



UNIVERSITATEA DIN BUCUREȘTI

FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

SPECIALIZAREA INTELIGENȚĂ ARTIFICIALĂ

Îmbunătățirea unui sistem de recomandare

---

LUCRARE DE DISERTAȚIE

COORDONATOR ȘTIINȚIFIC

CONF. DR. BOGDAN ALEXE

ABSOLVENT

ADRIAN ISPAS

BUCUREȘTI, ROMÂNIA

IUNIE 2019



# Abstract

Abstractul în limba română.

# Abstract

Abstractul în limba engleză.

# Cuprins

<b>Listă de figuri</b>	<b>6</b>
<b>Listă de tabele</b>	<b>7</b>
<b>1 Introducere</b>	<b>9</b>
1.1 Motivație . . . . .	9
1.2 Obiective propuse . . . . .	10
1.3 Structura lucrării . . . . .	10
<b>2 Fundamente teoretice</b>	<b>12</b>
2.1 Sisteme de recomandare . . . . .	12
2.1.1 Noțiuni generale . . . . .	12
2.1.2 Strategii de recomandare . . . . .	12
2.1.3 Funcții de loss . . . . .	14
2.2 Rețele neurale convoluționale . . . . .	18
2.2.1 Noțiuni generale . . . . .	18
2.2.2 VGG . . . . .	20
2.2.3 InceptionV3 . . . . .	21
2.2.4 ResNet . . . . .	22
2.2.5 NASNet . . . . .	25
2.3 Clustere . . . . .	28
2.3.1 Noțiuni generale . . . . .	28
2.3.2 K-nearest neighbors . . . . .	29
2.3.3 Metrice de evaluare a clusterelor . . . . .	30
<b>Bibliografie</b>	<b>33</b>

# Listă de figuri

2.1	Filtrarea colaborativă . . . . .	13
2.2	Filtrarea bazată pe conținut . . . . .	14
2.3	Matricea de interacțiuni . . . . .	15
2.4	Setul de antrenare . . . . .	16
2.5	Procedura de învățarea BPR . . . . .	16
2.6	Online WARP Loss Optimization . . . . .	18
2.7	Exemplu rețea convoluțională . . . . .	19
2.8	Exemplu de filtru aplicat peste input . . . . .	19
2.9	Exemplu de pooling . . . . .	20
2.10	Configurații VGG . . . . .	21
2.11	Factorizarea în filtre convoluționale mici . . . . .	22
2.12	Factorizarea spațială în convoluții asimetrice . . . . .	22
2.13	Clasificator auxiliar . . . . .	23
2.14	Reducerea eficientă a dimensiunii . . . . .	23
2.15	Arhitectura InceptionV3 . . . . .	24
2.16	Învățarea reziduală . . . . .	24
2.17	Rețeaua ResNet . . . . .	26
2.18	NAS . . . . .	27
2.19	Celule normale și de reducere . . . . .	27
2.20	Exemplu clustere . . . . .	28
2.21	Exemplu 1 de clasificare cu kNN . . . . .	31
2.22	Exemplu 2 de clasificare cu kNN . . . . .	31

## Listă de tabele





# Capitolul 1

## Introducere

### 1.1 Motivație

Volumul de date crește semnificativ de la an la an astfel până în 2020 se estimează că pentru fiecare persoană de pe planetă vor fi creați în fiecare secundă 1.7 MB de date, ceea ce înseamnă peste 13 milioane de GB creați în fiecare secundă în lume.

În 2018 în fiecare minut se vizionau peste 97 de mii de ore de conținut pe Netflix. Peste 4.3 milioane de videoclipuri erau vizionate pe Youtube. Pe Spotify se ascultau 750 de mii de melodii, iar Amazon pregătea peste o mie de pachete [1].

În România, Netflix pune la dispoziție 575 de filme și 208 seriale. În Regatul Unit sunt disponibile 2425 de filme și 542 de seriale, iar în Statele Unite Ale Americii sunt disponibile 2942 de filme și 629 de seriale [2].

Amazon oferă cumpărătorilor o gamă cu un total de peste 119 milioane de produse, dintre care 44.2 milioane de cărți, 10.1 milioane de electronice sau 4.5 milioane de produse realizate manual [3].

În primă fază, cu cât volumul de date pus la dispoziție de o platforma este mai mare cu atât este mai mare și necesitatea unui sistem de recomandare care să vină în ajutorul utilizatorului final pentru a explora gama de produse oferită de respectiva platformă. Ulterior, acel sistem de recomandare se vrea a fi îmbunătățit astfel încât să ofere fiecărui utilizator o experiență cât mai personalizată prin care se recomande, în cazul platformelor de streaming video, conținut relevant pentru a fi consumat de utilizatorul final, sau în cazul platformelor de e-commerce, produse pe care utilizatorul ar fi dispus să le cumpere.

În majoritatea cazurilor sistemele de recomandare se bazează pe metadatele utilizatorilor, precum: regiunea, vârsta, genul, ce alte produse a accesat sau cumpărat și metadatele produselor: categoria din care face parte, ratingul acestuia. La acestea se pot adăuga și alte informații precum: ce alte produse a apreciat un alt user cu un profil asemănător.

## 1.2 Obiective propuse

În majoritatea cazurilor primul contact pe care îl avem cu un clip de pe Youtube, cu un film sau serial de pe Netflix sau un produs de pe Amazon este contactul vizual cu imaginea de prezentare a acelui produs.

Astfel, prezenta lucrare de disertație are drept obiectiv principal introducerea în sistemul de recomandare de informații vizuale extrase din imaginile de prezentare ale produselor. Informațiile vizuale sunt reprezentate de clusterelor create peste imaginile asociate produselor. Fiecare produs are o imagine de prezentare, iar fiecare imagine are un cluster caruia îi aparține din intervalul  $[1, N]$  unde  $N$  este corelat cu numărul de categorii de produse din baza de date pe care se execută optimizarea.  $N$  poate fi ales și pe baza altor raționamente.

Scopul final al acestei abordări fiind acela de a observa evoluția metricilor de evaluare, în cazul nostru acuratețea și precizia@k, atunci când informația vizuală este introdusă într-un sistem de recomandare, fiind singura informație prezentă exceptând matricea de interacțiuni, dar și cum se comportă un sistem de recomandare când primește această informație împreună cu alte informații, spre exemplu categoria unui articol.

## 1.3 Structura lucrării

Prezenta lucrare de disertație începe prin detalierea fundamentelor teoretice în capitolul II unde sunt prezentate teoriile ce stau la baza realizării acestei lucrări.



# Capitolul 2

## Fundamente teoretice

### 2.1 Sisteme de recomandare

#### 2.1.1 Noțiuni generale

Sistemele de recomandare au scopul de oferi sugestii de articole utilizatorilor unei platforme pe baza unor strategii. Un sistem de recomandare poate folosi una sau mai multe strategii de recomandare.

În cazul în care se folosesc cel puțin două strategii, sistemul de recomandare devine un sistem de recomandare hibrid. Prin folosirea mai multor strategii se urmărește ca fiecare strategie să vină în completarea celorlalte cu avantajele sale.

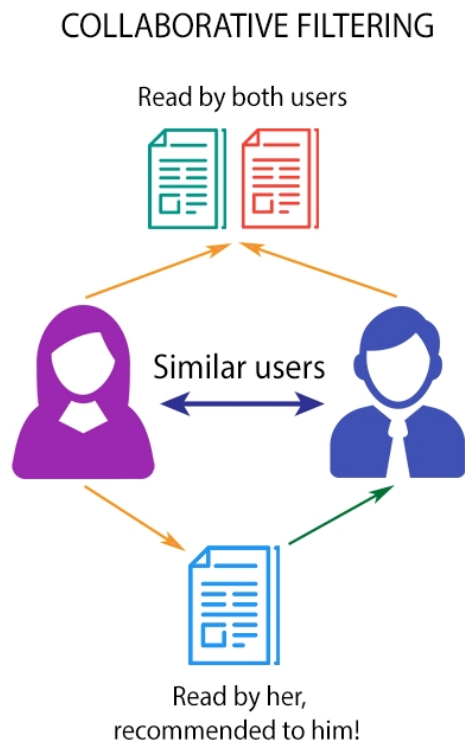
De cele mai multe ori, în implementarea unui sistem de recomandare, se folosește tehnica de filtrare colaborativă împreună cu o altă strategie de recomandare [4].

#### 2.1.2 Strategii de recomandare

##### Filtrarea colaborativă

Filtrarea colaborativă se bazează pe faptul că utilizatorii care au în prezent preferințe similare vor avea și în viitor preferințe destul de similare. Această abordare folosește ratingurile pe care le dau utilizatorii sau oricare altă formă de a da un feedback, îmi place/nu îmi place, pentru a identifica preferințele comune dintre grupurile de utilizatori. Odată identificate preferințele se generează recomandări pe baza similarităților dintre utilizatori.

Dezavantajul acestei strategii apare în momentul în care în sistem intră un nou utilizator. Datorită faptului că utilizatorul este nou, sistemul nu are un istoric al preferințelor lui, iar în consecință nu îl poate asigura unui grup de utilizatori pe baza preferințelor [4].



**Figura 2.1:** Filtrarea colaborativă. Imagine preluată din [5].

### Filtrarea bazată pe conținut

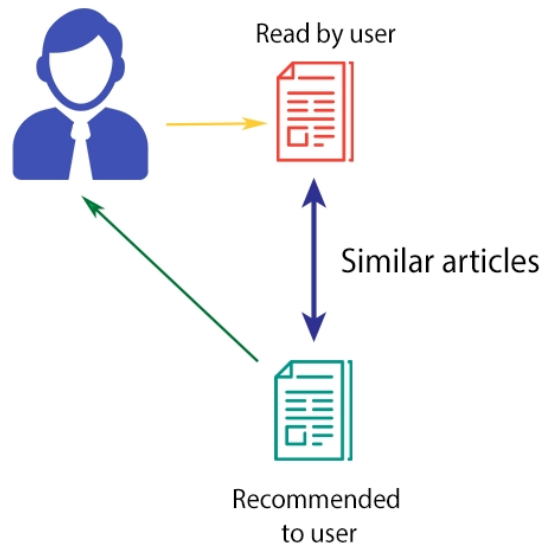
Filtrarea bazată pe conținut pleacă de la premisa că utilizatorii cărora le-au plăcut articole definite de anumite atribute în trecut, vor aprecia aceleași tip de articole și în viitor. Această abordare folosește atributele articolelor pentru a le compara cu profilul utilizatorilor și a oferi recomandări. Calitatea recomandărilor create folosind această strategie este influențată de setul de atribute ales pentru articole.

Similar cu filtrarea colaborativă și filtrarea bazată pe conținut prezintă dezavantaje în momentul în care în sistem intră un nou utilizator fără istoric [4].

### Filtrarea demografică

Filtrarea demografică folosește atribute precum vârsta, genul, educația, etc. pentru a identifica categoriile de utilizatori. Nu prezintă dezavantaje atunci când apar noi utili-

## CONTENT-BASED FILTERING



**Figura 2.2:** Filtrarea bazată pe conținut. Imagine preluată din [5].

zatori în sistem și nu se folosește de ratinguri, sau alt sistem de feedback, pentru a face recomandări.

Dezavantajul este reprezentat de faptul că procesul de colectare al datelor demografice poate fi îngreunat de legislație fapt ce reprezintă o limitare a acestei metode [4].

### Filtrarea bazată pe cunoștințe

Filtrarea bazată pe cunoștințe folosește cunoștințele despre utilizatori și articole pentru a spune ce articole îndeplinesc cerințele utilizatorilor și generează recomandări în consecință. Filtrare bazată pe cunoștințe are la bază constrângeri și este capabilă să recomande chiar și articole complexe care nu sunt cumpărate atât de des, precum mașini sau case [4].

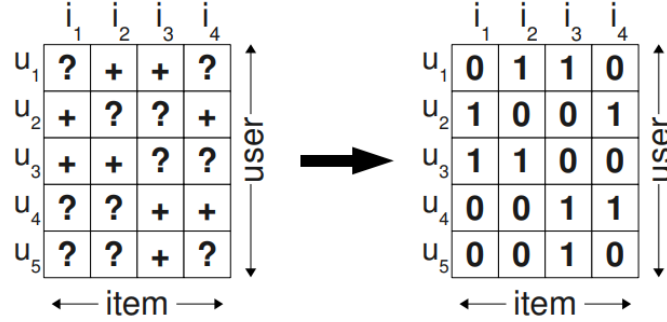
### 2.1.3 Funcții de loss

#### BPR: Bayesian Personalised Ranking

Este o metodă ce se bazează pe feedback implicit (click-uri, ratinguri, achiziții). Există multe metode ce se bazează pe acest feedback implicit, precum matrix factorization (MF),

k-nearest-neighbor (kNN), însă acestea nu sunt optimizate pentru ranguri. Metoda de învățare este bazată pe gradientul descendent. Metoda este recomandată atunci când se dorește optimizarea acurateții.

Definim în continuare  $U$  ca fiind mulțimea de utilizatori și  $I$  ca fiind mulțimea de articole. Feedback-ul implicit este reprezentat de mulțimea  $S \subseteq U \times I$ . De asemenea, definim  $I_u^+ := \{i \in I : (u, i) \in S\}$  și  $U_i^+ := \{u \in U : (u, i) \in S\}$ .



**Figura 2.3:** Matricea de interacțiuni, mulțimea  $S$ . Imagine preluată din [9].

O abordare uzuală pentru recomandarea de articole este să fie prezis scorul  $\hat{x}_{ui}$  care să reflecte preferința utilizatorului  $u$  pentru articolul  $i$ . Apoi fiecare articol primește un rang după sortarea scorurilor.

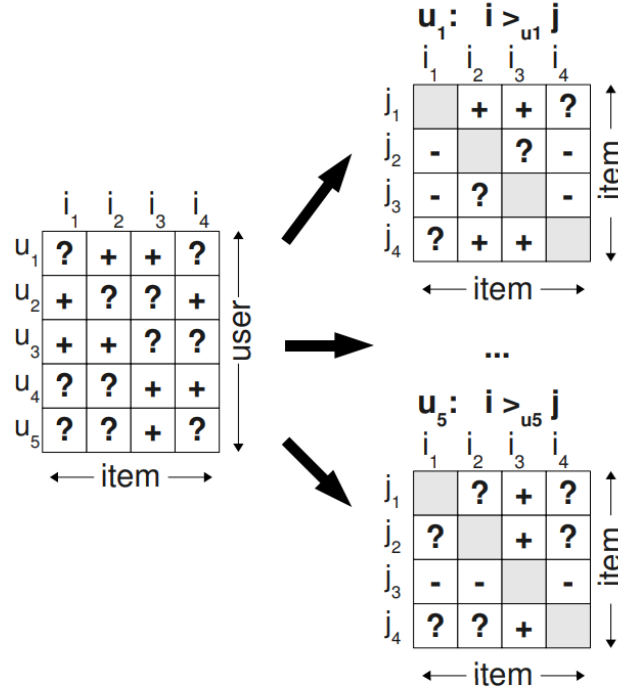
Setul de antrenare este definit de mulțimea  $D_S := \{(u, i, j) | i \in I_u^+ \wedge j \in I \setminus I_u^+\}$  unde  $(u, i, j)$  înseamnă că utilizatorul  $u$  preferă articolul  $i$  în detrimentul articolului  $j$ .

Criteriul de optimizare pentru pentru rangurile personalizate este definit după cum urmează:

$$BPR - OPT := \sum_{(u, i, j) \in D_S} \ln \sigma(\hat{x}_{uij}) - \lambda_{\Theta} \|\Theta\|^2 \quad (2.1)$$

unde  $\sigma$  este funcția sigmoid,  $\sigma(x) := \frac{1}{1+e^{-x}}$ ,  $\Theta$  reprezintă vectorul parametru al modelului care definește interacțiunea dintre utilizatorul  $u$ , articolul  $i$  și articolul  $j$ , iar  $\lambda_{\Theta}$  reprezintă parametrii de regularizare.

Cu aceste definiții putem defini și procedura de învățare a  $BPR$  după cum urmează.



**Figura 2.4:** Setul de antrenare. + reprezintă articolele  $i$  pe care utilizatorul le preferă în locul articolelor  $j$ , – utilizatorul preferă articolele  $j$  în loc de  $i$ , iar ? reprezintă lipsa informației despre acea interacțiune. Imagine preluată din [9].

```

1: procedure LEARNBPR( $D_S, \Theta$ )
2:   initialize  $\Theta$ 
3:   repeat
4:     draw  $(u, i, j)$  from  $D_S$ 
5:      $\Theta \leftarrow \Theta + \alpha \left( \frac{e^{-\hat{x}_{uij}}}{1+e^{-\hat{x}_{uij}}} \cdot \frac{\partial}{\partial \Theta} \hat{x}_{uij} + \lambda_{\Theta} \cdot \Theta \right)$ 
6:   until convergence
7:   return  $\hat{\Theta}$ 
8: end procedure

```

**Figura 2.5:** Optimizarea modelului bazată metoda gradientului descendent cu parametrul de învățare  $\alpha$  și regularizarea  $\lambda_{\Theta}$ . Imagine preluată din [9].

### WARP: Weighted Approximate-Rank

Această metodă își are originile în procesarea imaginilor și anume pentru un set de reprezentări ale unor imagini  $x \in R^d$  și pentru un set de reprezentări ale unor adnotări  $i \in \Upsilon = \{1, \dots, Y\}$  - indici într-un dicționar cu posibile adnotări, metoda învață să mapeze imagini din spațiul reprezentărilor într-un spațiu comun  $R^D$



$$\Phi_I(x) : R^d \rightarrow R^D \quad (2.2)$$

în același timp învățând și mapări pentru adnotări în același spațiu

$$\Phi_W(i) : 1, \dots, Y \rightarrow R^D \quad (2.3)$$

Scopul principal fiind acela de a oferi ranguri posibilelor adnotări pentru o imagine dată astfel încât cel mai mare rang să descrie cel mai bine conținutul semnificativ al imaginii.

Modelul folosit este următorul:

$$f_i(x) = \Phi_W(i)^T \Phi_I(x) \quad (2.4)$$

Metoda învață să producă ranguri optimizate pentru primele adnotări din listă, ceea ce înseamnă că optimizează precizia@k.

În ceea ce privește funcția de eroare definim:  $f(x) \in R^Y$  ce produce un scor pentru fiecare etichetă și unde  $f_i(x)$  este valoarea etichetei  $i$ . Definim funcția de eroare pentru ranguri ca fiind:

$$err(f(x), y) = L(rank_y(f(x))) \quad (2.5)$$

unde  $rank_y(f(x))$  este rangul etichetei corecte data de  $f(x)$ :

$$rank_y(f(x)) = \sum_{i \neq y} I(f_i(x) \geq f_y(x)) \quad (2.6)$$

unde  $I$  este funcția indicator, iar  $L(\cdot)$  transformă rangul în penalizare

$$L(k) = \sum_{j=1}^k \alpha_j, \quad cu \quad \alpha_1 \geq \alpha_2 \geq \dots \geq 0. \quad (2.7)$$

$L(\cdot)$  poate lua diferite forme în funcție de ce se dorește a se optimiza:  $\alpha_j = \frac{1}{Y-1}$  optimizează rangul mediu,  $\alpha_j = 1$  și  $\alpha_{j>1} = 0$  optimizează proporția de ranguri corecte aflate în top, iar valorile mari ale lui  $\alpha$  optimizează primele  $k$  în lista de ranguri[8].

Cu definițiile prezentate mai sus putem descrie algoritmul acestei metode după cum urmează.

---

**Algorithm 1** Online WARP Loss Optimization

---

**Input:** labeled data  $(x_i, y_i), y_i \in \{1, \dots, Y\}$ .  
**repeat**  
  Pick a random labeled example  $(x_i, y_i)$   
  Let  $f_{y_i}(x_i) = \Phi_W(y_i)^\top \Phi_I(x_i)$   
  Set  $N = 0$ .  
  **repeat**  
    Pick a random annotation  $\bar{y} \in \{1, \dots, Y\} \setminus y_i$ .  
    Let  $f_{\bar{y}}(x_i) = \Phi_W(\bar{y})^\top \Phi_I(x_i)$   
     $N = N + 1$ .  
  **until**  $f_{\bar{y}}(x_i) > f_{y_i}(x_i) - 1$  or  $N \geq Y - 1$   
  **if**  $f_{\bar{y}}(x_i) > f_{y_i}(x_i) - 1$  **then**  
    Make a gradient step to minimize:  
       $L(\lfloor \frac{Y-1}{N} \rfloor) |1 - f_{y_i}(x_i) + f_{\bar{y}}(x_i)|_+$   
    Project weights to enforce constraints (2)-(3).  
  **end if**  
**until** validation error does not improve.

---

**Figura 2.6:** Online WARP Loss Optimization. Imagine preluată din [8].

## 2.2 Rețele neurale convoluționale

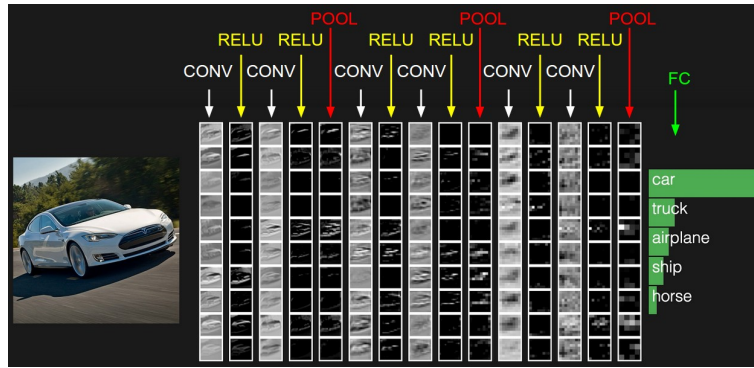
### 2.2.1 Noțiuni generale

Rețelele neurale convoluționale sunt foarte similare cu rețelele neurale fiind formate din neuroni ce învață ponderi ( $w$ ) și biasuri ( $b$ ). Scopul rețelei convoluționale este de a primi o imagine la input și de a scoate la output un scor pentru fiecare clasă ce corespunde imaginii.

Spre exemplu, la input se dă o imagine cu un autovehicul, iar rețeaua convoluțională poate spune că în imagine este o mașină în proporție de 80%, un camion în proporție de 10%, un avion în proporție de 6%, o barcă în proporție de 3% sau un cal în proporție de 1%.

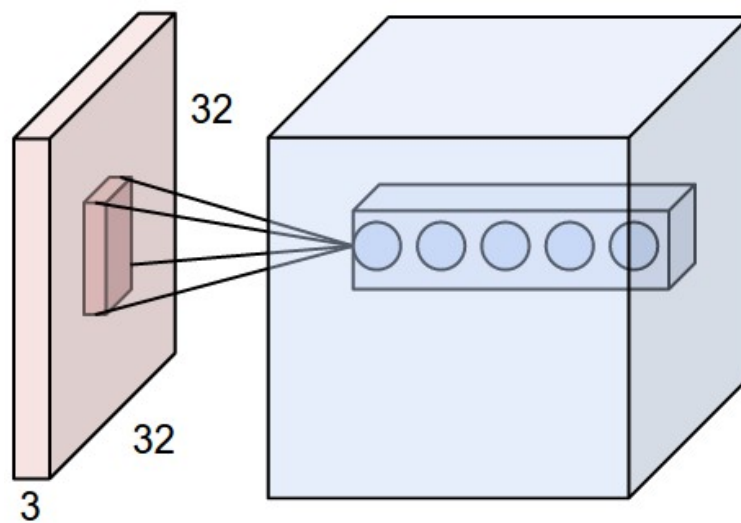
Rețelele convoluționale sunt compuse dintr-o secvență de layere ce poate fi împărțită în trei tipuri principale de layere [7]:

1. Convolutional Layer este layerul de bază într-o rețea convoluțională. Parametrii acestui layer sunt reprezentați de filtre învățabile, unde fiecare filtru reprezintă o mică bucată din imaginea de input. De exemplu, un filtru pentru acest layer poate avea dimensiunea de  $5 \times 5 \times 3$ , dimensiune ce reprezintă faptul că se iau 5 pixeli pe lățime și înălțime cu o adâncime de 3, unde adâncimea reprezintă canalele RGB. În continuare se glisează fiecare filtru peste input și se compune produsul



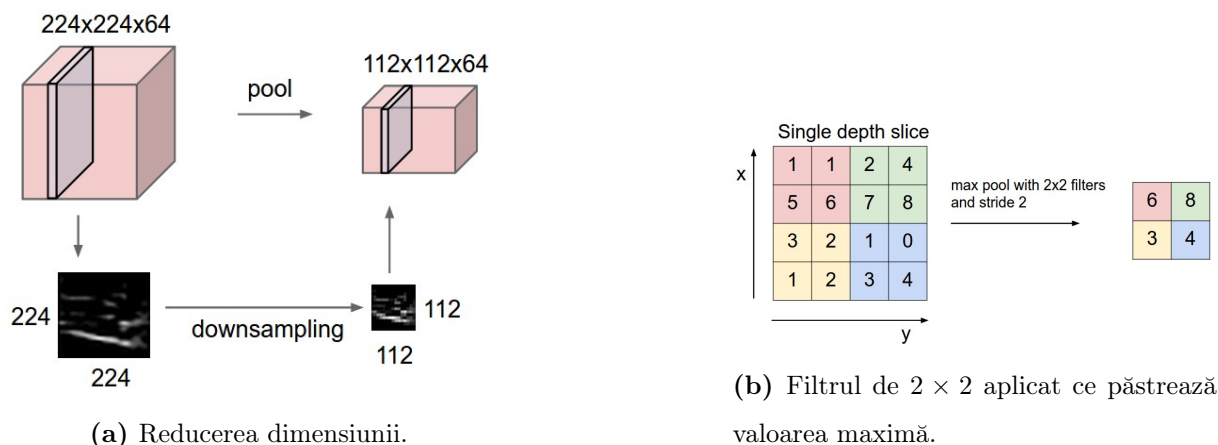
**Figura 2.7:** Exemplu de rețea convoluțională care primește la input o imagine și produce la output o listă de clase ce pot descrie imaginea de input. Imagine preluată din [7].

dintre filtre și input la fiecare poziție. În urma acestei operații se produce un vector de activare 2-dimensional care reprezintă răspunsul filtrului la fiecare poziție. Altfel spus, rețeaua va învăța filtre care se activează atunci când sunt prezente anumite tipuri de caracteristici, precum culoarea sau orientarea.



**Figura 2.8:** Exemplu de filtru aplicat peste input într-un layer convoluțional. Imagine preluată din [7].

2. Pooling Layer reprezintă o practică des folosită între mai multe layere convoluționale succesive. Această operație reduce numărul de parametrii (dimensiunea), computațiile din rețea și controlează overfittingul. Se execută independent pe fiecare nivel al adâncimii unui input și pastrează valoarea maximă a acelei zone. Rezultatul este o zonă de caracteristici mai mică dar care păstrează cea mai relevantă statistică.
3. Fully-Connected Layer este stratul în care caracteristicile sunt vectorizate pentru a



**Figura 2.9:** Exemplu de pooling. Imagine preluată din [7].

putea fi folosite.

### 2.2.2 VGG

VGG este o arhitectură de rețea cu filtre convoluționale foarte mici, de dimensiune  $3 \times 3$  și care poate avea o adâncime a straturilor de ponderi de 16 - 19.

În ceea ce privește arhitectura, inputul în rețeaua convoluțională este de dimensiune fixă și anume  $224 \times 224$  imagine RGB. Mai departe, imaginea este trecută printr-un set de straturi convoluționale unde sunt utilizate filtre de dimensiune mică,  $3 \times 3$  - fiind cea mai mică dimensiune ce poate captura noțiunile de stânga/dreapta, sus/jos, centru. Într-una dintre configurații se utilizează un filtru convoluțional de dimensiune  $1 \times 1$ . Pasul în straturile convoluționale este fixat la 1 pixel.

Poolingul este compus din cinci straturi de max-pooling care urmează după unele straturi convoluționale. Max-poolingul este calculat cu ferestre de  $2 \times 2$  pixel și cu pas de 2 pixeli.

Odată trecută imaginea prin straturile convoluționale și cele de pooling ajunge în trei straturi fully-connected. Primele două straturi au câte 4096 de canale fiecare, iar al treilea are 1000 de canale. Canalele celui de-al treilea strat sunt asociate claselor, fiecare canal reprezintă o clasă.

Ultimul strat din rețea este un strat soft-max [10].

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input ( $224 \times 224$ RGB image)					
conv3-64	conv3-64 <b>LRN</b>	conv3-64 <b>conv3-64</b>	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 <b>conv3-128</b>	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 <b>conv1-256</b>	conv3-256 conv3-256 <b>conv3-256</b>	conv3-256 conv3-256 conv3-256 <b>conv3-256</b>
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 <b>conv1-512</b>	conv3-512 conv3-512 <b>conv3-512</b>	conv3-512 conv3-512 conv3-512 <b>conv3-512</b>
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 <b>conv1-512</b>	conv3-512 conv3-512 <b>conv3-512</b>	conv3-512 conv3-512 conv3-512 <b>conv3-512</b>
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

**Figura 2.10:** Configurații ale rețelei VGG. Imagine preluată din [10].

### 2.2.3 InceptionV3

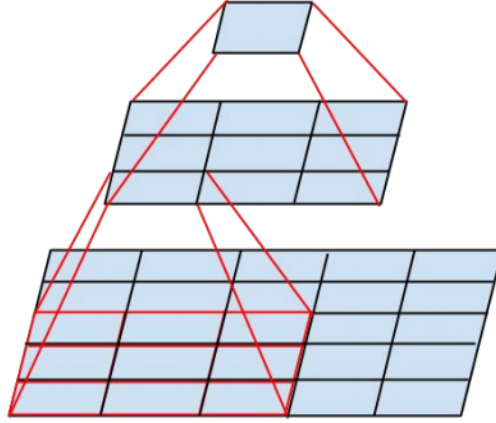
Prima arhitectura de Inception a apărut sub numele de GoogLeNet. O a doua versiune de Inception a fost definită prin introducerea de batch-uri normalizate. Iar mai apoi, versiunea a treia în care a fost adăugate idea de factorizare.

Factorizarea în filtre convoluționale mici presupune înlocuirea stratului cu filtru de dimensiune  $5 \times 5$  cu două straturi de dimensiune  $3 \times 3$  astfel reducându-se dimensiunea de la  $5 \times 5 = 25$  la  $3 \times 3 + 3 \times 3 = 18$ .

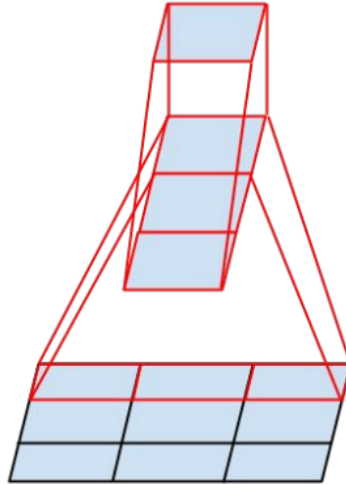
Factorizarea spațială în convoluții asimetrice presupune înlocuirea stratului cu filtru de dimensiune  $3 \times 3$  cu două straturi de dimensiune  $3 \times 1$  și  $1 \times 3$  astfel reducându-se dimensiunea de la  $3 \times 3 = 9$  la  $3 \times 1 + 1 \times 3 = 6$ .

Clasificatorul auxiliar este utilizat în InceptionV3 ca regulizator și este poziționat în partea superioară a utimelor  $17 \times 17$  straturi. Batch-urile normalizate sunt de asemenea folosite în clasificadorul auxiliar.

Reducerea eficientă a dimensiunii se face prin utilizarea a două blocuri paralele, fie



**Figura 2.11:** Factorizarea în filtre convoluționale mici. Filtrul de dimensiune  $5 \times 5$  înlocuit cu două de dimensiune  $3 \times 3$ . Imagine preluată din [14].



**Figura 2.12:** Factorizarea în filtre convoluționale mici. Filtrul de dimensiune  $3 \times 3$  înlocuit cu două de dimensiune  $3 \times 1$  și  $1 \times 3$ . Imagine preluată din [14].

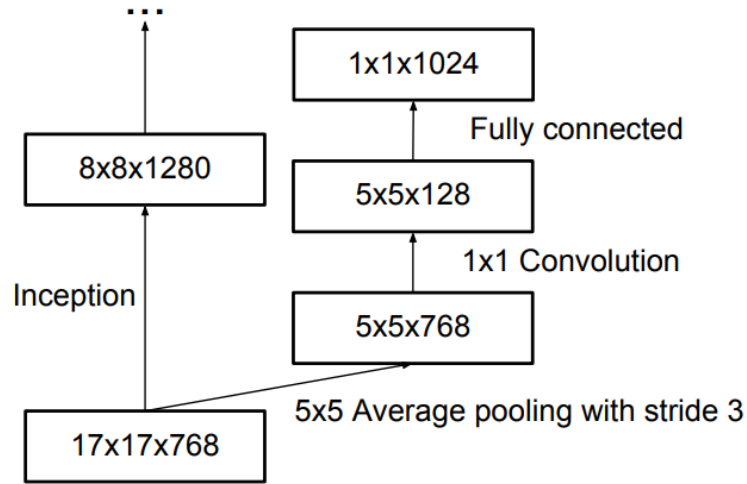
P și C. Primul dintre acestea, P, fiind un strat de activare de pooling (media sau maxim pooling). Ambele straturi au filtre cu pas 2 care sunt concatenate.

Arhitectura completă este prezentată în continuare.

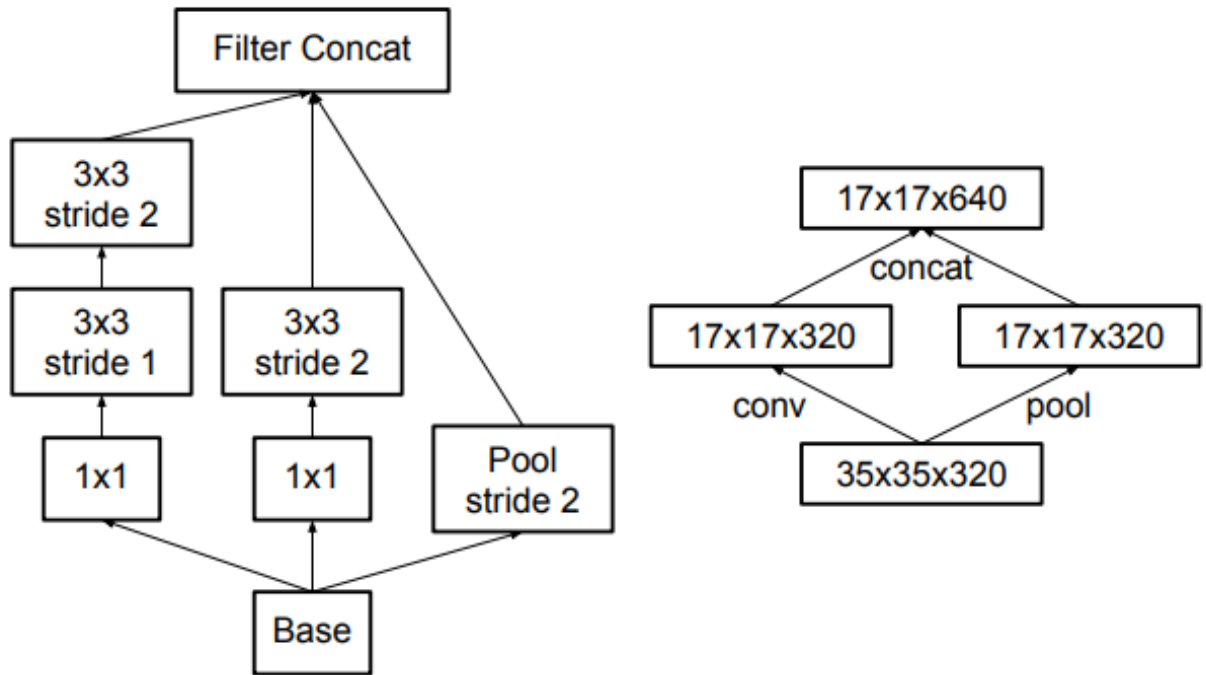
## 2.2.4 ResNet

Fie  $H(x)$  maparea de bază unde  $x$  reprezintă inputul. Funcția reziduală poate fi aproximată cu  $F(x) := H(x) - x$ , maparea de bază fiind  $F(x) + x$ .

Învățarea reziduală se aplică la câțiva grupuri de straturi. Putem defini un bloc de



**Figura 2.13:** Clasificatorul auxiliar. Imagine preluată din [14].



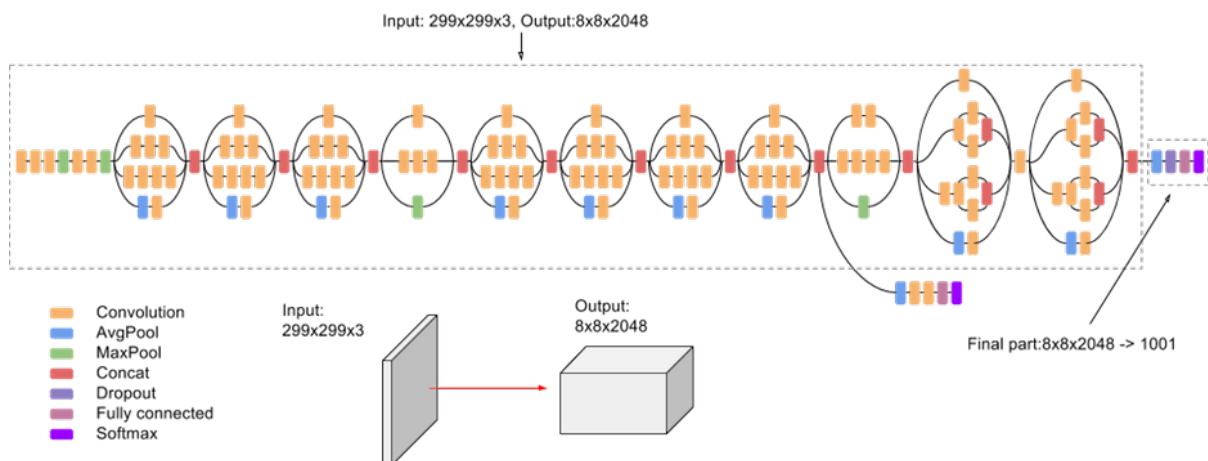
**Figura 2.14:** Modulul care reduce dimensiunea. Diagrama din dreapta reprezintă aceeași soluție însă din perspectiva dimensiunii rețelei. Imagine preluată din [14].

straturi ca fiind

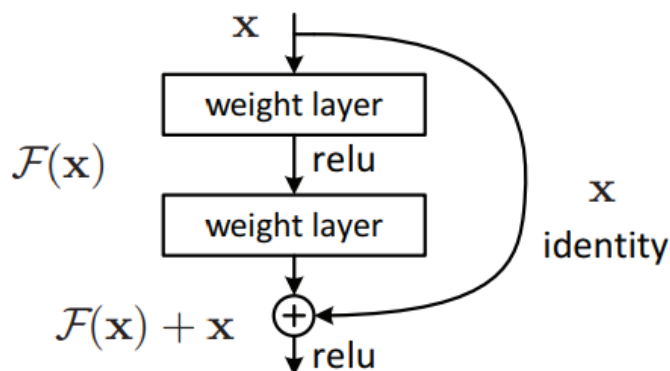
$$y = F(x, \{W_i\}) + x \quad (2.8)$$

unde  $x$  și  $y$  reprezintă inputul și outputul straturilor considerate. Funcția  $F(x, \{W_i\})$  reprezintă maparea reziduală ce trebuie învățată.

Dimensiunea lui  $x$  și  $F$  din ecuația de mai sus trebuie să fie egale. Redefinim ecuația



**Figura 2.15:** Arhitectura rețelei InceptionV3. Imagine preluată din [14].



**Figura 2.16:** Învățarea reziduală. Imagine preluată din [11].

după cum urmează

$$y = F(x, \{W_i\}) + W_s x \quad (2.9)$$

unde  $W_s$  este o proiecție liniară a scurtăturilor conexiunilor pentru ca dimensiunile să se potrivească.

ResNet pleacă de la o rețea simplă. Rețeaua simplă fiind inspirată de rețeaua VGG. Straturile convoluționale au în general filtre de dimensiune  $3 \times 3$  și se bazează de două reguli de design: - pentru outputuri cu același număr de caracteristici, straturile vor avea același număr de filtre; - dacă numărul de caracteristici este înjumătățit, numărul de filtre este dublat astfel încât să fie păstrată complexitatea de timp pe strat.

Poolingul se realizează după straturile convoluționale cu un pas de 2 pixeli. Rețeaua se termină cu un strat de pooling mediu și un strat fully-connected softmax cu 1000 de canale.



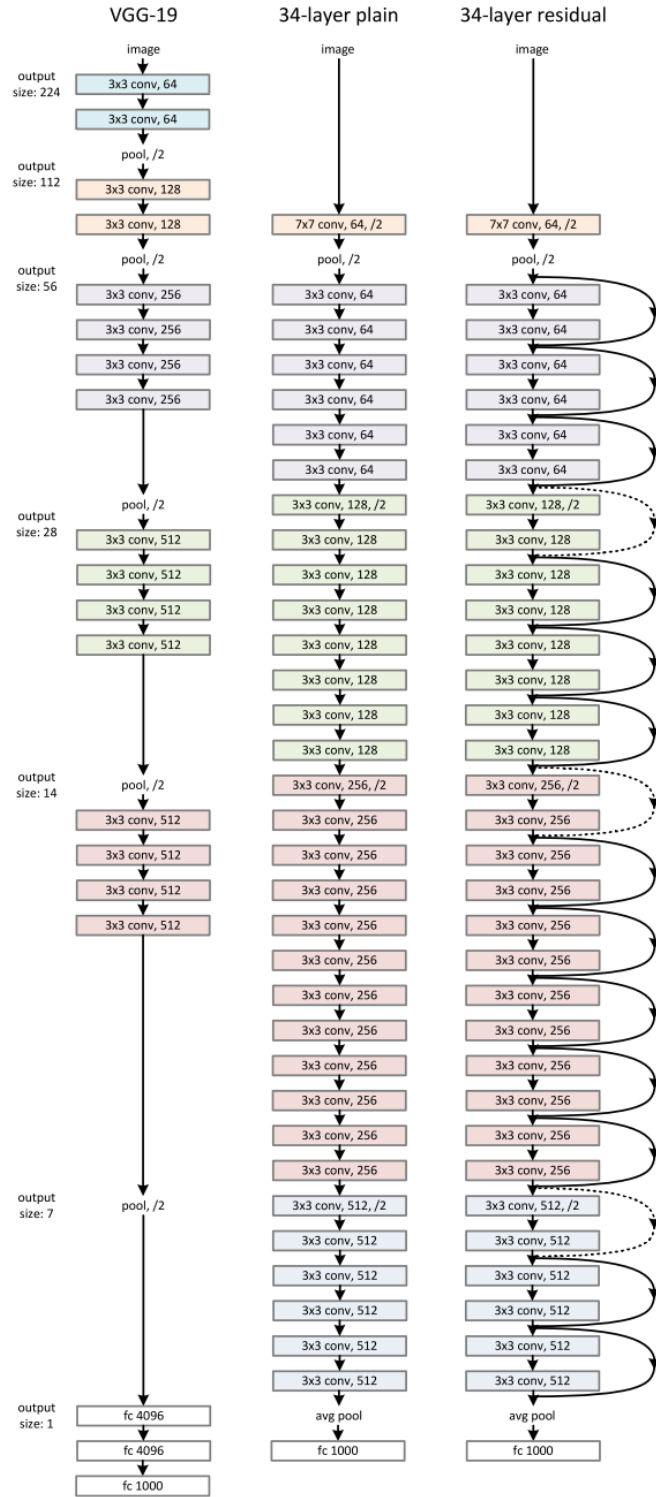
Bazată pe rețeaua descrisă mai sus, rețeaua reziduală presupune inserția unor scurtături. Scurtăturile identice (vezi formula 2.8) pot fi direct utilizate când inputul și outputul au aceeași dimensiune. Când dimensiunea crește considerăm două opțiuni: - scurtătura calculează în continuare maparea identității. Această opțiune nu introduce parametrii noi; - proiecția scurtăturii din formula 2.9 este utilizată pentru a potrivi dimensiunile. În ambele situații când se folosesc scurtăturile pentru a sări peste două straturi sunt calculate cu un pas de 2.

### 2.2.5 NASNet

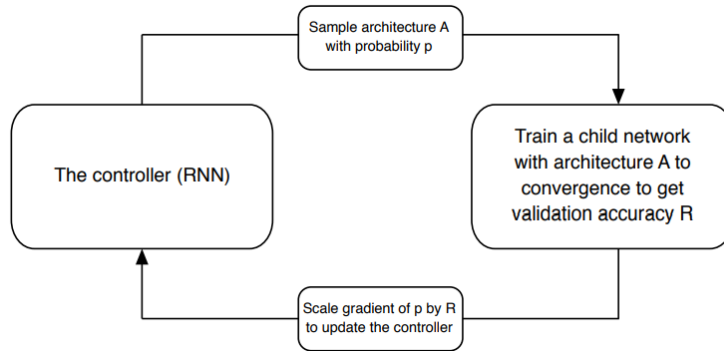
NASNet este o arhitectură de rețea bazată pe tehnica de căutare Neural Architecture Search (NAS). NAS presupune un controler cu o rețea neurală recurentă care conține mai multe rețele copii cu arhitecturi diferite. Rețelele copii sunt antrenate să convergă pentru a obține o anumită precizie pe un set de antrenare. Rezultatele sunt utilizate pentru a actualiza controlerul ceea ce înseamnă că acest controler va genera arhitecturi mai bune în timp.

Plusul principal pe care îl aduce rețeaua NASnet este reprezentat de proiectarea unui nou spațiu de căutare astfel încât cea mai bună arhitectură pe setul de date CIFAR-10 poate scala către rezoluții ale imaginilor cât mai mari într-un interval definit. Astfel, acest spațiu poartă numele de *NASNet search space*. În abordarea NASNet arhitecturile rețelelor convoluționale manual predeterminate, fiind compuse din celule convoluționale repetate de multe ori unde, fiecare celulă convoluțională are aceeași arhitectură dar ponderi diferite.

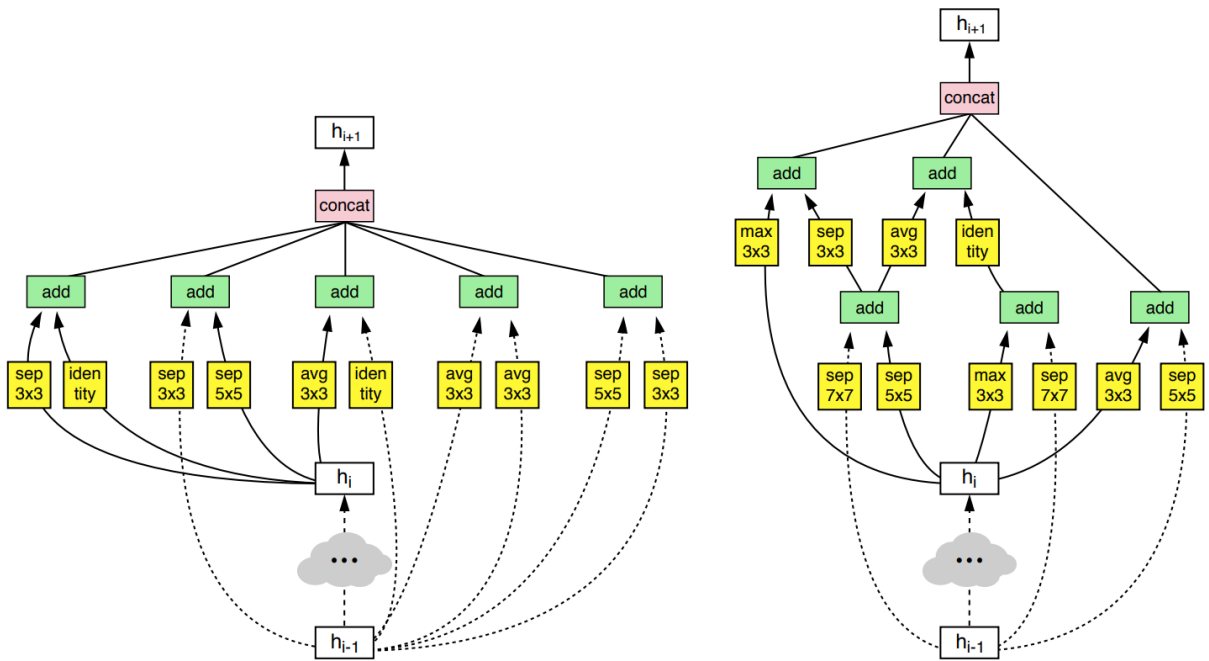
Pentru a construi mai ușor arhitecturi scalabile pentru imagini de orice dimensiune este nevoie de două tipuri de celule convoluționale pentru a îndeplini două funcții principale: - celule convoluționale care returnează o hartă de caracteristici cu aceeași dimensiune. Acest tip de celule se numesc *Celulă Normal*; - celule convoluționale care returnează o hartă de caracteristici cu înălțimea și lungimea hărții divizată cu un factor doi. Acest tip de celule se numesc *Celulă de reducere*.



**Figura 2.17:** Prima rețea (stânga) este o rețea VGG19. A doua rețea (centru) este o rețea simplă cu 34 de straturi. A treia rețea (dreapta) este o rețea reziduală cu 34 de straturi. Imagine preluată din [11].



**Figura 2.18:** Privire de ansamblu asupra unei Neural Architecture Search. Imagine preluată din [13].

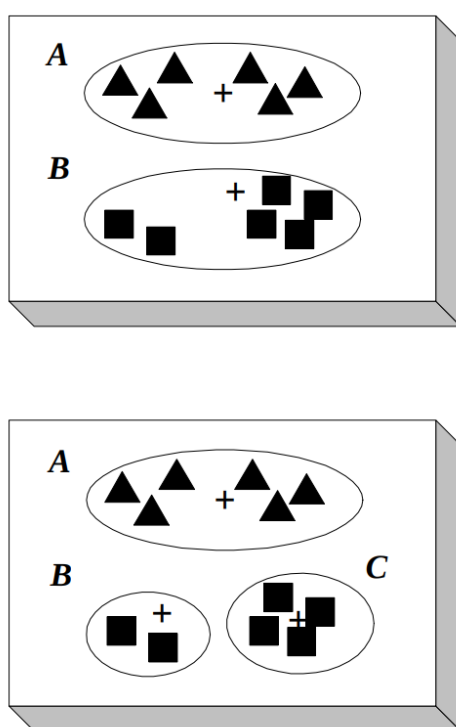


**Figura 2.19:** Celule normale (dreapta). Celule de reducere (stânga). Imagine preluată din [13].

## 2.3 Clustere

### 2.3.1 Noțiuni generale

Clusterizarea este un proces de grupare a unor articole în sensul în care articolele din același cluster sunt foarte similare între ele din punct de vedere al caracteristicilor. Această metodă este des utilizată în data mining, analiza datelor, machine learning, recunoașterea tiparelor sau regăsirea informației. Există mai multe tipuri de algoritmi de clusterizare, însă, ideea de bază este aceeași: clusterelor sunt grupuri cu distanțe foarte mici între membrii grupului (fie distanța euclidiană, de exemplu).



**Figura 2.20:** Exemplu de două clustere (sus). Exemplu de trei clustere (jos). Imagine preluată din [15].

Procesul de clustering poate fi împărțit în etape după cum urmează [15]:

1. Colectarea datelor: alegerea articolelor pentru care se va aplica clusterizarea;
2. Screening-ul inițial: presupune extragerea caracteristicilor relevante pentru fiecare articol din dataset;
3. Reprezentarea: presupune pregătirea datelor pentru a putea fi folosite de către algoritmul de clusterizare, tot aici alegându-se și măsura de similaritate;

4. Tendința de grupare: se verifică dacă datele au o tendință naturală de grupare; poate fi sărită pentru baze de date mari;
5. Strategia de clusterizare: se alege algoritmul de clusterizare și parametrii inițiali;
6. Validarea: se evaluează manual/vizual sau prin alte metode definite rezultatele obținute în urma clusterizării;
7. Interpretarea: în această se compară rezultatele pe mai multe clustere, combinații de clustere și se trag concluziile.

### 2.3.2 K-nearest neighbors

KNN reprezintă un model de clasificare simplu și eficient în multe cazuri. Pentru ca un articol  $t$  să fie clasificat sunt căutați cei mai apropiați  $k$  vecini formând regiunea lui  $t$ . Cei mai apropiați veci sunt căutați cu o măsură de similaritate, de obicei distanța euclidiană sau similaritatea cosinus. Votul majoritar din acea regiune este folosit pentru a decide clasificarea lui  $t$ .  $k$ -ul este valoarea de care depinde destul de mult rata de succes a clasificării, cea mai simplă metodă de a alege un  $k$  optim fiind reprezentată de rularea algoritmului pentru mai multe valori ale lui și observarea evoluției rezultatelor.

Fie  $D$  o colecție de  $n$  clase cunoscute  $\{d_1, d_2, \dots, d_n\}$ .  $Sim(d_i)$  - similaritatea celui mai îndepărtat punct din regiunea locală,  $N(d_i)$  - numărul de puncte din interiorul unei regiuni locale. Algoritmul de construcție al modelului este definit după cum urmează [15]:

1. selectăm o măsură de similaritate și creem o matrice de similaritate peste baza de date de antrenare;
2. setăm toate datele cu eticheta neclasificate;
3. pentru fiecare intrare cu eticheta de neclasificat căutăm cea mai mare regiune locală care acoperă cel mai mare număr de vecini cu aceeași categorie.
4. căutam intrarea  $d_i$  cu cea mai mare regiune  $N_i$  printre toate regiunile locale și creem o reprezentare  $\langle Cls(d_i), Sim(d_i), Num(d_i), Rep(d_i) \rangle$  în modelul  $M$  pentru a reprezenta toate intrările acoperite de regiunea  $N_i$  și setăm etichete pentru toate aceste intrări;

5. repetăm pași 3 și 4 până când toate intrările din baza de date de antrenare au fost clasificate;
6. modelul  $M$  este format din toate reprezentările setate în procesul de învățare descris mai sus.

Algoritmul de clasificare este definit după cum urmează:

1. pentru ca o nouă intrare  $d_t$  să fie clasificată, calculăm similaritatea ei cu toate celelalte reprezentări din model;
2. dacă  $d_t$  este acoperit doar de o reprezentare  $\langle Cls(d_j), Sim(d_j), Num(d_j), Rep(d_j) \rangle$  în sensul că distanța de la  $d_t$  la  $d_j$  este mai mică decât  $Sim(d_j)$ ,  $d_t$  este astfel clasificat ca făcând parte din clasa lui  $d_j$ ;
3. dacă  $d_t$  este acoperit de două sau mai multe clase, clasificăm  $d_t$  ca făcând parte din reprezentarea cu cea mai mare valoare a  $Num(d_j)$ , adică regiunea care acoperă cel mai mare număr de intrări din baza de date de antrenare;
4. dacă nu există nicio reprezentare în modelul  $M$  care să acopere  $d_t$ , clasificăm  $d_t$  cu o clasă nouă.

Un exemplu vizual de execuție a algoritmului de kNN, parcurs pas cu pas, este prezentat în figura 2.21:

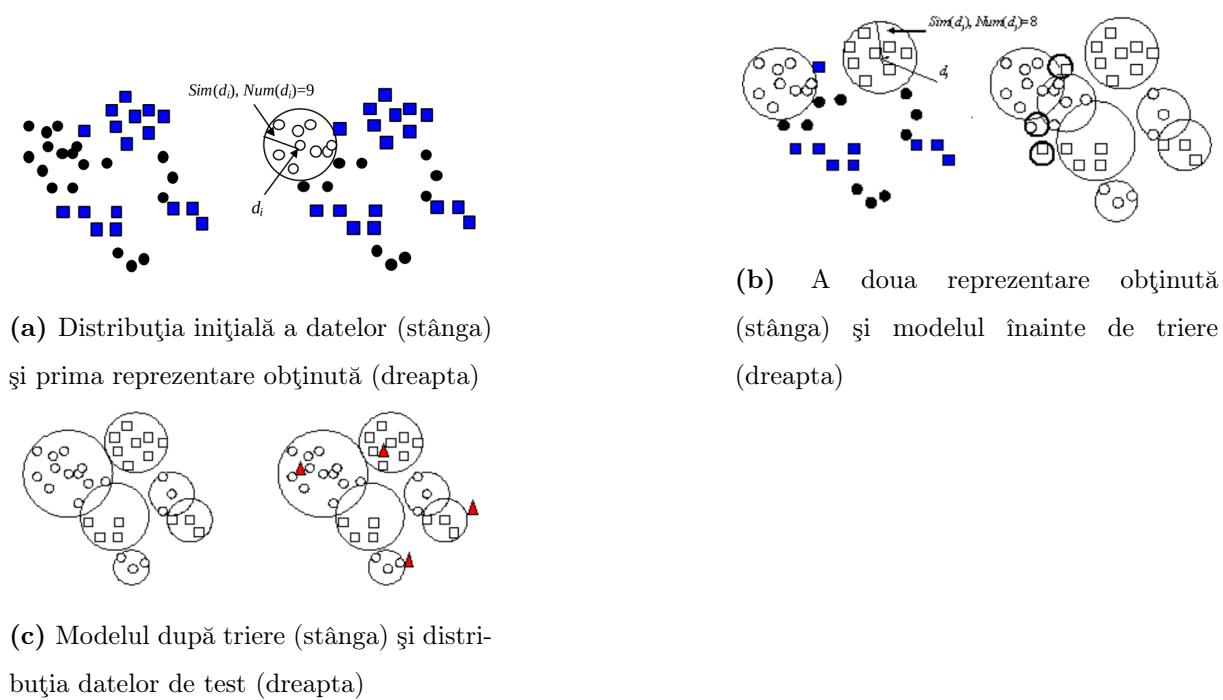
Exemplu conține 36 de intrări din 2 clase marcate prin pătrat și cerc. Datele de test sunt reprezentate prin tringhiuri.

Un al doilea exemplu de clasificare este prezentat în figura 2.22:

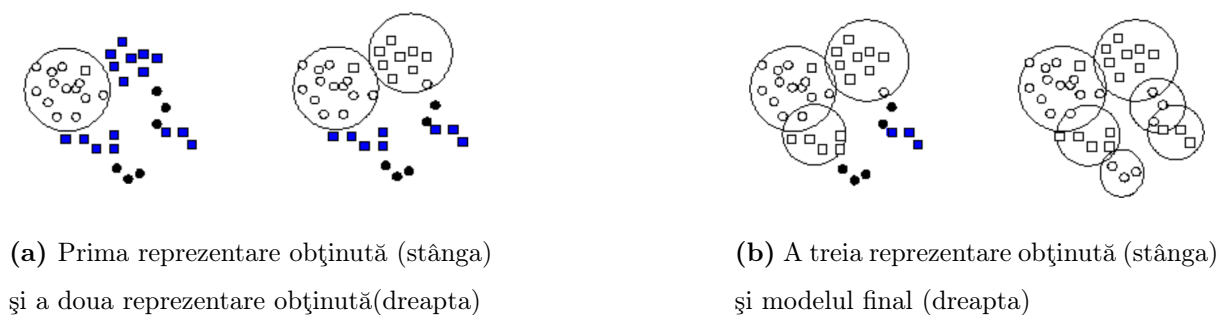
### 2.3.3 Metrice de evaluare a clusterelor

#### Coeficientul silhouette

Coeficient (formula 2.10) este folosit pentru a evalua clusterelor în învățarea nesupervizată. Este calculat utilizând distanța medie intra-cluster ( $a$ ) și distanța medie către cel mai apropiat cluster ( $b$ ) pentru fiecare intrare, adică distanța dintre o intrare și cel mai apropiat cluster din care nu face parte. Numărul de etichete trebuie să respecte constrangerea  $2 \leq nr_{etichete} \leq nr_{etichete} - 1$ . Valorile returnate de acest coeficient sunt cuprinse în intervalul  $[-1, 1]$ . Valorile apropiate de 0 indică clusterelor care se suprapun,



**Figura 2.21:** Exemplu 1 de clasificare cu kNN. Imagine preluată din [15].



**Figura 2.22:** Exemplu 2 de clasificare cu kNN. Imagine preluată din [15].

valorile negative în general indică că există intrări asignate în clusterul greșit, iar valorile apropiate de 1 indică o separație bună între clustere [16].

$$\frac{b - a}{\max(a, b)} \tag{2.10}$$



# Bibliografie

- [1] Data never sleeps 6.0  
<https://www.domo.com/learn/data-never-sleeps-6>
- [2] Netflix International: What movies and TV shows can I watch, and where can I watch them?  
<https://www.finder.com/global-netflix-library-totals>
- [3] How Many Products Does Amazon Sell? – April 2019  
<https://www.scrapehero.com/number-of-products-on-amazon-april-2019/>
- [4] Erion Çano, Maurizio Morisio. *Hybrid Recommender Systems: A Systematic Literature Review*. Intelligent Data Analysis, vol. 21, no. 6, pp. 1487-1524, 2017
- [5] An Overview of Recommendation Systems  
<http://datameetsmedia.com/an-overview-of-recommendation-systems/>
- [6] LightFM 1.15 - documentation  
<http://lyst.github.io/lightfm/docs/lightfm.html>
- [7] CS231n: Convolutional Neural Networks for Visual Recognition  
<http://cs231n.stanford.edu/2018/syllabus.html>
- [8] Jason Weston, Samy Bengio, Nicolas Usunier. *Wsabie: Scaling up to large vocabulary image annotation*. IJCAI. Vol. 11. 2011.
- [9] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner and Lars Schmidt-Thieme. *BPR: Bayesian personalized ranking from implicit feedback*. Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence. AUAI Press, 2009.
- [10] Karen Simonyan, Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. ICLR, 2015.

- [11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. *Deep Residual Learning for Image Recognition*. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), p770-778, 2016.
- [12] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna. *Rethinking the Inception Architecture for Computer Vision*. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), p2818-2826, 2016.
- [13] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, Quoc V. Le. *Learning Transferable Architectures for Scalable Image Recognition*. IEEE/CVF Conference on Computer Vision and Pattern Recognition, p8697-8710, 2018.
- [14] Advanced Guide to Inception v3 on Cloud TPU  
<https://cloud.google.com/tpu/docs/inception-v3-advanced>
- [15] Data Clustering Techniques  
<http://www.cs.toronto.edu/~periklis/pubs/depth.pdf>
- [16] Gongde Guo, Hui Wang, David Bell, Yaxin Bi, Kieran Greer. *KNN Model-Based Approach in Classification*. Lecture Notes in Computer Science, vol 2888. Springer, Berlin, Heidelberg, 2003.
- [17] Sklearn metrics - silhouette score  
[https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette\\_score.html](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html)