UNIVERSITATEA POLITEHNICA DIN BUCUREȘTI

Facultatea de Electronică, Telecomunicații și Tehnologia Informației

Proiect 3

Detecție de emoții din fețe

|  |  |
| --- | --- |
| Profesor coordonator:  Anamaria Dumitrescu | Studenți:  Diaconeasa Ioana-Ștefania 441G  Dumitru Ștefan-Rareș 442G |

CUPRINS

INTRODUCERE ……………………………………………………………………………………….. 3

1. EMOȚIILE ............................................................................................................................................3

2. DECIZIE ȘI CLASIFICARE ................................................................................................................5

2.1. CONTEXT..............................................................................................................................5

2.2. MACHINE LEARNING ȘI TIPURI DE ÎNVĂȚARE .........................................................5

2.2.1. ÎNVĂȚAREA SUPERVIZATĂ...............................................................................6

2.2.2. ÎNVĂȚAREA NESUPERVIZATĂ..........................................................................6

2.2.3. ÎNVĂȚAREA SEMISUPERVIZATĂ......................................................................6

2.3. CLASIFICARE. TIPURI DE CLASIFICARE........................................................................6

2.4. EVALUAREA UNUI MODEL DE ÎNVĂȚARE AUTOMATĂ.............................................7

3. DETECȚIE DE EMOȚII UTILIZÂND REȚELE NEURALE CONVOLUȚIONALE........................8

3.1. DEEP LEARNING. REȚELE NEURALE CONVOLUȚIONALE (CNN)............................8

3.2. SUPRAÎNVĂȚARE ȘI SUBÎNVĂȚARE..............................................................................9

3.3. IMPLEMENTAREA PRACTICĂ A DETECȚIEI DE EMOȚII............................................9

REFERINȚE BIBLIOGRAFICE.............................................................................................................13

ANEXE....................................................................................................................................................14

INTRODUCERE

Desideratul acestui proiect constă în elaborarea unui sistem software dedicat identificării și clasificării emoțiilor umane exprimate în imagini faciale, concentrându-se în special pe distingerea între stările de furie (anger) și cele non-furie (not anger).

Pentru a atinge obiectivele proiectului, s-au avut în vedere următoarele etape:

1. Crearea unei interfețe de afișare și evaluare a rezultatelor algoritmului automat de analiză a imaginilor faciale pentru identificarea și interpretarea expresiilor de furie și non-furie.

2. Implementarea algoritmilor de extragere a trăsăturilor semnificative din imagini pentru o descriere automată a expresiilor faciale și pentru clasificarea corectă între starea de furie și cele non-furie.

3. Testarea sistemului software pe un set standard de imagini ce conține expresii faciale de furie și non-furie.

4. Realizarea de simulări și comparații pentru evaluarea performanței și eficienței sistemului în identificarea și clasificarea corespunzătoare a expresiilor faciale de furie și non-furie.

Dezvoltarea software s-a desfășurat utilizând limbajul de programare Python, având ca mediu de lucru Pycharm, iar datele de imagini utilizate au provenit din surse publice conținând imagini faciale asociate cu diverse emoții umane (FER-2013). Alegerea setului de date FER-2013 este justificată de provocările întâlnite în vederea obținerii acurateții crescute la clasificare (în prezent, cele mai eficiente modele ating o acuratețe de 75%, în timp ce creierul uman are o acuratețe de 65 ± 5%).

Primul capitol al lucrării oferă informații teoretice despre expresiile faciale, relevanța lor în procesul comunicării și instrumentele utilizate pentru analiza acestora.

În cadrul celui de-al doilea capitol, vor fi explorate conceptele de decizie și clasificare, cu accent pe aplicabilitatea lor în domeniul machine learning în identificarea expresiilor faciale de furie și non-furie.

În ultimul capitol, se va introduce teoria deep learning și a rețelelor neuronale convoluționale, urmată de implementarea unei astfel de rețele pentru identificarea automată a expresiilor faciale de furie și non-furie. Vor fi prezentate rezultatele obținute în urma testării rețelei pe setul de imagini faciale.

1. EMOȚIILE

În prezent, ne confruntăm cu un viitor promițător pentru utilizarea mașinilor în diferite industrii, datorită dezvoltării accelerate și continue a inteligenței artificiale. Acestea au devenit o parte importantă a vieții de zi cu zi a oamenilor, reprezentând un mijloc substanțial de facilitare a proceselor complexe și automate. Cu cât este mai lungă expunerea între oameni și mașini, cu atât interacțiunile devin mai naturale și mai bine ordonate. Noi, oamenii, din natura noastră, ne folosim și ne bazăm pe simțuri pentru a învăța despre mediul înconjurător și pentru a anticipa diverse evenimente și situații prin evoluție. În mod similar, scopul mașinilor este de a reproduce acest mecanism uman de percepție și învățare pentru a interacționa cu circumstanțele și situațiile lor.

Emoțiile pe care le exprimăm sunt adesea evidente, chiar și atunci când nu suntem conștienți de ele, reprezentând una dintre cele mai esențiale forme de comunicare între oameni. Investiția în înțelegerea și interpretarea acestora direct din expresiile faciale este esențială pentru o mai bună înțelegere a persoanelor din jur și pentru îmbunătățirea comunicării în atingerea unor obiective comune sau evitarea situațiilor tensionate. Recunoașterea expresiilor faciale are multiple aplicații. De exemplu, este crucială în dezvoltarea cognitivă sănătoasă a sugarilor, ajutând la detectarea și tratamentul precoce al unor tulburări precum tulburarea de spectru autist (TSA) și tulburarea de deficit de atenție și hiperactivitate (TDAH). De asemenea, în scopuri de sănătate, poate fi utilizată în centrele de îngrijire pentru persoanele în vârstă, ajutându-le să depășească anxietățile sau fobiile și să facă față dificultăților vieții lor. Expresiile faciale pot fi, de asemenea, indicatori ai sănătății fizice precare, mai ales în cazul persoanelor cu o istorie familială de boli cronice precum cancerul, bolile de inimă, diabetul sau osteoporoza.

Pe lângă beneficiile terapeutice, există și un potențial semnificativ pentru educație și training. Înțelegerea reacțiilor emoționale ale elevilor poate ajuta educațtorii să adapteze metodele de predare în timp real, evaluând nivelul de angajare al studenților în timpul cursurilor online sau a conferințelor.

Aceste aplicații subliniază importanța recunoașterii expresiilor faciale în diverse domenii ale vieții noastre, de la sănătate și securitate, până la educație și interacțiunile zilnice.Expresiile faciale în comunicarea umană conțin informații nonverbale cruciale care pot oferi înțelesuri și semnificații suplimentare conversațiilor verbale. Conform anumitor cercetări, comunicarea nonverbală reprezintă între 60-80% din totalul comunicării. Comunicarea nonverbală cuprinde expresiile faciale, contactul vizual, tonurile vocii, gesturile cu mâinile și distanța fizică. Analiza expresiilor faciale, în special, a constituit un domeniu major de cercetare. În domeniul interacțiunii om-computer (HCI), recunoașterea emoțiilor faciale (FER - facial emotional recognition) a fost utilizată în domenii precum pilotul automat, educație, îngrijire medicală, tratament psihologic, supraveghere și analiză psihologică în viziunea computerizată.

Emoțiile sunt clasificate ca modele categorice sau dimensionale (valență și excitație) în psihologie și viziune computerizată. Ekman et al. au caracterizat emoțiile umane primare în modelul categoric ca fiind fericire, furie, dezgust, frică, tristețe și surpriză. Emoția în modelul dimensional este evaluată folosind scale numerice continue pentru a determina valența și excitația. FER este o sarcină esențială în viziunea computerizată cu mai multe aplicații practice, iar numărul studiilor despre FER a crescut în ultimii ani ca rezultat al îmbunătățirilor în rețelele neurale profunde. În special, rețelele neurale convoluționale (CNN-uri) au obținut rezultate remarcabile în ceea ce privește extragerea caracteristicilor.

În psihologia socială, o micro-expresie este o expresie facială ușor de observat și de distins ca strategie de comunicare. Expresiile faciale indică obiectivele și ambițiile noastre, oferă informații despre emoțiile noastre și joacă un rol important în interacțiunea umană. Recunoașterea și înțelegerea expresiilor faciale în mod automat promovează comunicarea intenționată. Detectarea feței, extragerea caracteristicilor și clasificarea expresiilor faciale umane sunt cele trei etape în categorisirea expresiilor faciale umane.



Figura 1.1

În Figura 1.1 putem observa principalele trăsături distinctive ale expresiei de furie, care includ: sprâncenele care se unesc și se îndreaptă în jos, privirea pătrunzătoare a ochilor și îngustarea buzelor. Aceste caracteristici ale expresiei sunt distinctive pentru emoția pe care trebuie să o analizăm și ne ajută să facem o paralelă între o față "furioasă" și una "non-furioasă".

Cu toate acestea, există diverse dificultăți întâmpinate în procesul de mapare. Multe persoane pot înșela sistemele prin falsificarea emoțiilor sau prin nedemonstrarea acestora deloc. Cel mai bun și cel mai cunoscut exemplu este cel al actorilor de teatru și film care pot simula aceste emoții la fel de ușor cum doresc, fără nevoia unor stimuli specifici în spatele lor. În această categorie, putem menționa și cazul persoanelor care suferă de paralizie facială totală sau parțială. Există însă o diferență: un experiment a arătat că în cazul unei persoane cu jumătate din față paralizată, atunci când i s-a spus să zâmbească la comandă s-a observat că doar jumătate din gura sa se ridica, în timp ce când aceeași persoană a fost supusă spectacolelor de comedie (prin urmare, acțiunea râsului a fost naturală), s-a observat că persoana respectivă a ridicat părți din ambele jumătăți ale gurii. Prin urmare, în ceea ce privește calculatoarele, sunt investigate și dezvoltate multe metode pentru a ne ajuta să ne îmbunătățim capacitatea de a exprima și recunoaște emoțiile și de a face diferența între cele reale și cele false.

2. DECIZIE ȘI CLASIFICARE

2.1. CONTEXT

În rutina noastră zilnică, frecvent ne găsim în situații în care trebuie să facem alegeri, fără să cunoaștem cu certitudine consecințele acestora. În aceste momente, luăm o decizie. Prin prisma unei definiții generale decizia poate fi descrisă ca o "hotărâre luată după analiza unei probleme sau a unei situații, adoptând una din mai multele soluții posibile". Această realitate este întâlnită frecvent și în domeniul tehnic, mai ales în sfera inteligenței artificiale, unde decizia constituie fundamentul majorității acțiunilor întreprinse de modelele de inteligență artificială.

2.2. MACHINE LEARNING ȘI TIPURI DE ÎNVĂȚARE

În mediul actual, abundăm în informații, iar necesitatea uneltelor care să faciliteze și să eficientizeze prelucrarea acestora este tot mai evidentă. În acest context, apare în prim-plan domeniul învățării automate (cunoscut și sub numele de machine learning). Machine learning, sau învățarea automată, reprezintă o ramură a inteligenței artificiale și a informaticii, focalizată pe utilizarea datelor și a algoritmilor pentru a emula modul în care oamenii învață din experiențele anterioare, gradual îmbunătățindu-și precizia.

În cadrul modelelor de machine learning, procesul de învățare poate fi împărțit în trei categorii: supervizată, nesupervizată și semisupervizată.

2.2.1 ÎNVĂȚAREA SUPERVIZATĂ

Învățarea supervizată se definește prin faptul că datele de intrare sunt deja etichetate, modelului de machine learning revenindu-i sarcina de a învăța caracteristicile datelor pentru ca ulterior să poată eticheta noi date de intrare în categoriile corespunzătoare. În cadrul acestui tip de învățare se pot distinge două tipuri de algoritmi: regresie și clasificare. Diferența dintre regresie și clasificare este următoarea: o regresie produce un rezultat concret (de exemplu, estimarea valorii monedei Bitcoin peste doi ani), în timp ce o clasificare reprezintă o predicție a apartenenței unor date la o clasă, adică o etichetă.

2.2.2. ÎNVĂȚAREA NESUPERVIZATĂ

Învățarea nesupervizată este un tip de învățare care se definește prin faptul că datele de intrare sunt neetichetate, modelului de machine learnng revenindu-i sarcina de a identifica atât caracteristicile datelor de intrare cât și clasificarea acestora în baza unor trăsături comune, clasificare care, în absența unor etichete, se denumește clustering.

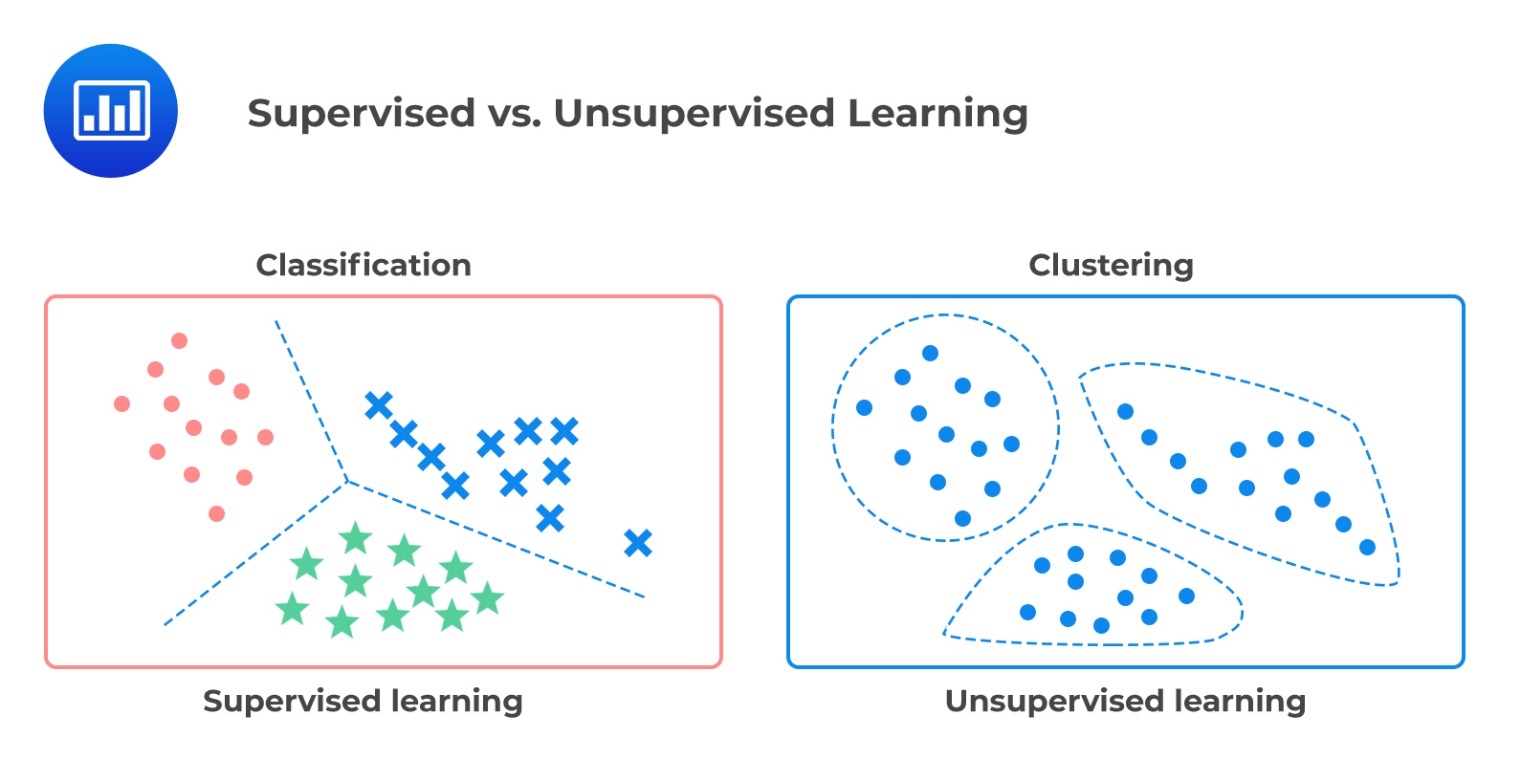


Fig 2.1 – Clasificare vs. Clustering

2.2.3. ÎNVĂȚAREA SEMISUPERVIZATĂ

Învățarea semisupervizată este o combinație între modelele supervizate și cele nesupervizate. În contextul acestui tip de învățare, modelul primește la intrare atât date etichetate cât și neetichetate.

2.3. CLASIFICARE. TIPURI DE CLASIFICARE

După cum am menționat anterior, clasificarea este un caz particluare al învățării supervizate, întrucât presupune încadrarea datelor într-o anumită categorie deja existentă, detaliu cu care vom lucra pentru restul capitolelor. În acest sens, clasificarea se poate caracteriza după mai multe aspecte:

1. după numărul de clase:

- clasificare binară: presupune existența a doar două clase (0 și 1)

- clasificare multiclasă: este o generalizare a clasificării binare, presupunând existența a mai mult de două etichete (exemplu, tipuri de emoții – furie, fericire, tristețe, frică)

2. după numărul de etichete atribuite la clasificare:

- clasificare cu etichetă unică: se atribuie o singură etichetă fiecărei secvențe de date

- clasificare cu etichete multiple: datele se încadrează în mai multe categorii (de exemplu, un semnal audio reprezentând o secvență dintr-o melodie, se poate încadra în mai multe genuri muzicale)

3. după numărul de date de intrare pentru antrenare:

- clasificare echilibrată: se primește același număr de date de antrenare pentru fiecare clasă

- clasificare neechilibrată: datele de intrare pentru antrenare sunt împărțite pe clase inegal

În cele ce urmează, ne vom axa pe definirea unui algoritm pentru rezolvarea tipului de problemă propus anterior, și anume, o clasificare binară neechilibrată.

2.4. EVALUAREA UNUI MODEL DE ÎNVĂȚARE AUTOMATĂ

Evaluarea unui model în urma unei clasificări se poate realiza urmărind matricea de confuzie pentru setul de testare,reprezentand un instrument analitic ce compară valorile reale cu valorile prezise de către model. Matricea de confuzie este un vector bidimensional N x N, unde N este numărul de clase/etichete. În cele ce urmează, vom studia matricea de confuzie pentru cazul binar (discuția poate fi extinsă la o matrice de confuzie de dimensiuni mai mari, pe aceleași principii, dar avem nevoie de o problemă ce presupune clasificare multiclasă, lucru care nu face subiectul discuției).

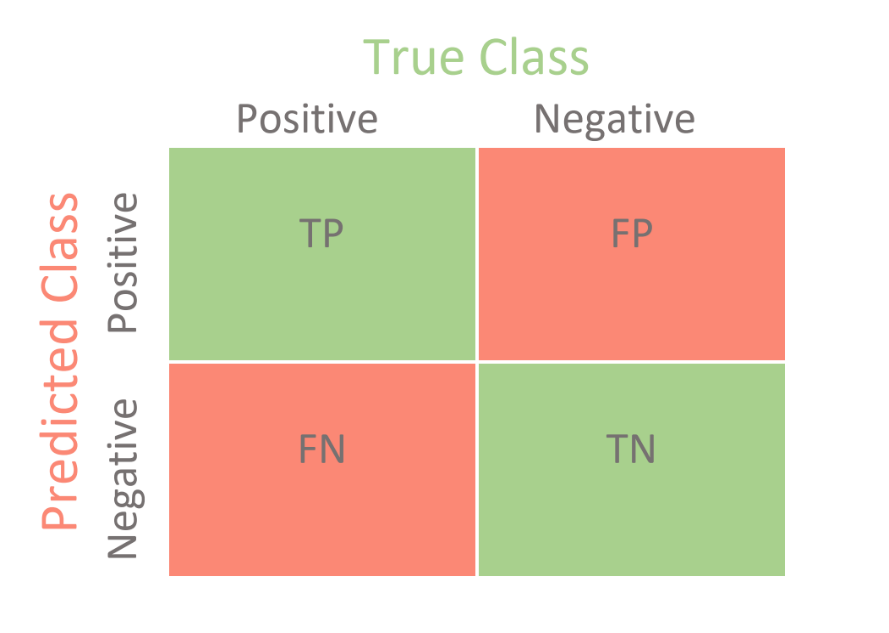


Figura 2.2 – Matrice de confuzie, clasificare binară

Din figura 2.2 putem observa cele două clase, pe care le vom asocia cu 0 și 1 (anger și not\_anger). Pentru fiecare celulă, putem distinge valorile prezise corect (T = true), și valorile prezise greșit (F = false).

Rata subiecților adevărat pozitivi (TP) – se referă la imaginile în care se regăsește o expresie reală de furie, iar predicția este tot pentru o expresie de furie.

Rata subiecților fals pozitivi (FP) – se referă la imagini în care se regăsește o expresie reală de furie, dar modelul a prezis că nu este o expresie de furie.

Rata subiecților adevărat negativi (TN) – se referă la imagini care nu conțin expresii de furie, iar predicția este tot pentru o expresie de non-furie

Rata subiecților fals negativi (FN) – se referă la imagini care nu conțin expresii de furie, iar predicția este pentru o expresie de furie.

Acuratețea, după care calculăm performanța modelului se definește folosind următoarea formulă:

3. DETECȚIE DE EMOȚII UTILIZÂND REȚELE NEURALE CONVOLUȚIONALE

3.1. DEEP LEARNING. REȚELE NEURALE CONVOLUȚIONALE (CNN)

Un model bazat pe deep learning prezintă caracteristici similare cu cele ale unui model bazat pe algoritmi tradiționali de machine learning, dar aduce și trăsături specifice. O caracteristică distinctivă a modelului bazat pe deep learning este capacitatea sa de a extrage automat trăsături semnificative din date, cum ar fi din imagini, fără necesitatea implementării manuale a algoritmilor pentru acest proces.

De asemenea, un alt aspect specific al modelelor bazate pe deep learning este abilitatea lor de a rezolva complet o problemă, cum ar fi clasificarea, în timp ce algoritmii clasici de machine learning necesită de obicei divizarea problemei în mai multe etape pentru a fi rezolvată, iar rezultatele trebuie integrate ulterior pentru a obține soluția finală.

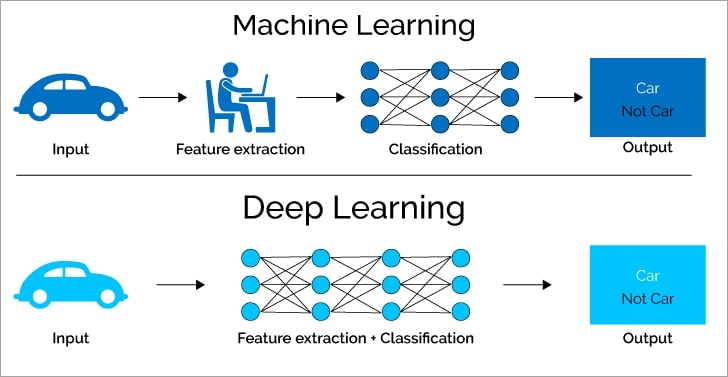


Figura 3.1 – Machine learning vs. Deep learning

Unul dintre cele mai cunoscute modele DL este cel reprezentat de rețele neurale convoluționale (în limba engleză – CNN – convolutional neural networks), care sunt utilizare în special pentru analiza imaginilor și clasficarea acestora. O rețea convoluțională generică are în componența sa unul sau mai multe straturi convoluționale, însoțite, de cele mai multe ori, de straturi de redimensionare (pooling), și un strat de vectorizare (flatten) la ieșirea din rețea (Figura 4.2). Cele mai relevante sunt în general, straturile convoluționale, întrucât ele efectuează operații matriceale între imagine și un nucleu pe post de filtru, rezultând într-o matrice ce conține elementele relevante ale imaginii de input. Ulterior, se aceste noi matrici (care în esență, sunt tot imagini) vor fi furnizate altor straturi, pentru ca algoritmul să poată învăța noi trăsături. Stratul de pooling este adesea utilizat în strânsă legătură cu stratul convoluțional, și are rolul de a reduce dimensiunile imaginii de ieșire, ceea ce implică reducerea numărului de parametri care trebuie antrenați (și mai implicit de atât, eficiență computațională). Totuși, stratul de pooling are și dezavantaje, întrucât se poate pierde o cantitate de informație relevantă pentru clasificare (de aceea, proiectarea unei astfel de rețele, este și un proces de trial and error).

A diagram of a diagram

Description automatically generated

Figura 3.2 – Rețea convoluțională generică

Printre ultimele componente ale arhitecturii le constituie straturile complet conectate, care realizează efectiv clasificarea. Ultimul strat al sub-rețelei de clasificare trebuie să conțină întocmai atâți neuroni câte etichete există (în cazul nostru, doi). Acesta reprezintă o distribuție probabilistică drept rezultat al clasificării, adică o pondere (o probabilitate) pentru fiecare clasă.

3.2. SUPRAÎNVĂȚARE ȘI SUBÎNVĂȚARE

În dorința de a obține rezultate precise, se întâmplă adesea să se configureze parametrii rețelelor neuronale la valori foarte mari, chiar dacă setul de date este limitat, conducând astfel la supraînvățare. Supraînvățarea apare atunci când rețeaua neuronală devine proficientă în recunoașterea detaliilor setului de date de antrenare, dar nu poate generaliza pentru datele noi. Acest fenomen poate fi identificat în timpul antrenării atunci când acuratețea rămâne constantă sau începe să scadă. Pe de altă parte, subînvățarea apare atunci când modelul nu reușește să învețe caracteristicile setului de date de antrenare sau să le aplice în mod corespunzător pe datele de testare.

3.3. IMPLEMENTAREA PRACTICĂ A DETECȚIEI DE EMOȚII

Pentru implementarea practică a fost folosit limbajul de programare Python, care conține o multitudine de librării ce facilitează proiectarea unor modele de învățare automată, dar și o serie de funcții ce permit prelucrarea datelor de intrare și vizualizarea rezultatelor. În realizarea proiectului s-a abordat o modalitate ce presupune construcția unui model propriu de CNN (Anexa 1). În vederea scrierii codului sursă și rulării cu succes a acestuia, au fost deduse următoarele etape:

1. Importarea bibliotecilor necesare, **tensorflow** (implicit **keras**)pentru proiectarea rețelei neurale: modelul **Sequential** pentru că există un singur input și un singur output, straturile **Conv2D, MaxPooling2D, Flatten, Dense, Dropout** pentru proiectarea straturilor, optimizatorul **Adam** și funcția de pierderi **BinnaryCrossentropy**. De asemenea, s-au folosit bibliotecile **matplotlib.pyplot** pentru afișarea imaginilor și graficelor, **numpy** pentru generare de secvențe de imagini (batch-uri de antrenare și testare), **sklearn.metrics** pentru interpretarea rezultatelor și **seaborn** pentru a produce heatmap-ul matricii de confuzie.

import tensorflow as tf

import matplotlib.pyplot as plt

from keras.models import Sequential, load\_model

from keras.layers import Conv2D,MaxPooling2D,Flatten,Dense,Dropout

from keras.optimizers import Adam

from keras.losses import BinaryCrossentropy

import numpy as np

from sklearn.metrics import confusion\_matrix, classification\_report

import seaborn as sns

2. Au fost definite directoarele pentru imaginile de train și test după preprocesarea imaginilor în subfoldere. Acest lucru s-a realizat micșorând baza de date originială la un model redus și balansat, atât pentru eficiență cât și pentru acuratețe mai bună la validare și testare. Cu ajutorul funcției **tf.keras.utils.image\_dataset\_from\_directory** s-a realizat etcihetarea automată a imaginilor din director, după care, utilizând funcția as\_numpy\_iterator s-au generat batch-urile de imagini ce vor fi furnizate rețelei (16 imagini per batch, pentru o eficiență mai bună din punct de vedere al timpului de execuție)

data\_train=tf.keras.utils.image\_dataset\_from\_directory(train\_dir,batch\_size=16,image\_size =(48,48))

data\_gen\_train = data\_train.as\_numpy\_iterator()

batch\_train = data\_gen\_train.next()

data\_test=tf.keras.utils.image\_dataset\_from\_directory(test\_dir,batch\_size=32,image\_size =(48,48))

data\_gen\_test = data\_test.as\_numpy\_iterator()

batch\_test = data\_gen\_test.next()

3. S-a realizat prelucrarea seturilor de imagini (prin normalizare cu valori între 0 și 1, pentru a faclita procesul de antrnare), după care s-a realizat împărțirea de 80% date de antrenare și 20% date de validare.

data\_train = data\_train.map(lambda x, y: (x/255, y))

data\_train.as\_numpy\_iterator().next()

data\_test = data\_test.map(lambda x, y: (x/255, y))

data\_test.as\_numpy\_iterator().next()

train\_dataset\_size = int(len(data\_train) \* 0.8)

validation\_dataset\_size = int(len(data\_train) \* 0.2)

train\_dataset = data\_train.take(train\_dataset\_size)

validation\_dataset= data\_train.skip(train\_dataset\_size).take(validation\_dataset\_size)

4. S-a realizat proiectarea modelului CNN cu trei straturi de convoluție + pooling, la care s-a adăugat câte un strat de dropout, menit să dezactiveze secvențial neuroni, pentru a evita procesul de supraînvățare. S-a realizat vectorizarea prin stratul de flatten, după care s-au folosit straturile de dense pentru rețeaua complet conectată. S-au ales funcții de activare ReLU pentru că nu activează toți neuronii în același timp, este mai generică, în timp ce pentru ultimul strat s-a folosit funcția clasică sigmoid, întrucât este cea mai potrivită pentru o problemă de clasificare binară.

CNN\_model = Sequential()

CNN\_model.add(Conv2D(16,(3,3),1,activation='relu', input\_shape=(48,48,3)))

CNN\_model.add(MaxPooling2D(pool\_size=(2,2)))

CNN\_model.add(Dropout(0.1))

CNN\_model.add(Conv2D(32, (3,3),1,activation='relu'))

CNN\_model.add(MaxPooling2D(pool\_size=(2,2)))

CNN\_model.add(Dropout(0.2))

CNN\_model.add(Conv2D(16, (3,3),1,activation='relu'))

CNN\_model.add(MaxPooling2D(pool\_size=(2,2)))

CNN\_model.add(Dropout(0.1))

CNN\_model.add(Flatten())

CNN\_model.add(Dense(256, activation='relu'))

CNN\_model.add(Dense(1, activation='sigmoid'))

5. S-a realizat antrenarea efectivă a modelului

CNN\_model.compile(optimizer=Adam(learning\_rate=0.0001),loss=BinaryCrossentropy(), metrics=['accuracy'])

history=CNN\_model.fit(train\_dataset, epochs=50, validation\_data=validation\_dataset)

6. S-au afișat graficele referitoare la parametrii de acuratețe și cost din timpul antrenării, rezultate din performanțele pe parcurs. Odată ce rezultatele au fost suficient de bune, s-a salvat modelul.

plt.subplot(121)

plt.plot(history.history['accuracy'])

plt.plot(history.history['val\_accuracy'])

plt.title('model accuracy')

plt.ylabel('accuracy')

plt.xlabel('epoch')

plt.subplot(122)

plt.plot(history.history['loss'])

plt.plot(history.history['val\_loss'])

plt.title('model loss')

plt.ylabel('loss')

plt.xlabel('epoch')

plt.show()

CNN\_model.save('emotion\_detection\_cnn\_50epochs\_Dropout\_v102.h5', include\_optimizer=True)

7. S-a realizat testarea efectivă a modelului pe datele de testare după care s-au etichetat predicțiile.

predicted\_labels = []

batch = data\_test.as\_numpy\_iterator().next()

test\_images, test\_labels = batch

predictions = CNN\_model.predict(test\_images)

for p in predictions:

if p < 0.5:

predicted\_labels.append(0)

else:

predicted\_labels.append(1)

print(CNN\_model.evaluate(data\_test))

print(classification\_report(test\_labels,predicted\_labels))

8. S-a reprezentat grafic matricea de confuzie:

cm = confusion\_matrix(test\_labels, predicted\_labels)

plt.figure()

plt.title('version 1.0.2')

sns.heatmap(cm,annot=True,fmt='g',xticklabels=['angry','not\_angry'], yticklabels=['angry','not\_angry'])

plt.show

Pentru seturile de antrenare s-au obținut performanțele de mai jos pentru acuratețe și cost:

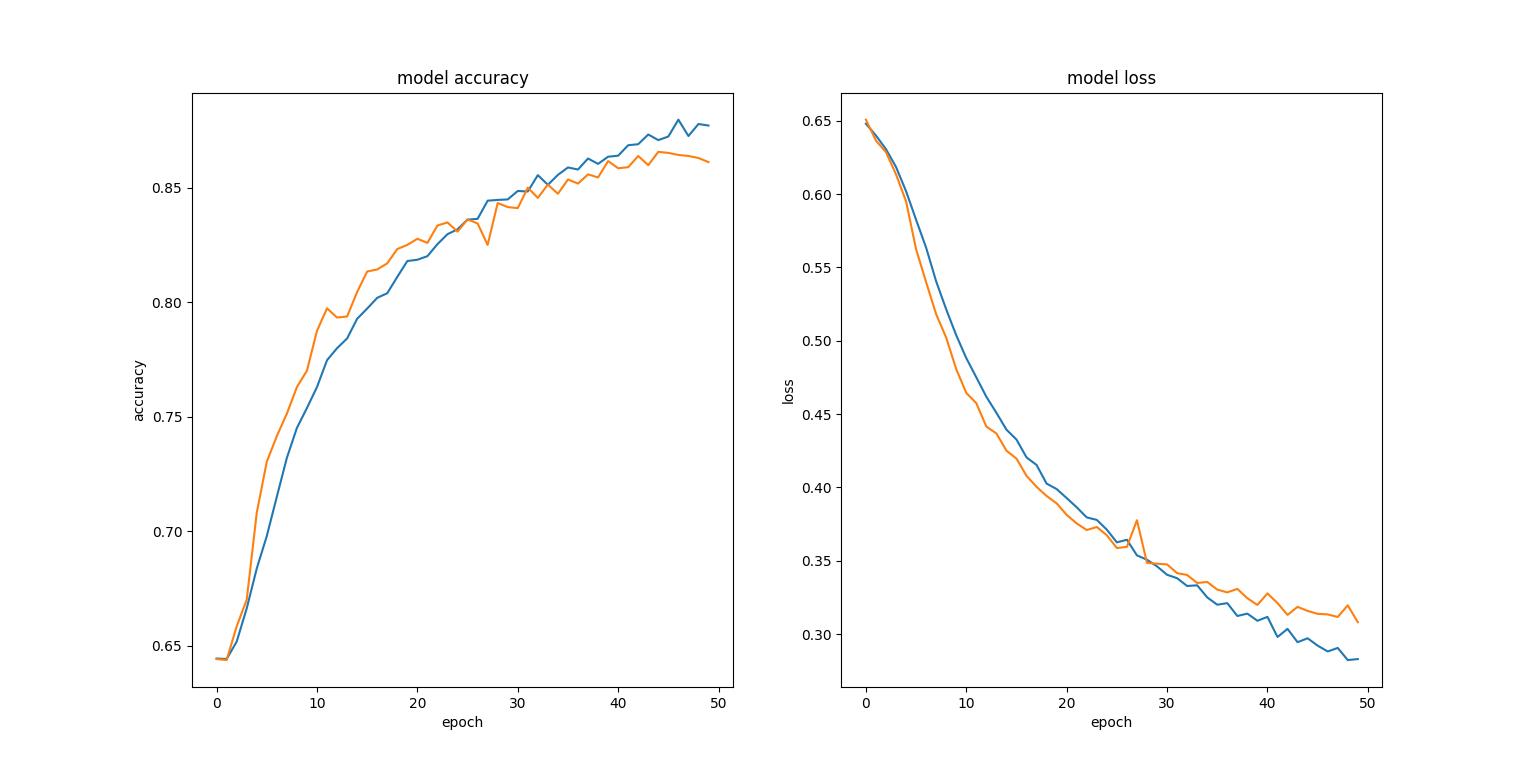


Figura 3.3 – Acuratețe și pierderi pentru 50 de epoci, learning rate = 0.001

Matricea de confuzie aferentă setului de testare, batch size 32:

A diagram of different colors

Description automatically generated

Figura 3.4 – Matricea de confuzie, batch size = 32

Acuratețea la testare este de 86%, iar cea la testare este de 92%. Acest lucru este unul normal, având în vedere că modelul testează mereu pe aceleași seturi de date, iar și iar. Indicatorul de performanță este afișat și în matricea de confuzie, pentru un batch ales aleator de 32 de imagini. Observăm așadar că din 32 de imagini, etichetele pentru 31 dintre ele au fost prezise cu succes (acuratețe 98%). Desigur, acest lucru poate varia, dacă am alege un batch de 1000 de imagini, acuratețea ar putea fi afectată, întrucât imaginile cu emoții de furie sunt în general cel mai greu de etichetat corect, din întreaga bază de date FER-2013. Totuși, aceasta ar trebui să graviteze în jurul numerelor 86-90%.

REFERINȚE BIBLIOGRAFICE

[1] Zahara, Lutfiah & Musa, Purnawarman & Prasetyo, Eri & Karim, Irwan & Musa, Saiful. (2020). The Facial Emotion Recognition (FER-2013) Dataset for Prediction System of Micro-Expressions Face Using the Convolutional Neural Network (CNN) Algorithm based Raspberry Pi. 1-9. 10.1109/ICIC50835.2020.9288560.

[2] Jie, Shao & Yongsheng, Qian. (2019). Three Convolutional Neural Network Models for Facial Expression Recognition in the Wild. Neurocomputing. 355. 10.1016/j.neucom.2019.05.005.

[3] Oguine, Ozioma & Kinfu, Kaleab & Oguine, Kanyifeechukwu & Bisallah, Hashim & Ofuani, Daniel. (2022). Hybrid Facial Expression Recognition (FER2013) Model for Real-Time Emotion Classification and Prediction. 10.48550/arXiv.2206.09509.

[4] Khaireddin, Yousif & Chen, Zhuofa. (2021). Facial Emotion Recognition: State of the Art Performance on FER2013.

[5] Bettadapura, Vinay. (2012). Face Expression Recognition and Analysis: The State of the Art. CoRR. abs/1203.6722.

[6] \*\*\* Classification in Machine Learning : An Introduction -<https://www.datacamp.com/blog/classification-machine-learning> – accesat la data 11.11.2023

[7] \*\*\* Convolutional Neural Networks - <https://www.ibm.com/topics/convolutional-neural-networks>

- accesat 12.11.2023

[8] \*\*\* Convolutional Neural Networks for Image Classification - <https://medium.com/@khwabkalra1/convolutional-neural-networks-for-image-classification-f0754f7b94aa> - accesat 12.11.2023

[9] \*\*\* Keras, <https://keras.io/> - accesat la data 13.11.2023

[10] \*\*\* Tensorflow, <https://www.tensorflow.org/> - accesat la data 13.11.2023

[11] \*\*\* Scikit-learn, <https://scikit-learn.org/stable/> - accesat la data 13.11.2023

[12] \*\*\* Seaborn, <https://seaborn.pydata.org/> - accesat la data 13.11.2023

ANEXE

ANEXA 1 – COD SURSĂ CNN

import tensorflow as tf

import matplotlib.pyplot as plt

from keras.models import Sequential, load\_model

from keras.layers import Conv2D,MaxPooling2D,Flatten,Dense,Dropout

from keras.optimizers import Adam

from keras.losses import BinaryCrossentropy

import numpy as np

from sklearn.metrics import confusion\_matrix, classification\_report

import seaborn as sns

train\_dir = "train"

test\_dir = "test"

labels = ("angry", "not\_angry")

# image batch generators

data\_train = tf.keras.utils.image\_dataset\_from\_directory(train\_dir,batch\_size=16,image\_size =(48,48))

data\_gen\_train = data\_train.as\_numpy\_iterator()

batch\_train = data\_gen\_train.next()

data\_test = tf.keras.utils.image\_dataset\_from\_directory(test\_dir,batch\_size=32,image\_size =(48,48))

data\_gen\_test = data\_test.as\_numpy\_iterator()

batch\_test = data\_gen\_test.next()

# show a full batch of labeled train images

images, cls = batch\_train

fig, axs = plt.subplots(4, 4, figsize=(12, 12))

axs = axs.flatten()

for i in range(len(images)):

axs[i].imshow(images[i].astype("uint8"))

axs[i].set\_title(f"Label: {labels[cls[i]]}")

axs[i].axis("off")

plt.show()

# image preprocessing & splitting

data\_train = data\_train.map(lambda x, y: (x/255, y))

data\_train.as\_numpy\_iterator().next()

data\_test = data\_test.map(lambda x, y: (x/255, y))

data\_test.as\_numpy\_iterator().next()

train\_dataset\_size = int(len(data\_train) \* 0.8)

validation\_dataset\_size = int(len(data\_train) \* 0.2)

train\_dataset = data\_train.take(train\_dataset\_size)

validation\_dataset= data\_train.skip(train\_dataset\_size).take(validation\_dataset\_size)

# CNN architecture design

CNN\_model = Sequential()

CNN\_model.add(Conv2D(16,(3,3),1, activation='relu', input\_shape=(48,48,3)))

CNN\_model.add(MaxPooling2D(pool\_size=(2,2)))

CNN\_model.add(Dropout(0.1))

CNN\_model.add(Conv2D(32, (3,3),1,activation='relu'))

CNN\_model.add(MaxPooling2D(pool\_size=(2,2)))

CNN\_model.add(Dropout(0.2))

CNN\_model.add(Conv2D(16, (3,3),1,activation='relu'))

CNN\_model.add(MaxPooling2D(pool\_size=(2,2)))

CNN\_model.add(Dropout(0.1))

CNN\_model.add(Flatten())

CNN\_model.add(Dense(256, activation='relu'))

CNN\_model.add(Dense(1, activation='sigmoid'))

CNN\_model.compile(optimizer=Adam(learning\_rate=0.0001),loss=BinaryCrossentropy(), metrics=['accuracy'])

history=CNN\_model.fit(train\_dataset, epochs=50, validation\_data=validation\_dataset)

#version 1.0.1

#accuracy = 85.13% la validare

plt.subplot(121)

plt.plot(history.history['accuracy'])

plt.plot(history.history['val\_accuracy'])

plt.title('model accuracy')

plt.ylabel('accuracy')

plt.xlabel('epoch')

plt.subplot(122)

plt.plot(history.history['loss'])

plt.plot(history.history['val\_loss'])

plt.title('model loss')

plt.ylabel('loss')

plt.xlabel('epoch')

plt.show()

CNN\_model.save('emotion\_detection\_cnn\_50epochs\_Dropout\_v102.h5', include\_optimizer=True)

#saved\_model= load\_model('emotion\_detection\_cnn\_50epochs\_Dropout\_v102.h5',compile=False)

predicted\_labels = []

batch = data\_test.as\_numpy\_iterator().next()

test\_images, test\_labels = batch

predictions = CNN\_model.predict(test\_images)

for p in predictions:

if p < 0.5:

predicted\_labels.append(0)

else:

predicted\_labels.append(1)

print(CNN\_model.evaluate(data\_test))

print(classification\_report(test\_labels,predicted\_labels))

cm = confusion\_matrix(test\_labels, predicted\_labels)

plt.figure()

plt.title('version 1.0.2')

sns.heatmap(cm,annot=True,fmt='g', xticklabels=['angry','not\_angry'], yticklabels=['angry','not\_angry'])

plt.show()