De asemenea, kucrand impreuna cu aceasta tehnologie, se pot deschide noi frontiere al cunoasterii.



# Bibliografie

<https://towardsdatascience.com/first-neural-network-for-beginners-explained-with-code-4cfd37e06eaf>

Brandon Rohrer, “How Deep Neural Networks Work” , https://youtu.be/ILsA4nyG7I0?si=-JgRaegRcFyttEN-

The Independent Code, “Neural Network from Scratch”, <https://www.youtube.com/watch?v=pauPCy_s0Ok>

The Independent Code, “Convolutional Neural Network from Scratch”, https://youtu.be/Lakz2MoHy6o?si=6rHpbW5iISPV0qQs

<https://www.oreilly.com/library/view/learning-tensorflow/9781491978504/ch04.html>

3Blue1Brown, “But what is a convolution?”, https://youtu.be/KuXjwB4LzSA?si=qhy2ckF-QrVD4o\_Z

<https://dennybritz.com/posts/wildml/understanding-convolutional-neural-networks-for-nlp/>

StatQuest, “Recurrent Neural Networks”, <https://www.youtube.com/watch?v=AsNTP8Kwu80>

Algorithmic Simplicity, “But what is a neural network REALLY?”, <https://www.youtube.com/watch?v=FBpPjjhJGhk>