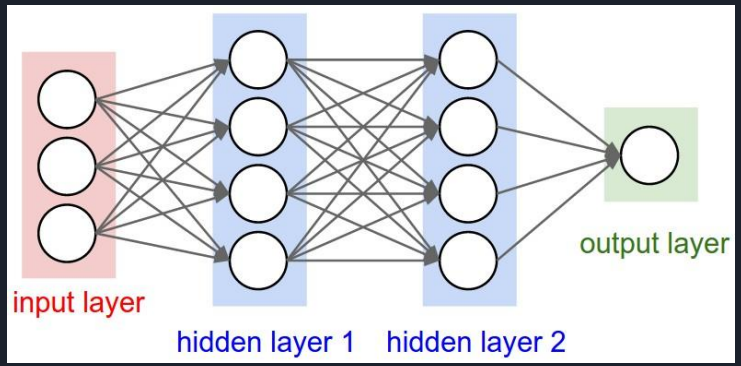
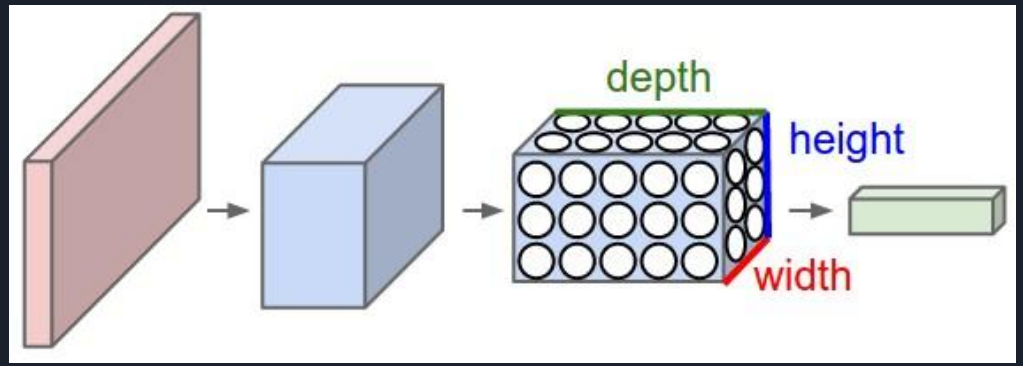
Învățare Automată în Arta Vizuală

Curs 4: Rețele Neurale Convoluționale

Rețele Neurale Convoluționale vs. Rețele Fully-Connected (recap.)





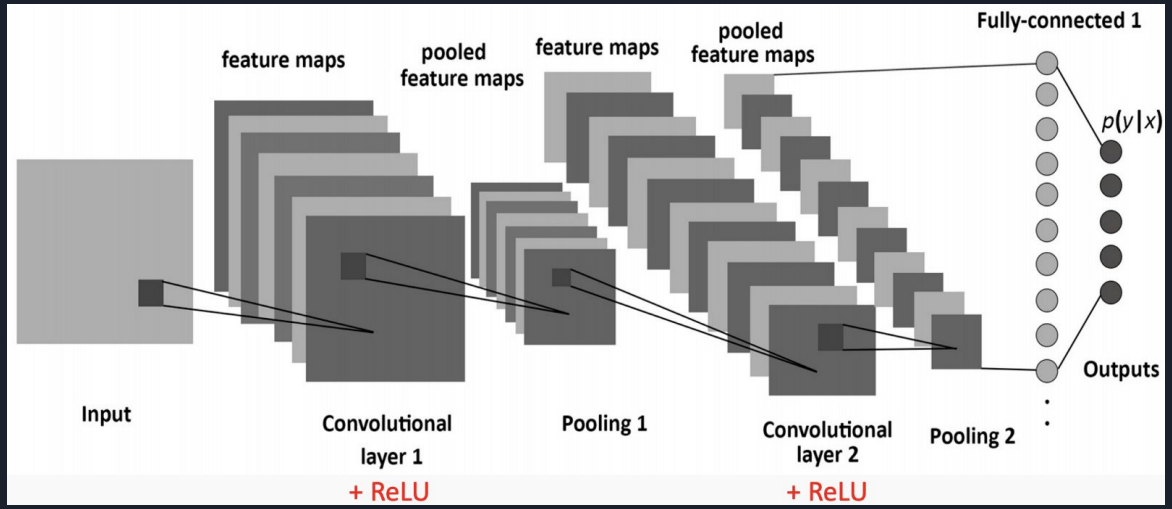
Fully-Connected:

- nu scalează pentru imagini mari
- ignoră informația spațială (poziția pixelilor și corelația cu vecinii)
- nu sunt invariabile la translație

CNN:

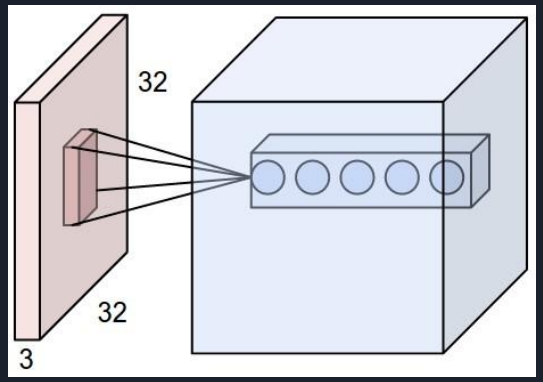
- poziția pixelilor și vecinii conțin informație semantică
- elementele de interes pot apărea oriunde în imagine

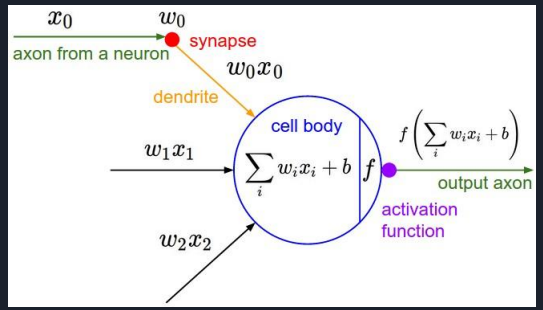
Rețele Neurale Convoluționale - Structură



- INPUT - structură 3D, conține valorile pixelilor imaginii
- CONV - calculează output-ul neuronilor conectați la regiuni din volumul anterior
- RELU - funcție de activare a fiecărui element
- POOL - reduce dimensiunea input-ului, de-a lungul axelor spațiale (lățime, înălțime)
- FC - calculează scorurile claselor; fiecare neuron este conectat la toți neuronii din volumul anterior

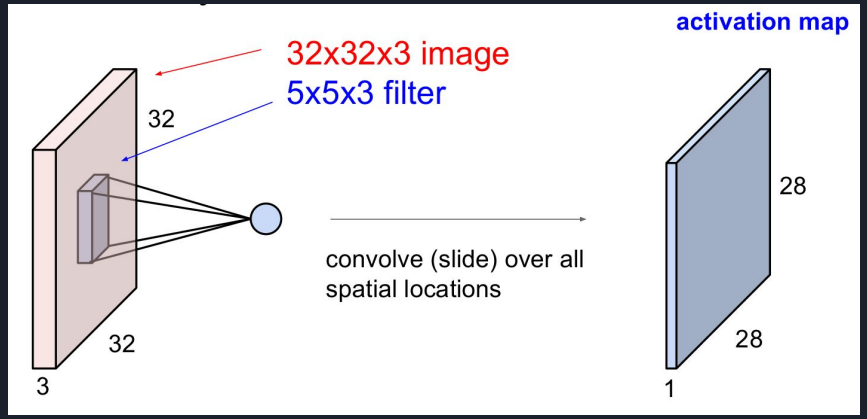
Convoluție

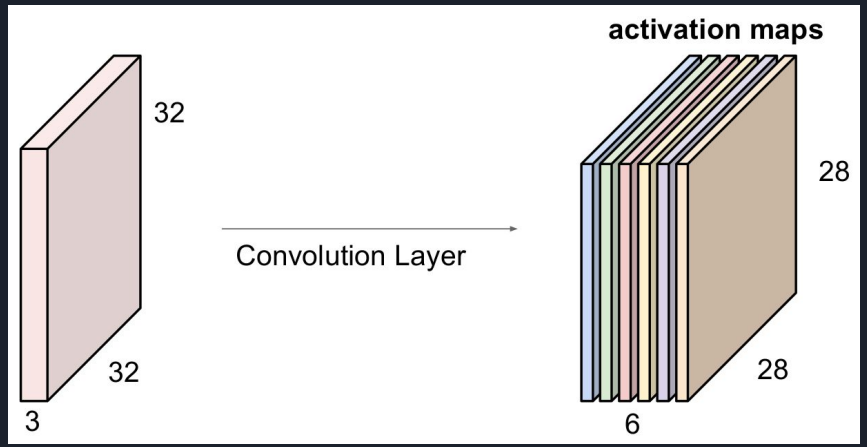




- filtru ce constă dintr-un set de parametri *învățabili*
- fiecare filtru este mic pe înălțime și lățime (dimensiuni standard: 1x1, 3x3, 5x5, 7x7), dar se extinde pe toată adâncimea volumului de intrare
- în timpul antrenării, se glisează fiecare filtru de-a lungul lățimii și înălțimii volumului anterior și se calculează produsul scalar între elementele sale și regiunea corespunzătoare din input, la fiecare poziție
- rezultatul convoluției este o hartă 2D (*activation map*), reprezentând răspunsul aplicării filtrului respectiv la fiecare poziție din input

Convoluție





- Un filtru produce un singur activation map
- Mai multe filtre produc mai multe activation maps - un volum similar cu imaginea de input
- Fiecare filtru învață un aspect (feature) diferit din input

Convoluție - Proprietăți

- **Conectivitate Locală:**
  - în cazul imaginilor de dimensiuni mari, este nefezabilă conectarea neuronilor cu toți neuronii din volumul anterior
  - neuronii sunt conectați doar la o regiune locală din volumul anterior (***receptive field***, echivalent cu dimensiunea filtrului)
  - dimensiunea în adâncime a filtrului de convoluție este **întotdeauna** egală cu adâncimea volumului de input (convoluția este locală pe lățime și adâncime, dar completă pe adâncime)

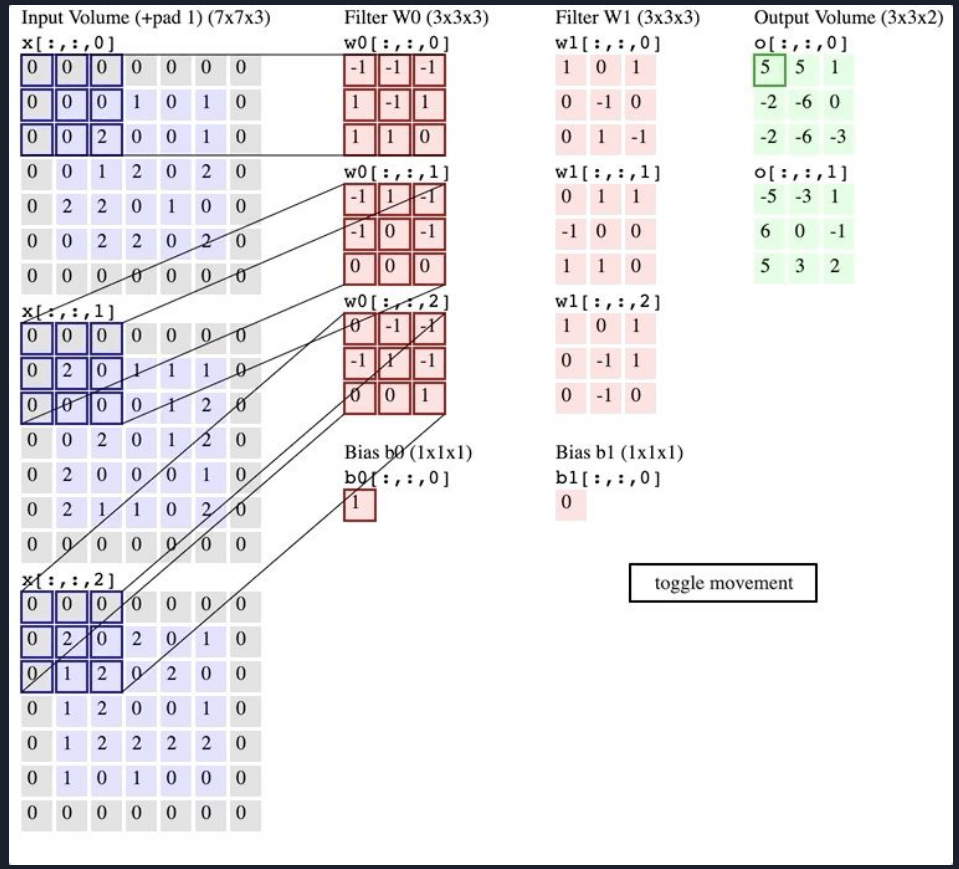
Convoluție - Proprietăți

- **Aranjarea spațială:**
  - 3 hiperparametri controlează dimensiunea volumului rezultat:
    - **depth** - corespunde numărului de filtre pe care vrem să le folosim, fiecare filtru activându-se la un feature diferit din input
    - **stride -** pasul cu care glisăm filtrul; pentru stride 1, glisăm filtrul cu câte un pixel la un moment dat; pentru stride 2, glisăm filtru cu câte 2 pixeli la un moment dat, rezultând un volum mai mic din punct de vedere spațial decât volumul de input
    - **padding** - păstrează dimensiunea spațială a volumului de input

Convoluție - Proprietăți

- **Partajarea parametrilor (parameter sharing):**
  - controlează numărul de parametri
  - numărul de parametri poate fi redus drastic dacă se face următoarea asumpție rezonabilă: dacă un feature este suficient de bun pentru a fi calculat la poziția (x1, y1), atunci ar trebui să fie util și la poziția (x2, y2). Astfel constrângem toți neuronii de la un anumit nivel din adâncime dintr-un strat de convoluție (e.g. un volum de dimensiune [55x55x96] are 96 de niveluri în adâncime) să folosească aceiași parametri.
  - asumpția anterioară este validă datorită structurii de **invarianță la translație** a imaginilor

Convoluție - Demo



Convoluție - Sumar

Acțiune

- aplică filtre pentru a extrage features
- filtrele au dimensiune spațială mică
- un singur bias pentru fiecare filtru
- aplică o funcție de activare pe fiecare element din output

Hiperparametri

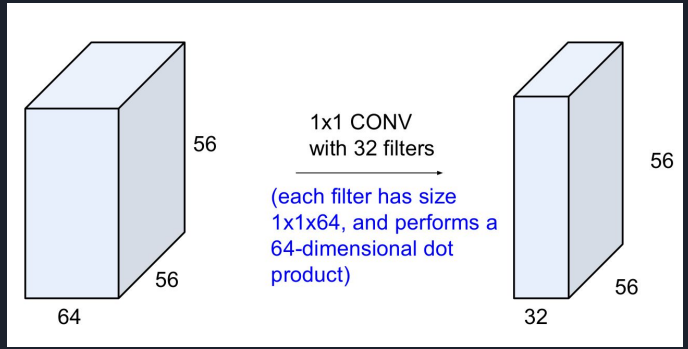
- numărul de filtre
- dimensiunea filtrelor (doar W și H; D este dat de input)
- funcția de activare
- stride, padding
- tipul de regularizare

I/O

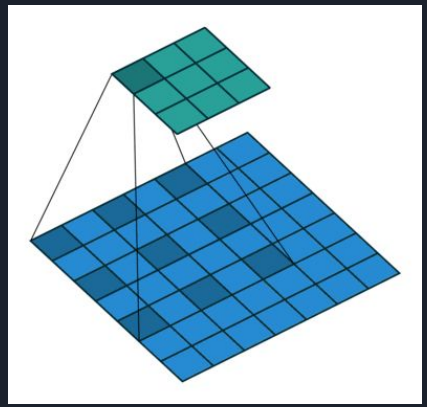
- input: structură 3D (set anterior de activări)
- output: structură 3D, câte o hartă 2D pentru fiecare filtru convoluțional

Convoluție - alte tipuri

- **Convoluția 1x1:**
  - afectează doar adâncimea volumului (nu altereză dimensiunile spațiale W și H)
  - rol în reducerea dimensionalității



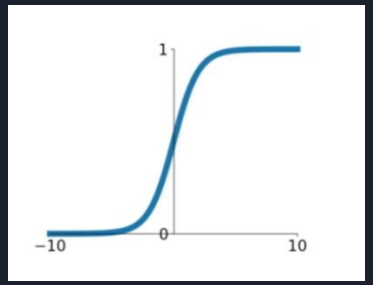
- **Convoluția dilatată:**
  - filtrele au spații între elemente
  - folosite pentru a mări receptive field-ul mult mai rapid



Comparație Funcții de Activare (1)

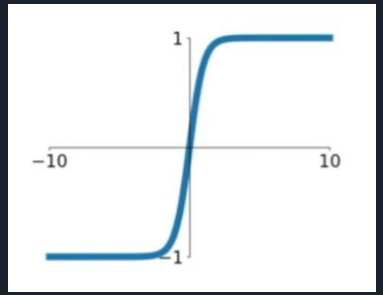
Sigmoid :

$$\sigma(x) = 1/(1 + e^{-x})$$



- Constrânge inputul în range-ul [0,1]
- Gradienții din zonele plate sunt 0
- Outputul nu este “centrat în zero”
- “exp()” este o functie computațional scumpă

Tanh :

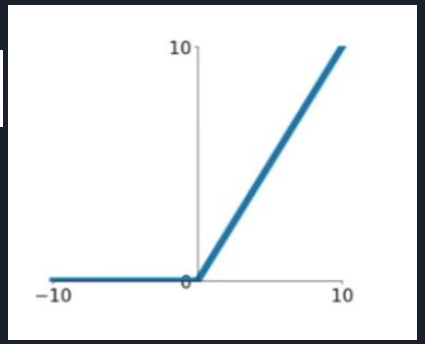


- Constrânge inputul în range-ul [-1,1]
- Outputul este “centrat în zero”
- Gradienții din zonele plate sunt 0

Comparație Funcții de Activare (2)

ReLU : (Rectified Liniar Unit)

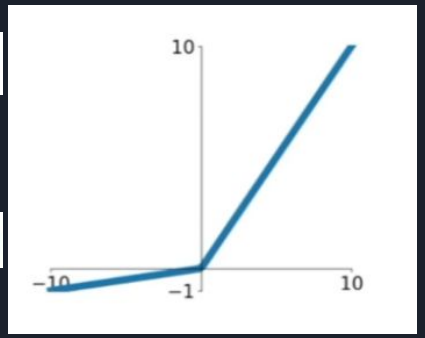
$$f(x) = \max(0,x)$$



- Gradienții sunt non-zero în regiunea “+” (input >0)
- Computațional eficientă
- Plauzibil biologic
- Outputul nu este “centrat în zero”

Leaky ReLU :

$$f(x) = \max(0.01x, x)$$



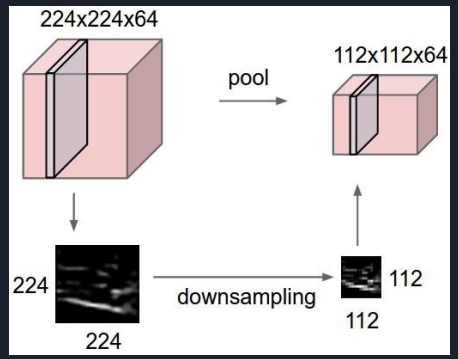
- Gradienții sunt non-zero pe tot spectrul inputului!

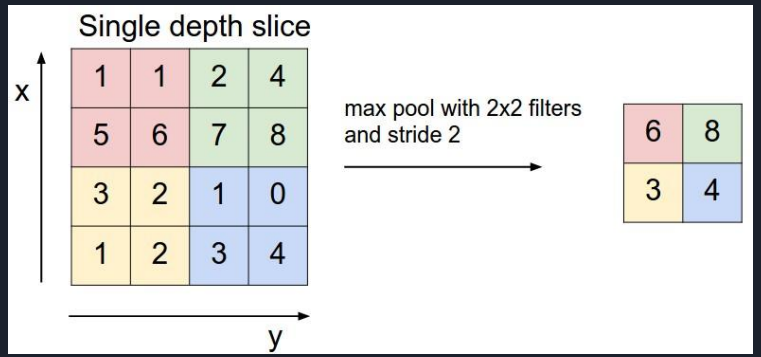
Parametric ReLU :

$$f(x) = \max(\alpha x, x)$$

- Alpha determină panta funcției pentru input <0
- Alpha este un parametru antrenabil

Pooling





- reduce progresiv dimensiunea spațială a input-ului pentru a reduce numărul de parametri și, implicit, de calcule în rețea și astfel controlează overfitting-ul.
- operează independent pe fiecare nivel din adâncime
- tipuri de pooling: MAX, AVG, L2-norm
- cea mai comună formă de aplicare este cu filtre de dimensiune 2x2 și stride 2, și înjumătățește lățimea și înălțimea input-ului și elimină 75% din activări.
- alternativă: convoluție cu stride 2

Pooling - Sumar

Acțiune

- reduce dimensionalitatea
- extrage maximul sau media dintr-o regiune
- abordare cu fereastră glisantă

Hiperparametri

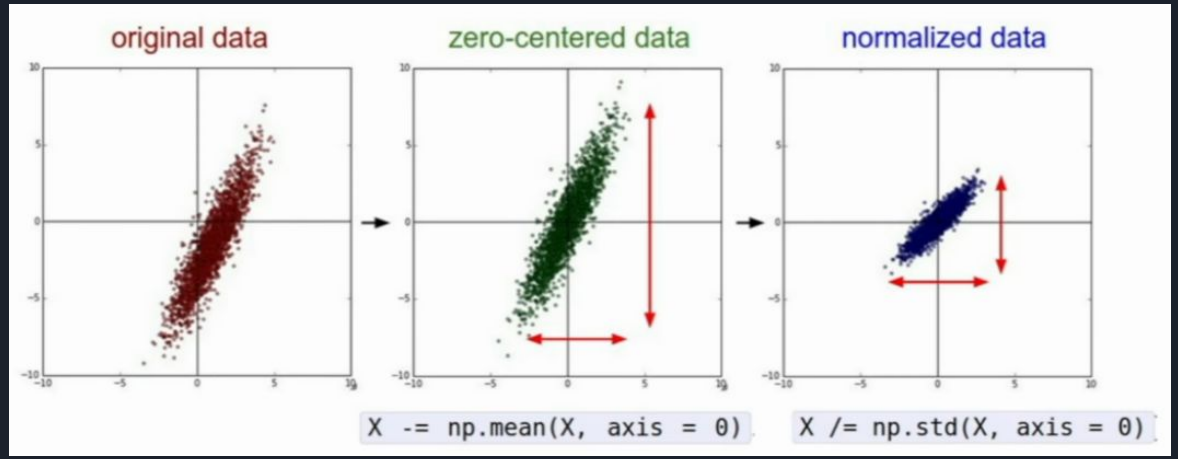
- stride
- dimensiunea ferestrei

I/O

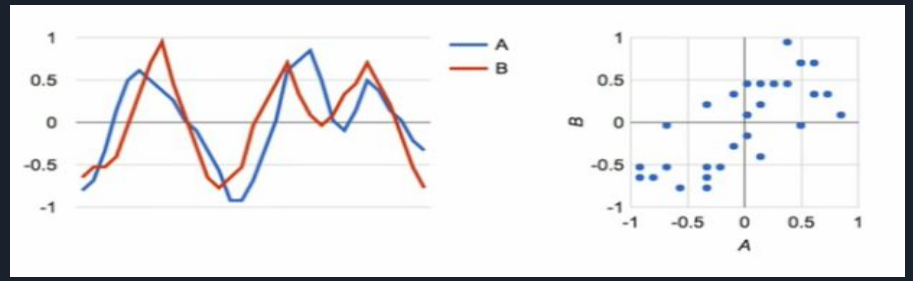
- input: structură 3D (set anterior de activări)
- output: structură 3D, câte o hartă 2D pentru fiecare filtru de pooling, cu dimensiuni spațiale reduse

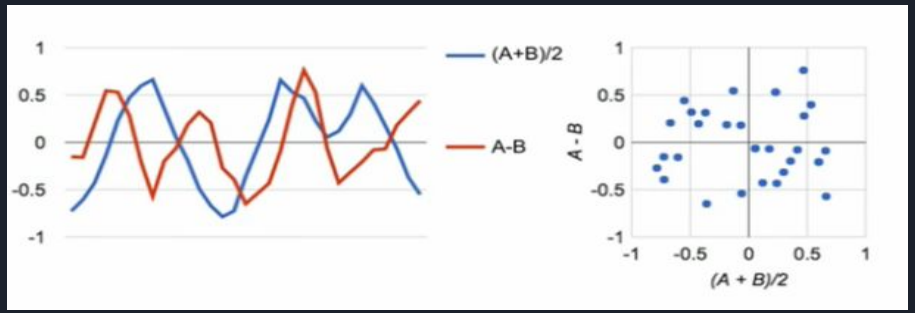
Preprocesarea Datelor (1)

- Centrarea datelor elimina posibilele “biasuri” pe care le poate avea poziția datelor in spațiu.
- De asemenea, după centrare, se pot observa, vizual, relații surprinzătoare între două distribuții diferite de date.
- Normalizarea volumului de input se face pentru ca toate caracteristicile acestuia să contribuie în mod egal (daca unele “features” sunt foarte puternic “observabile”, atunci contribuie prin valori mai mari decât restul spațiului de date, care poate fi la fel de important)



Preprocesarea Datelor (2) / “Data Whitening”





Caz ipotetic : Data points (A,B)

- Inițial : A se regăsește in [0 , 20] ; B în [0 , 2]

Aplicăm “Data whitening” : scădem media, împărțim la standard deviation => zero-centered, rescaled

Observație : A și B sunt foarte puternic corelate (evoluează in tandem, reflectă același fenomen)

- În cazul inițial, această corelație era greu de observat, iar activările provocate de A erau mult mai puternice decât cele provocate de B.

Decorelare : folosim un semnal de average (A+B)/2 și un semnal de diferență (A-B) => 2 data-seturi suficient de “diferite” încât sa fie folositoare, cu aceeași pondere, unei rețele neurale.

Batch Normalization (1)

Exemplu : “Albirea Datelor” se poate exprima matematic astfel :

new A	new B	=	A	B	x	0.05 0.61	0.12 -1.23	+	-1.45 0.12

[A’ B’] = [A B] * (matrice de scala / rotatie) + (translatie / shift)

[A’ B’] = [A B] * W + b

Batch Normalization:

- Un layer în rețea care determină automat care este “albirea” necesară a datelor.
- Re-normalizeaza outputul unui layer, înainte de activare

Observații in practică:

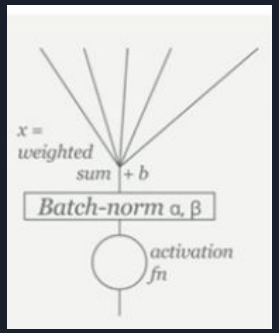
- Biasurile din layere nu mai sunt folositoare! (translația din BN va prelua acest rol)
- O metodă foarte eficientă de regularizare => se poate scoate DropOut-ul complet
- Activări curate => learning rate mai mare => training mai rapid.

Batch Normalization (2)

$$\hat{x} = \frac{x - avg_{batch}(x)}{stdev_{batch}(x) + \epsilon}$$

- Calculeaza media si varianța pe fiecare mini-batch
- Centrează si re-scalează outputul unui layer (pregătirea inputului pentru următorul layer)
- Are parametrii de scală si translație “antrenabili”

$$BN(x) = \alpha\hat{x} + \beta$$



- Dacă înlocuim **α** cu stdev(x) și **β** cu avg(x) obținem BN(x) = x
- Oferim rețelei neurale șansa de a lăsa setul de date **neafectat**, *dacă aceasta este cea mai bună soluție pentru reprezentarea lor.* (ne dorim să nu distrugem informația prin “albire”)
- BN se aplică înainte de funcția de activare!
- BN adaugă 2 grade de libertate suplimentare per neuron => computation cost.