

Universitatea Națională de Știință și Tehnologie “Politehnica”
București

Facultatea de Electronică, Telecomunicații și Tehnologia Informației

*Autentificare folosind algoritmi de recunoaștere facială
cu sistem anti-spoofing*

Proiect de diplomă

Prezentat ca cerință parțială pentru obținerea
titlului de *Inginer* în domeniul *Electronică și Telecomunicații*,
programul de studii de licență *Electronică Aplicată*

Conducători Științifici

Conf. Dr. Ing. Mădălin-Corneliu FRUNZETE

As. Drd. Ing. Alin-Alexandru ȘERBAN

Absolvent

Ovidiu-Cristian IRIMESCU

2024

Universitatea "Politehnica" din București
Facultatea de Electronică, Telecomunicații și Tehnologia Informației
Program de studiu

Anexa 1

TEMA PROIECTULUI DE DIPLOMĂ
a studentului **IRIMESCU I. Ovidiu-Cristian, 441B**

1. Titlul temei: Autentificare folosind algoritmi de recunoaștere facială cu sistem anti-spoofing

2. Descrierea temei și a contribuției personale a studentului (în afara părții de documentare): Lucrarea presupune analiza metricilor, cât și implementarea unei aplicații de autentificare folosind algoritmi de recunoaștere facială, constând dintr-un sistem anti-fraudă ce verifică autenticitatea imaginii capturate cu ajutorul unei rețele convoluționale, suplimentată de un filtru ce analizează spectrul de frecvență al imaginilor. Resurse folosite în dezvoltarea proiectului: Limbaje de programare: Python (biblioteci OpenCV, Tensorflow etc.); Baza de date va fi implementată folosind serviciul de backend cloud computing FireBase.

3. Discipline necesare pt. proiect:
RNSF, TTI, BD, TPI

4. Data înregistrării temei: 2024-02-10 19:01:53

Conducător(i) lucrare,
As. drd. ing Alin ȘERBAN

Student,
IRIMESCU I. Ovidiu-Cristian

Conf. Dr. Ing. Mădălin FRUNZETE

Director departament,

Decan,
Prof. dr. ing. Mihnea UDREA

Cod Validare: 8c306250ed

Declarație de onestitate academică

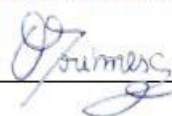
Prin prezenta declar că lucrarea cu titlul “ Autentificare folosind algoritmi de recunoaștere facială cu sistem anti-spoofing”, prezentată în cadrul Facultății de Electronică, Telecomunicații și Tehnologia Informației a Universității Naționale de Știință și Tehnologie POLITEHNICA București ca cerință parțială pentru obținerea titlului de *Inginer* în domeniul *Electronică și Telecomunicații*, programul de studii de licență *Electronică Aplicată*, este scrisă de mine și nu a mai fost prezentată niciodată la o facultate sau instituție de învățământ superior din țară sau străinătate.

Declar că toate sursele utilizate, inclusiv cele de pe Internet, sunt indicate în lucrare, ca referințe bibliografice. Fragmentele de text din alte surse, reproduse exact, chiar și în traducere proprie din altă limbă, sunt scrise între ghilimele și fac referință la sursă. Reformularea în cuvinte proprii a textelor scrise de către alți autori face referință la sursă. Înțeleg că plagiatul constituie infracțiune și se sancționează conform legilor în vigoare.

Declar că toate rezultatele simulărilor, experimentelor și măsurărilor pe care le prezint ca fiind făcute de mine, precum și metodele prin care au fost obținute, sunt reale și provin din respectivele simulări, experimente și măsurători. Înțeleg că falsificarea datelor și rezultatelor constituie fraudă și se sancționează conform regulamentelor în vigoare.

București,
24.06.2024

Absolvent,
Ovidiu-Cristian IRIMESCU



Cuprins

Listă de figuri	-----	i
Listă de tabele	-----	ii
Listă de abrevieri	-----	iii
Introducere	-----	1
1. Prelucrarea imaginii digitale	-----	3
1.1. Rețele neuronale convoluționale (CNN)	-----	3
1.2. Transformata Wavelet Discretă	-----	7
2. Algoritmi uzuali de recunoaștere a fețelor	-----	10
2.1. Principal Component Analysis (PCA) / Teorema Karhunen-Loeve	-----	10
2.2. Linear Discriminant Analysis (LDA)	-----	16
3. Anti-Spoofing	-----	19
3.1. Topologie	-----	19
3.2. Principii de funcționare	-----	21
3.3. State Of The Art	-----	24
4. Rezultate practice	-----	29
Concluzii	-----	45
Bibliografie	-----	47
Anexa 1 – Codul sursă al aplicației		

Listă de figuri

0.1	Prima poză digitală, inventată la NIST -----	2
1.1	Modelul unei rețele neuronale cu 3 straturi -----	3
1.2	Litera X -----	4
1.3	Primul nivel de overlapping -----	5
1.4	Tipuri de wavelet -----	7
1.5	Descompunerea unei imagini folosind 2-D DWT -----	7
1.6	Descompunerea unei imagini cu canal R dominant -----	8
1.7	Imaginea propusă pentru ilustrarea raportului de transformare -----	9
2.1	Imagine scrisă sub formă de combinație liniară -----	11
2.2	Grafic set de date -----	11
2.3	Ilustrare proiecție set de date -----	12
2.4	Grafic componente principale -----	13
2.5	Grafic scor discriminant -----	16
3.1	Topologia unei aplicații cu sistem Anti-Spoofing -----	19
3.2	Topologie tehnologii Anti-Spoofing -----	20
3.3	Semnal PPG, harta de amplitudine pentru o zonă înregistrată a intestinului -----	21
3.4	Componente sistem FaceID -----	22
3.5	Performanțe ACER framework-uri actuale -----	25
3.6	Performanțe EER framework-uri actuale -----	25
3.7	Evoluție metode deep-learning utilizate -----	27
4.1	Diagramă cod sursă -----	29
4.2	Subfoldere prezente în directorul cu bibliotecile -----	30
4.3	Spațiu ocupat de către biblioteci -----	30
4.4	Biblioteci main.py -----	31
4.5	Porțiuni cod main -----	32
4.6	Fereastra principală GUI -----	33
4.7	Moduri aplicație -----	33
4.8	Grafic evoluție memorie utilizată -----	34
4.9	Exemplu afișaj consolă -----	34
4.10	Ilustrare bază de date FireBase -----	35
4.11	Resursele necesare filtrului Anti-Spoofing -----	35
4.12	Rezultat practic aplicație cu detecție zâmbet -----	36
4.13	Detalii fișier poză -----	36
4.14	Platou persistență subiect 1 -----	38
4.15	Platou persistență subiect 2 -----	38
4.16	Platou persistență subiect 3 -----	38
4.17	Platou persistență subiect 4 -----	38
4.18	Platou persistență subiect 5 -----	38
4.19	Platou persistență subiect 6 -----	38
4.20	Platou persistență subiect 7 -----	39
4.21	Platou persistență subiect 8 -----	39
4.22	Durate persistențe recunoaștere -----	39
4.23	Evoluție coeficient subiect 1 -----	40

4.24	Evoluție coeficient subiect 2	-----	40
4.25	Evoluție coeficient subiect 3	-----	41
4.26	Evoluție coeficient subiect 4	-----	41
4.27	Evoluție coeficient subiect 5	-----	41
4.28	Evoluție coeficient subiect 6	-----	41
4.29	Evoluție coeficient subiect 7	-----	41
4.30	Evoluție coeficient subiect 8	-----	41
4.31	Algoritm Regresie Liniară	-----	43

Listă de tabele

2.1	Set de date 2D -----	11
3.1	Baze de date pentru antrenare Anti-Spoofing -----	26
4.1	Prima serie de subiecți -----	37
4.2	A doua serie de subiecți -----	37
4.3	Prima serie de subiecți pentru analiza 2 -----	40
4.4	A doua serie de subiecți pentru analiza 2 -----	40
4.5	Tabel rezultate pentru timp de encodare și procent memorie rulată -----	42
4.6	Tabel rezultate prezise pentru timp de encodare și procent memorie rulată -----	43

Listă de acronime

ACER *Average Classification Error Rate* – Rata Medie de Eroare la Clasificare

APCER *Attack Presentation Classification Error Rate* – Rata de Eroare la Clasificarea Prezentării Atacului

AS *Anti-Spoofing* – Anti-Fraudare

AUC *Area Under The Curve* – Aria de Sub Curbă

BGR *Blue Green Red* – Albastru Verde Roșu

BGRA *Blue Green Red Alpha* – Albastru Verde Roșu Nivel de Transparență

BMP *Bitmap*

BPCER *Bonafide Presentation Classification Error Rate* - Rata de Eroare la Clasificarea Prezentării Autentice

CNN *Convolutional Neural Network* – Rețea Neuronală Convoluțională

CPU *Central Processing Unit* – Unitate Centrală de Procesare

DRAM *Dynamic Random Access Memory* – Memorie Dinamică cu Acces Aleatoriu

DWT *Discrete Wavelet Transform* – Transformata Wavelet Discretă

EER *Equal Error Rate* – Rata de Eroare Egală

FAR *False Acceptance Rate* – Rata de Acceptare Falsă

FAS-SGTD *Face Anti-Spoofing using Spatial Gradient and Temporal Depth* – Anti-Spoofing facial folosind gradient spațial și profunzime temporală

FRR *False Rejection Rate* – Rata de Rejecție Falsă

GB *GigaByte* – 10^9 octeți

GHz *GigaHertz* - 10^9 Hertz

GIF *Graphics Interchange Format*

HEIF *High Efficiency Image Format*

HTER *Half Total Error Rate* – Rata Înjumătățită a Erorilor Totale

ID *Identifier*

IDE *Integrated Development Environment* – Mediu de dezvoltare integrat

IMDB-Wiki *Internet Movie Database Wikipedia* – Baza de date regăsită pe Internet precum Wikipedia pentru industria cinematografică

JPEG *Joint Photographic Experts Group*

JSON *JavaScript Object Notation* – Notarea Obiectelor JavaScript

LDA *Linear Discriminant Analysis* – Analiza Discriminantă Liniară

MiniFASNet *Minivision Face Anti-Spoofing Network* – Rețea Minivision pentru Anti-Fraudare Facială

PCA *Principal Component Analysis* – Analiza Componentelor Principale

PNG *Portable Network Graphics*

PPG *Photoplethysmography* – Fotopletismografie

RAM *Random Access Memory* – Memorie cu Acces Aleatoriu

ReLU *Rectified Linear Unit* – Funcție identitate redresată

RF *Recunoaștere Față*

RGB *Red Green Blue* – Roșu Verde Albastru

RSGB *Residual Spatial Gradient Block* – Bloc de Gradient Spațial Rezidual

SSAN *Spectral Spatial Attention Network* - Rețea de Atenție Spectral-Spațială

STPM *Spatio-Temporal Propagation Module* – Modul de Propagare Spațio-Temporală

SVG *Scalable Vector Graphics*

TIFF *Tagged Image File Format*

XML *eXtensible Markup Language* – Limbaj de Marcare Extensibil

INTRODUCERE

În era digitală în continuă expansiune, se conturează nevoia imperativă de securitate și accesibilitate în diversele sisteme și platforme. În acest context, dezvoltarea unei aplicații de autentificare bazată pe recunoașterea facială devine tot mai pertinentă și esențială. Această tehnologie înseamnă nu doar o metodă inovatoare de securizare a accesului, ci și o soluție intuitivă și comodă pentru utilizatori. Lucrarea de licență propusă vizează investigarea, proiectarea și implementarea unui sistem complex ce integrează algoritmi avansați de recunoaștere a feței, îmbinând cunoștințe din domeniul învățării automate, prelucrării imaginilor și securității cibernetice. Scopul fundamental al acestui proiect este să ofere o soluție eficientă și fiabilă pentru autentificarea utilizatorilor, reducând vulnerabilitățile de securitate și oferind în același timp o experiență fluidă și prietenoasă pentru aceștia.

Aplicația elaborată pentru autentificare facială se bazează pe algoritmi avansați de inteligență artificială, care este un domeniu pentru care se văd rezultate excelente din 2022 încoace după cercetări și tentative de aplicații extinse pe o perioadă îndelungată, cea mai veche încercare existând din mileniul trecut.

Trecerea de la un format analogic la un format digital este un proces în plină desfășurare pentru viața de zi cu zi, deși a început acum o jumătate de secol, însă boom-ul tehnologic, implicit dezvoltarea arhitecturii sistemelor de calcul, a condus la modificarea criteriilor de conversie, de aceea, pe măsură ce se trecea de la un microprocesor la altul, capacitățile de stocare s-au mărit, schimbându-se astfel și tehnicile software necesare pentru manipularea acestor noi aparate.

Un punct de cotitură poate fi considerat portretul lui Walden Kirsch, ce este prima imagine digitală, creată în 1957 prin scanarea pozei pe care Russell Kirsch o avea la îndemână cu fiul său de doar 3 luni.

Chintesența unei imagini în format electronic este existența unui instrument matematic puternic: o imagine este văzută precum o matrice, iar un singur element din această matrice poartă denumirea de pixel. Folosindu-ne de noțiunea de pixel, se pot deduce atribute importante prin care se poate caracteriza o figură.

Un atribut extrem de important este unul cantitativ, rezoluția: dacă la acel moment (1957), fotografia avea o rezoluție de 176x176 pixeli, acum un telefon mobil precum Iphone 13 Pro produce imagini de o rezoluție de 4032x3024 pixeli, ceea ce este o diferență de peste 2 ordine de mărime.

Un alt atribut cantitativ este numărul de canale al unei poze. O poză color este interpretată ca și combinația la nivel de pixel dintre 3 culori primare în tehnologia informației: roșu (R- red), verde (G- green) și albastru (B- blue). În prezent, orice imagine funcționează în format RGB, fiind o pondere între 3 medii de culoare.

Fiecărui pixel îi este asociat un mediu de culoare, un așa-zis strat, intensitatea fiind dată de o valoare, 0 fiind cea mai mică, iar 255, cea mai mare, așadar o imagine modernă este interpretată precum un tensor, o matrice multistrat, iar pentru a putea face prelucrări la nivel larg, este indispensabil să avem parte de un sistem de calcul suficient de puternic, excluzându-se computerele produse înainte de 2014, ele fiind uzate moral și lente pentru asemenea operații.



Figura 0.1: Prima poză digitală, inventată la NIST

Sursa: [1]

Se remarcă în ziua de astăzi următoarele formate de imagini digitale:

BMP (Bitmap): dezvoltat de Windows, fiind printre cele mai vechi.

GIF (Graphics Interchange Format): introdus de CompuServe în 1987 și folosit pentru imagini color dinamice

TIFF (Tagged Image File Format): dezvoltat de Aldus Corporation în 1986

SVG (Scalable Vector Graphics)

RAW: format de dimensiune mare, de obicei de ordinul Mega, ce elimină compresia și redă imaginea direct capturată de pe senzorul camerei

WebP: dezvoltat de Google, cu o compresie bună

HEIF (High Efficiency Image Format): format modern ce suportă și imagini în mișcare

JPEG (Joint Photographic Experts Group): creat în 1992, fiind cel mai utilizat standard de compresie din lume

PNG (Portable Network Graphics): realizat în 1996, fiind un standard de compresie fără pierdere de informație.

Ultimele 2 formate vor fi de interes pentru această lucrare, îndeosebi formatul PNG, deoarece pe lângă absența pierderii de calitate față de JPEG, PNG suportă transparența, ceea ce înseamnă că anumiți pixeli pot fi total sau parțial transparenți, permițând imaginii să fie suprapusă pe un fundal fără a afecta fundalul, de asemenea, PNG este mai bine adaptat pentru imagini cu text, grafică sau ilustrații cu zone mari de culoare uniformă și margini netede. Deoarece utilizează o compresie fără pierderi, acesta este mai potrivit pentru imagini în care este importantă păstrarea detaliilor precise și a transparenței, așadar pretează mai bine pentru aplicații de recunoaștere facială.

CAPITOLUL 1: PRELUCRAREA IMAGINII DIGITALE

1.1. Rețele neuronale convoluționale (CNN)

O rețea neuronală convoluțională este o replică informatică a unui neuron uman, cu excepția că este folosită pentru aplicații de recunoaștere ale unei imagini, de exemplu o aplicație simplă este discernarea caracterelor "X" și "O", pentru a crea un jucător digital, un așa-zis bot pentru un joc celebru precum "Tic-tac-toe".

O rețea neuronală obișnuită prezintă:

- *un strat de intrare (în cazul figurii 1.1 avem 3 elemente I_{1-3}), în care avem ca date de intrare valori independente, de pildă, dacă dorim proiectarea unui sistem inteligent de predicție a apariției ploii, ca date de intrare putem avea valorile presiunii atmosferice pe o anumită perioadă de timp, pentru care avem asociate o anumită etichetă (label), ce ia de obicei valori binare, în acest caz având de-a face cu un sistem de învățare supervizată, pentru că vor fi preluate valori cunoscute (din trecut în cazul nostru, cu o referire la faptul dacă a plouat sau nu în circumstanța acelor valori);*

- *un strat ascuns (în cazul figurii 1.1 avem 4 elemente H_{1-4}), ce are rolul de a regulariza, preveni supraînvățarea, cât și creșterea capacității de învățare, prin crearea unor relații mai complexe între date;*

- *un strat de ieșire (în cazul figurii 1.1 avem un singur element y , însă pot fi mai multe elemente), care prin aplicarea finală a unei funcții matematice oferă un output, o ieșire, o predicție.*

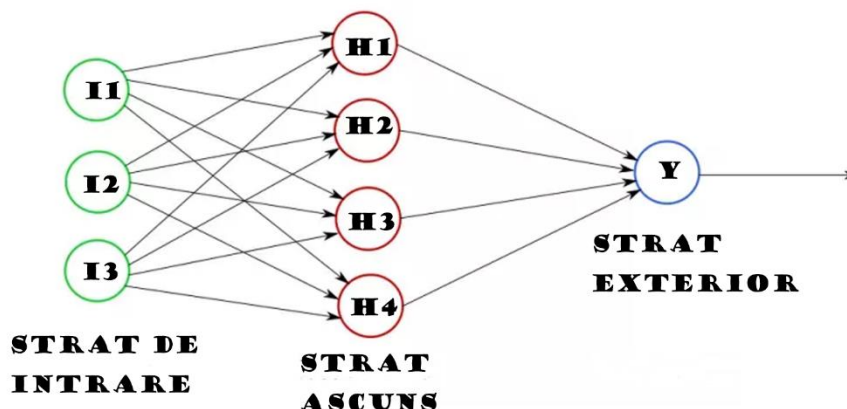


Figura 1.1: Modelul unei rețele cu 3 straturi

În cazul unei imagini, reprezentarea digitală este o matrice, de aceea, ca intrare, se vor desfășura elementele matricei pe o singură coloană (notând elementele cu $a_{i,j}$ o să avem desfășurarea pe coloană $a_{1,1}, a_{1,2}, a_{1,3} \dots a_{n,n-2}, a_{n,n-1}, a_{n,n}$). Dacă avem de-a face cu o imagine 6x6, vom avea 36 noduri în straturi de intrare, ceea ce ar conduce la 36 legături pentru un singur element din stratul de intrare, până acum, acest lucru nu impune probleme mari, însă dacă vorbim despre o imagine 100x100, o să existe 10000

legături pentru un singur nod din stratul ascuns, ceea ce conduce la probleme computaționale în privința eficienței, motiv pentru care se preferă altă abordare.

În consecință, o rețea convoluțională face o clasificare fezabilă prin reducerea numărului de noduri de intrare, tolerarea micilor translații ale pixelilor din imagine, cât și prin bazarea pe corelația observată în imagini complexe.

Funcționarea este următoarea: unei imagini date ca intrare îi este aplicată un filtru (kernel), ce este de obicei o matrice mai mică de dimensiune 3x3. Acest filtru se aplică matricei imaginii din input, parcurgându-se prin translație de la stânga la dreapta pe întreaga matrice, până când toate rândurile matricei au fost parcurse. Această parcurgere se efectuează făcând produsul scalar dintre porțiunea matricei mari și kernel.

Vom considera un model simplist matriceal al literei X, a cărei imagini este ilustrată în figura 1.2: [2]

$$X = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Unde, valorile de 1 reprezintă pixelii de culoarea neagra, de intensitate maximă, iar valorile de 0 reprezintă pixelii de culoare albă, de intensitate maximă.

Vom considera kernelul (filtrul) K, ce reprezintă porțiunea din dreapta sus a figurii 1.2:

$$K = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$



Figura 1.2

Ulterior, prin transpunerea lui K începând din colțul din stânga sus al matricei X, o să avem de efectuat produs Hadamard între:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{și} \quad \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

Rezultând

$$\begin{pmatrix} 1x0 & 0x0 & 0x1 \\ 0x0 & 1x1 & 0x0 \\ 0x1 & 0x0 & 1x0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Pentru acest rezultat, efectuând ”produs scalar”, adică însumând elementele, obținem valoarea 1, ce poate fi însumată cu un bias, spre exemplu, -2, rezultând valoarea -1. Valoarea de adineauri va reprezenta primul element din matricea F, numită mapă de caracteristici sau Feature Map.

Făcând o deplasare spre coloana a doua a matricei X, se efectuează din nou produs Hadamard, se însumează toate elementele matricei rezultate, apoi prin adunarea cu bias-ul se obține al doilea element al matricei F, mai exact valoarea -2.

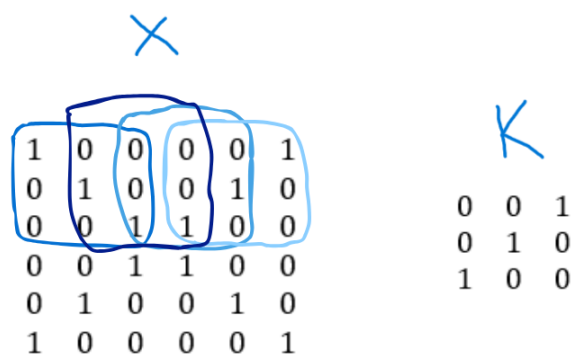


Figura 1.3: primul nivel de overlapping

Avansând pe a 3-a coloană, parcurgând aceleași operații, obținem -1, după care, pentru a patra coloană, vom avea elementul 1, întrucât avem potrivire pentru elementele de pe diagonala secundară.

Analog, respectând algoritmul, se face overlapping pentru celelalte rânduri, ultimul nivel de overlapping fiind cel ce începe de la rândul patru al matricei X și vom avea următorul Feature Map:

$$F = \begin{pmatrix} -1 & -2 & -1 & 1 \\ -2 & -1 & 1 & -1 \\ -1 & 1 & -1 & -2 \\ 1 & -1 & -2 & -1 \end{pmatrix}$$

Următorul pas este să trecem F printr-o funcție de activare. Funcțiile de activare sunt funcții matematice utilizate în rețelele neuronale pentru a introduce non-linearități în model, permițând acestuia să învețe relații complexe între intrările și ieșirile rețelei. Dintre astfel de funcții se numără: funcția liniară, sigmoidală, tangenta hiperbolică și ReLU. Cea mai eficientă funcție pentru astfel de aplicații este ReLU (Rectified Linear Unit), care în ciuda simplității sale, reduce problema dispariției gradientului.

ReLU se definește astfel:

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

În acest sens, va rezulta matricea $F' = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$

Operația de max pooling oferă ulterior o reducere a dimensiunii lui F' , selectându-se valoarea maximă din fiecare regiune de pooling și eliminându-se celelalte valori. Concret, se extrage elementul maxim aferent fiecărei porțiuni de dimensiune 2×2 ce reprezintă un colț al matricei F' .

Se evaluează pe rând "colțurile":

$$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \text{ pentru care maximul este } 0;$$

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \text{ pentru care maximul este } 1;$$

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \text{ pentru care maximul este } 1;$$

$$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \text{ pentru care maximul este } 0;$$

$$\text{Se obține matricea } P = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

P se desfășoară pe coloană și tocmai acum se poate face o întoarcere către o rețea neuronală obișnuită ce poate avea ponderi inițializate sub o distribuție normală sau Gauss, care prin operații continue de ajustare, specifice antrenării rețelei prin back-propagation, vor lua valori aproximative precum: -0,8 (corespunzătoare lui 0); -0,07 (corespunzătoare lui 1); 0,2 (corespunzătoare lui 1); 0,17 (corespunzătoare lui 0). Produsul ponderilor cu elementele din intrare se însumează ulterior cu un bias de valoare 0,97, rezultatul este trecut printr-o funcție ReLU, după aceea, se va face o dihotomie în stratul de ieșire, la prima ieșire, prima pondere având valoarea -1.3 și bias-ul 1.45, iar la a doua ieșire, a doua pondere are valoarea 1.3 și bias-ul -0.45. Făcând calculele, pentru prima ieșire, rezultatul este aproximativ 0, reprezentând probabilitatea ca imaginea oferită spre clasificare să fie un "0", iar pentru a doua ieșire, rezultatul este în jur de 1, reprezentând probabilitatea ca această imagine să fie "X". De asemenea, pentru îmbunătățirea și luarea unei decizii cât mai rapide pe baza acestor probabilități, se poate folosi o funcție softmax pentru rotunjirea către valoarea binară corespunzătoare.

1.2. Transformata Wavelet Discretă

Transformata Wavelet este descrisă de următoarea formulă:

$$F(\tau, s) = \frac{1}{\sqrt{|s|}} \int_{-\infty}^{+\infty} f(t) \psi\left(\frac{t-\tau}{s}\right) dt$$

s – parametru de scalare

$\psi^*(t)$ – conjugata lui $\psi(t)$, numit wavelet

Aceasta este salutară în analizarea unui semnal pe frecvențe diferite la rezoluții diferite, fiind o analiză multirezoluție. [3]

$$DWT(x)(a,b) = \sum_{n=-\infty}^{\infty} x(n) \cdot \psi_{a,b}(n)$$

$x(n)$ este semnalul de intrare sau semnalul original

$\psi_{a,b}(n)$ este funcția de undă (wavelet) scalată și translatată, dedusă de parametrul de scală și translație a și b

(a,b) sunt parametrii de scalare și translație care controlează cum este aplicată funcția de undă la semnalul de intrare

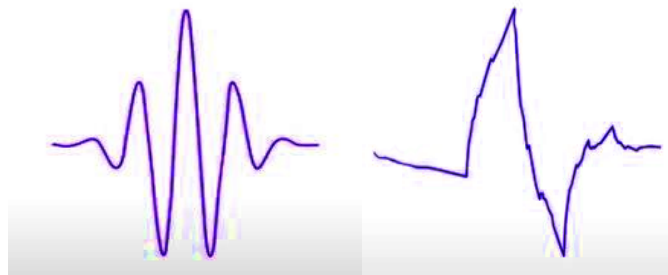


Fig. 1.4: tipuri de wavelet

O funcție de tip wavelet are alura unei unde seismice, începe cu o amplitudine nulă, crește, după aceea scade. Cu această transformată se pot descompune imaginile în mai multe sub-semnale sau se poate extrage nivelul de zgomot prezent în acestea.

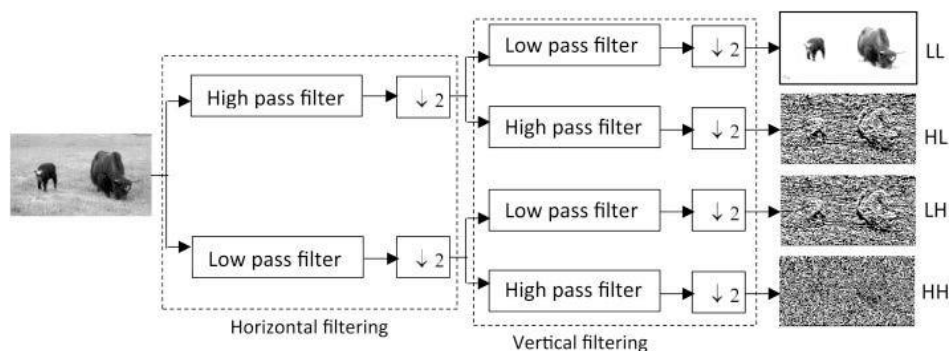


Fig. 1.5: Descompunerea unei imagini folosind 2-D DWT [4]

Un semnal este trecut prin două filtre, filtrul trece-sus și filtrul trece-jos. Imaginea este apoi decompusă în componente de frecvență înaltă (detalii) și componente de frecvență joasă (aproximări).

La fiecare nivel, obținem 4 sub-semnale. Aproximarea arată o tendință generală a valorilor pixelilor, iar detaliile reprezintă 3 tipuri de componente: orizontale, verticale și diagonale.

Dacă detaliile nu sunt semnificative, pot fi neglijate, evaluate nule, iar astfel nu au un impact asupra imaginii, realizându-se astfel filtrare și compresie.

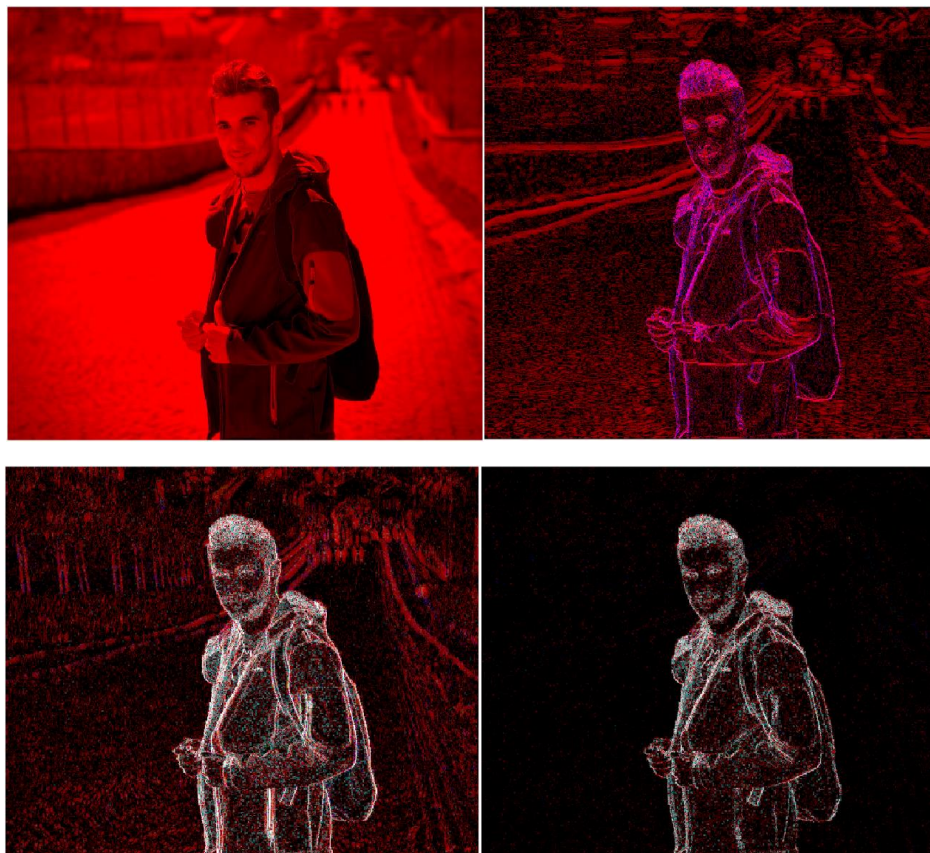


Fig. 1.6: Descompunerea unei imagini cu canal R dominant

Există o multitudine de avantaje ale Transformatei Wavelet Discretă (DWT) în cazul unei aplicații de autentificare folosind recunoaștere facială:

În primul rând, se reprezintă eficient informația, DWT comprimă imaginea fără a pierde detalii importante, ceea ce este vital în cazul gestionării și analizării unei cantități mari de date;

În al doilea rând, se localizează prompt caracteristici, cum ar fi caracteristici faciale cheie precum marginile feței, forma ochilor, mărimea buzelor, a nasului;

În al treilea rând, nu există sensibilitate la variații în iluminare și poziție, se pot extrage astfel caracteristici invariante;

Un avantaj important constă în oportunitatea de a elimina semnalul de fundal nedorit sau redundant, permitându-se astfel algoritmilor de recunoaștere facială să se concentreze strict pe trăsăturile semnificative ale feței;

Și nu în ultimul rând, suportul multirezoluțional oferă capacitatea de a analiza imagini la diverse rezoluții, ceea ce implică faptul că se vor putea analiza fețe la distanțe diferite sau în condiții mai dificile, reducându-se astfel costurile pentru necesitatea posedării unui senzor foto de o calitate superioară.

Există totuși cazuri în care aplicarea DWT conduce la obținerea unei imagini de dimensiune mai mare, lucru ce se datorează lucrului cu o imagine de rezoluție mare, în cazul experimental propriu s-a lucrat cu o imagine .jpg de rezoluție 2048x1686 și s-au obținut următoarele rezultate în urma elaborării unui algoritm în Python cu librăria *pywt*, mai exact, s-a aplicat Transformata Wavelet Discretă asupra imaginii inițiale convertită în format gri, ulterior s-a reconstruit imaginea finală din toți coeficienții obținuți anterior: imaginea inițială a avut dimensiunea de 796.83 KB, iar cea finală dimensiunea de 26976 KB, fiind un raport de transformare de 33.85. Algoritmul a fost adaptat conform documentației referinței [5].

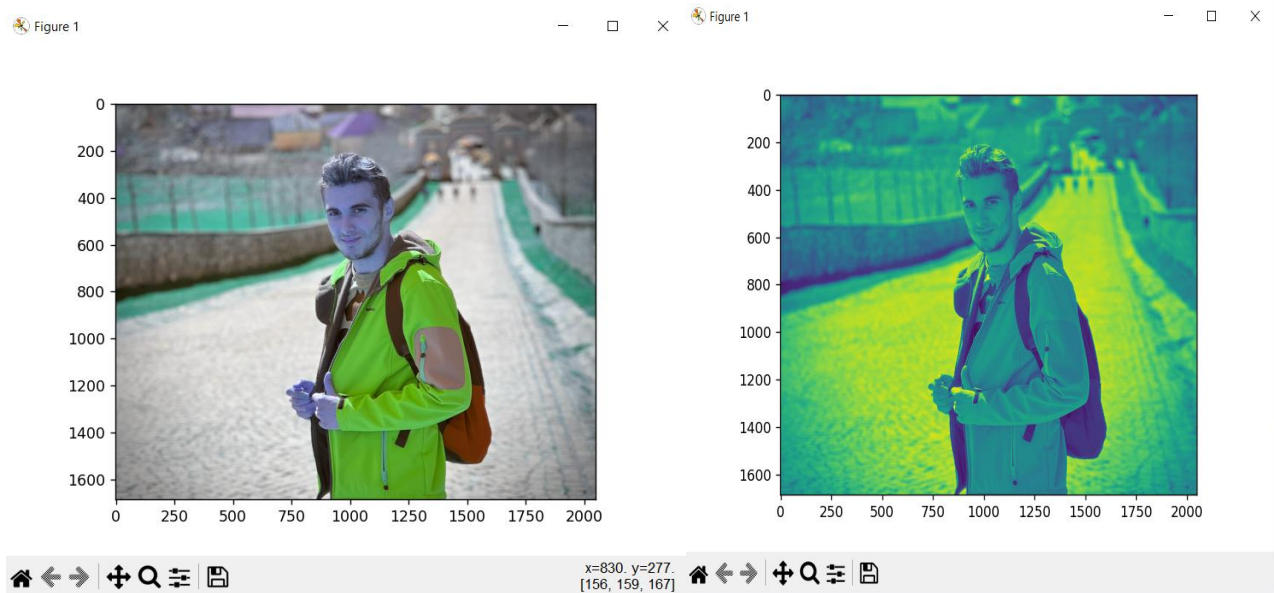


Fig. 1.7: Imaginea propusă pentru ilustrarea raportului de transformare

CAPITOLUL 2: ALGORITMI UZUALI DE RECUNOAȘTERE A FEȚELOR

2.1. Principal Component Analysis (PCA) / Teorema Karhunen-Loeve

Tehnica regăsită în PCA este utilă întrucât facilitează efectuarea de analize pe seturi de date multidimensionale, care în mod tradițional, erau limitate de spațiul tridimensional (3D). Astfel, se poate face un grafic în format 2D pentru un grafic care ar fi fost 3D, 4D, 5D și așa mai departe.

Este mult mai fezabilă această tehnică decât utilizarea unei matrici de covarianță, care prin existența multor dimensiuni, mărește volumul de calcul din cauza combinațiilor extrem de numeroase ale indicilor.

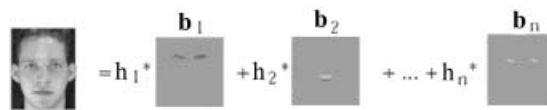

$$\text{Image} = h_1 \cdot \mathbf{b}_1 + h_2 \cdot \mathbf{b}_2 + \dots + h_n \cdot \mathbf{b}_n$$

Fig. 2.1: imagine scrisă sub formă de combinație liniară [6]

Pentru a înțelege cât mai bine fundamentul tehnicii, se propune un exemplu tabelat în care au fost extrase scorurile a 6 elevi pentru 2 discipline distincte. Scopul este să se afle care trăsătură, în cazul nostru, care scor este mai important în acest set de date și de asemenea, se poate realiza figurarea punctelor pe grupe, lucru numit clusterizare. Clusterizarea are ca scop găsirea unor grupuri astfel încât instanțele din același grup să fie mai similare între ele decât cu instanțele altor grupuri, acest lucru este salutar segmentării imaginilor: identificarea regiunilor având trăsături similare, pentru detecția obiectelor. [7]

	Student 1	Student 2	Student 3	Student 4	Student 5	Student 6
Scor 1	10	11	8	3	2	1
Scor 2	6	4	5	3	2.8	1

Tabel 2.1: Set de date 2D

Având în vedere că sunt tabelate doar două scoruri, vom avea de-a face cu un grafic obișnuit pe două dimensiuni, ceea ce simplifică exemplificarea.

Graficul obținut în urma considerării valorilor axei Ox valorile de la Scor 1, respectiv valorilor axei Oy cele de la Scor 2 este următorul:

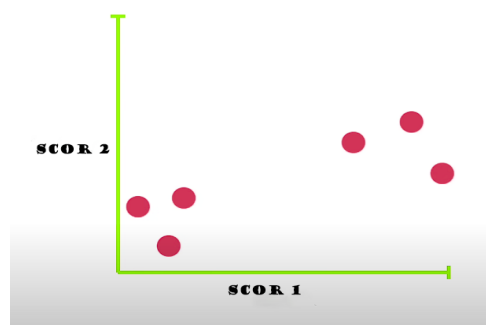


Fig. 2.2: Grafic set de date

Următorul pas este să se translateze datele astfel încât centrul să fie în origine, după aceea se găsește o dreaptă ce "cuprinde" cel mai bine punctele de pe grafic.

Acest lucru se face ținând cont de Teorema lui Pitagora, se pornește cu o dreaptă ce este perpendiculară cu axa Ox, se face o proiecție a punctelor pe acea dreaptă și se calculează distanța fiecărui punct proiectat până la origine, așa cum se poate ilustra în figura de mai jos, unde avem marcată o distanță D .

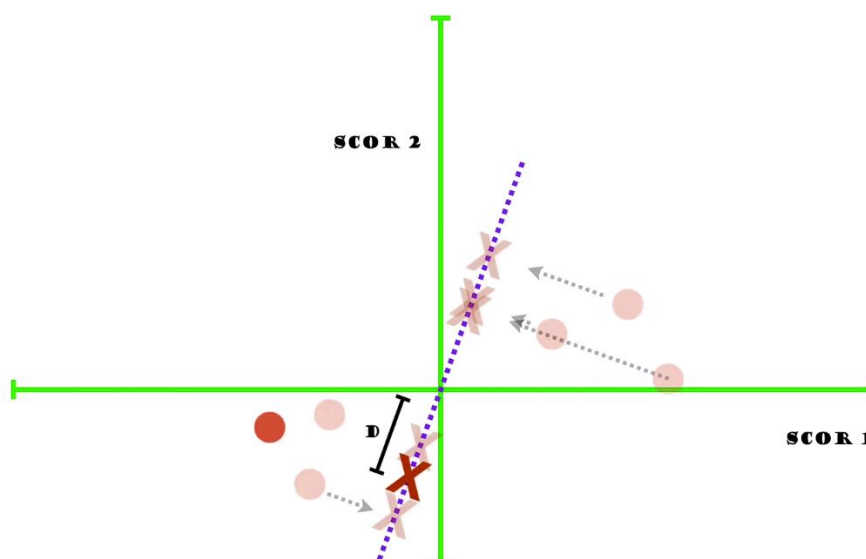


Fig. 2.3: Ilustrare proiecție set de date

Toate cele șase distanțe se ridică la pătrat și se însumează. Se variază panta dreptei, iar procesul se repetă până când se găsește o valoare maximă a sumei, moment în care se reține panta dreptei și se marchează drept cea mai potrivită.

Media aritmetică a tuturor acestor sume de distanțe pătratice poartă denumirea de Eigenvalue.

Pentru acest exemplu, se obține panta 0.25, semnificând faptul că majoritatea datelor sunt împrăștiate de-alungul axei Scor 1. Panta semnifică practic că pentru PC1 este nevoie de 4 elemente din Scor 1 și un element din Scor 2, lucru cunoscut în matematică drept combinație liniară, pentru care un exemplu concret în recunoaștere facială este ilustrat în figura 2.1.

În continuare, realizând un triunghi dreptunghic ce are ca ipotenuză această dreaptă găsită (de pantă 0.25 în exemplu), obținem valoarea ipotenuzei aplicând Pitagora:

$$c = \sqrt{a^2 + b^2}$$

$$a = 4$$

$$b = 1$$

De unde se află $c = 4.12$. Ulterior se efectuează o normare astfel încât c să aibă valoare unitară.

Astfel, noile valori vor fi $a = 0.97$ și $b = 0.242$, componentele vectorului v , cu $v_x = a$ și $v_y = b$.

Acest vector unitate poartă denumirea de Eigenvector, aferent lui PC1.

În continuare, pentru a afla PC2, se găsește dreapta perpendiculară cu dreapta de pantă 0.25, astfel, ținând cont că produsul pantelor a două drepte ortogonale este -1, deducem că a doua dreaptă are panta -4, deci normând, obținem $a' = -0.242$ și $b' = 0.97$. În final, pentru a obține graficul final, se efectuează o rotație a dreptelor astfel încât acestea să se îmbine pe axele Ox și Oy, iar ulterior se trasează punctele inițiale ținând cont de proiecțiile efectuate anterior.

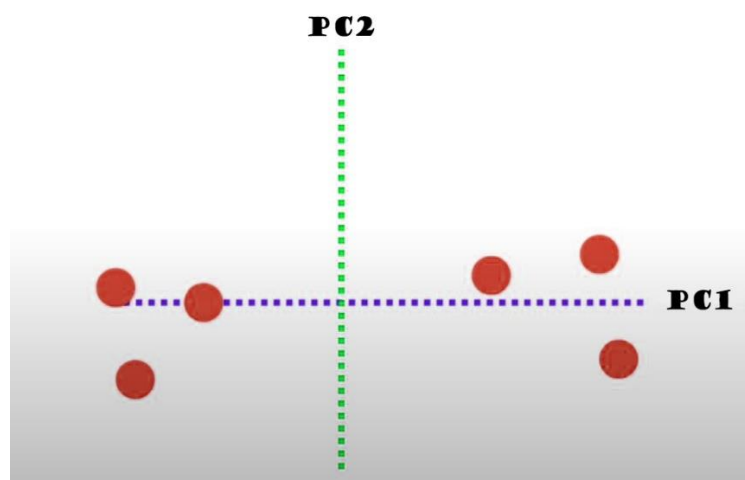


Fig. 2.4: Grafic componente principale

Prin valorile proprii (eigenvalues), se află măsurători ale variațiilor prezente în setul de date.

De exemplu, dacă $PC1 = 16$, iar $PC2 = 5$, variația totală este 21, iar individual $PC1$ este responsabil pentru 76% din variații, iar $PC2$ pentru 24% din variații.

Cu alte cuvinte, pentru un set mai mare de date, extinzând la n componente principale:

$PC1$ – indică direcția primei celei mai mari variații de date

$PC2$ – indică direcția cu a doua cea mai mare variație de date

$PC3$ - indică direcția cu a treia cea mai mare variație de date

PCn - indică direcția cu a n -a cea mai mare variație de date [2]

Aferent cazului nostru și pentru a fi mai riguroși matematic, se efectuează aceste operații:

Se consideră un set de m imagini de dimensiune $N \times N$. Aceste imagini se convertesc în vectori de dimensiune $N \times N$, apoi se calculează media acestor vectori de fețe ψ , după care fiecare vector x_i este scăzut cu ψ , obținând a_i .

$$\psi = \frac{1}{m} \sum_{i=1}^m x_i$$

$$a_i = x_i - \psi$$

Acum luăm toți vectorii feței astfel încât să obținem o matrice de dimensiune $N^2 \times M$.

$$A = [a_1, a_2, a_3, a_4, \dots, a_m].$$

Ulterior, găsim matricea de covarianță prin înmulțirea lui A cu A^T . Matricea A are dimensiunile $N^2 \times M$, deci A^T are dimensiunile $M \times N^2$. Când acestea sunt înmulțite, obținem o matrice de dimensiune $N^2 \times N^2$, ceea ce ne oferă N^2 vectori proprii de dimensiune N^2 , ceea ce nu este eficient din punct de vedere computațional. Așadar, calculăm matricea noastră de covarianță prin

înmulțirea lui A^T cu A . Acest lucru ne oferă o matrice de dimensiune $M * M$ care are M vectori proprii de dimensiune M .

În acest pas calculăm valorile proprii și vectorii proprii ai matricei de covarianță de mai sus folosind formula de mai jos: [8]

$$\begin{aligned} A^T A v_i &= \lambda_i v_i \\ A A^T A v_i &= \lambda_i A v_i \\ C' u_i &= \lambda_i u_i, \text{ unde } C' = A A^T \text{ și } u_i = A v_i \end{aligned}$$

Din afirmația de mai sus se poate concluziona că C și C' au valori proprii comune și vectorii proprii sunt legați de ecuația $u_i = A v_i$. Astfel, cele M valori proprii (și vectorii proprii) ale matricei de covarianță oferă M cele mai mari valori proprii (și vectorii proprii) ai lui C' .

După aceea calculăm vectorii și valorile proprii ale acestei matrice de covarianță reduse și îi mapăm în C' folosind formula lui u_i .

Acum selectăm cei K vectori proprii ai lui C' corespunzători celor K mai mari valori proprii. Acești vectori proprii au dimensiunea N^2 . În acest pas, am folosit vectorii proprii obținuți în pasul anterior. Luăm vectorii feței de antrenament normalizați (x_i) și reprezentăm fiecare vector de față într-o combinație liniară a celor mai buni K vectori proprii.

$$x_i - \psi = \sum_{j=1}^K w_j u_j, \text{ unde } u_j \text{ se numesc EigenFaces}$$

Următorul pas este să luăm coeficienții u_j și reprezentăm fețele de antrenament sub forma unui vector al acestor coeficienți, y_i .

$$y_i = \begin{pmatrix} w_{1,i} \\ w_{2,i} \\ w_{3,i} \\ \vdots \\ w_{n,i} \end{pmatrix}$$

Dacă avem o față necunoscută f , trebuie mai întâi să preprocesăm fața pentru a o centra în imagine și pentru a avea dimensiuni comune ca și fața de antrenament, pe urmă scădem fața din fața medie ψ .

$$\phi = f - \psi$$

În continuare, se proiectează vectorul normalizat în spațiul propriu pentru a obține combinația liniară a eigenfaces-urilor.

$$\phi = \sum_{i=1}^K w_i u_i$$

Se generează astfel vectorul Ω .

$$\Omega = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_n \end{pmatrix}$$

Se ia vectorul de sus și se scade din imaginea din lotul de antrenare pentru a obține cea mai mică distanță dintre vectorii de antrenament și vectorii de testare.

$$e_r = \min_l \| \Omega - \Omega_l \|$$

Dacă se obține e_r sub o anumită toleranță T_r , atunci fața este recunoscută cu una din setul de antrenament (l spre exemplu), iar în caz contrar, nu există potrivire cu nicio față din lotul de antrenament.

2.2. Linear Discriminant Analysis (LDA)

Analiza discriminantă lineară este o metodă similară cu PCA, al cărei scop este să maximizeze separația dintre două sau mai multe grupuri de date.

Modul în care funcționează LDA constă în găsirea unei proiecții a datelor într-un spațiu de dimensiuni mai mici, astfel încât să maximizeze dispersia între clase (în cazul aplicației de recunoaștere facială, între diferitele fețe) și să minimizeze dispersia în cadrul claselor identice (adică între imagini diferite cu aceeași persoană). Această proiecție se găsește prin calculul vectorilor proprii și al valorilor proprii ale matricei de dispersie a datelor.

Tehnica poate fi descrisă prin următoarea formulă:

$$LD = \alpha_1 X_1 + \alpha_2 X_2,$$

unde X_1 și X_2 reprezintă variabile de interes, iar α_1 și α_2 sunt ponderi

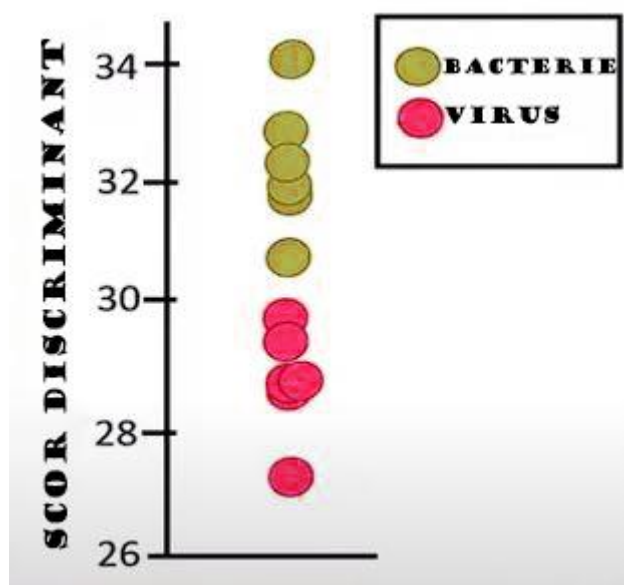


Fig. 2.5: grafic scor discriminant [9]

Pentru exemplul din figura de mai sus, s-a propus extrapolarea a doi parametri: concentrația de proteină C reactivă [mg/L] și temperatura, pentru a putea face o clasificare între prezența unui virus sau al unei bacterii în corpul uman. Se poate observa că se poate face o separare clară folosind o linie orizontală. O separare bună este redată de o distanță mare între valorile medii individuale aferente fiecărei clase. Altfel spus, separarea dintre clase depinde de varianța dintre grupuri și varianța din interiorul unui grup.

$$\sigma^2 = \frac{\sum (x_i - \mu)^2}{n-1}, \text{ varianța}$$

În formă matriceală, separarea se exprimă astfel:

$$S = W^{-1} B, \text{ unde}$$

B este matricea de covarianță dintre clase și W matricea de dispersie în cadrul claselor

Pentru a afla W , se calculează matricea de covarianță pentru fiecare clasă în parte (pentru clasa "Virus" și clasa "Bacterie" spre exemplu), apoi se face media aritmetică a acestora pentru a obține W . Dacă dimensiunile celor 2 matrici nu corespund, se face o medie ponderată, precum formula:

$$W = \frac{(n1-1)cov(A)+(n2-1)cov(B)}{n1+n2-2}$$

Ulterior, pentru a calcula B , se calculează matricea de covarianță pentru toate datele, T , deoarece se cunoaște următoarea egalitate:

$$T = B + W,$$

$$\text{De unde rezultă } B = T - W$$

Știind W , se face matricea inversă W^{-1} , ceea ce este salutar pentru aflarea lui S . Următorul pas constă în aflarea vectorilor proprii pentru această matrice, lucru ușor de făcut utilizând un software online. Valorile primului vector propriu constituie ponderile α_1 și α_2 , cu care puteam afla un scor nestandardizat, însă dacă se dorește un scor standardizat, se face o rescalare a valorilor astfel încât varianța mediată a scorurilor să fie 1. Astfel, se face media aritmetică a varianțelor individuale, iar apoi valorile inițiale ale ponderilor se împart la radical din această medie pentru a obține noile ponderi.

Este esențial să se ia în considerare și limitările LDA în recunoașterea facială, cum ar fi necesitatea de a avea un număr suficient de eșantioane pentru antrenarea fiecărei clase și presupunerea că datele sunt distribuite normal. De asemenea, LDA poate fi sensibilă la datele cu dimensiuni mari sau la prezența unor clase de fețe extrem de variate, totuși, este un instrument matematic puternic cu avantaje precum: maximizarea separației între clase, reducerea dimensionalității, crearea de caracteristici robuste, cât și eficiență computațională.

CAPITOLUL 3: ANTI-SPOOFING

3.1. Topologie

Prin Anti-Spoofing, înțelegem ansamblul de tehnici folosite în domeniul securității cibernetice pentru a identifica și stopa pachetele ce au o adresă de sursă falsă, acțiuni malițioase denumite atacuri de spoofing, ce modifică adresa sursă a unui pachet pentru a da impresia că originea acestuia este de la o sursă cunoscută și de încredere.

În contextul unei aplicații de autentificare folosind algoritmi de recunoaștere facială, tehnica Anti-Spoofing se referă la totalitatea modalităților hardware și software sau hibride hardware-software menite să stopeze logarea unui utilizator ce se folosește de identitatea altui user, apelând fie la o fotografie digitală sau fizică, fie la o înregistrare cu acel user conectat la acea bază de date, în unele cazuri apelându-se și la măști faciale ce replică înfățișarea individului a cărei identitate se fraudează.

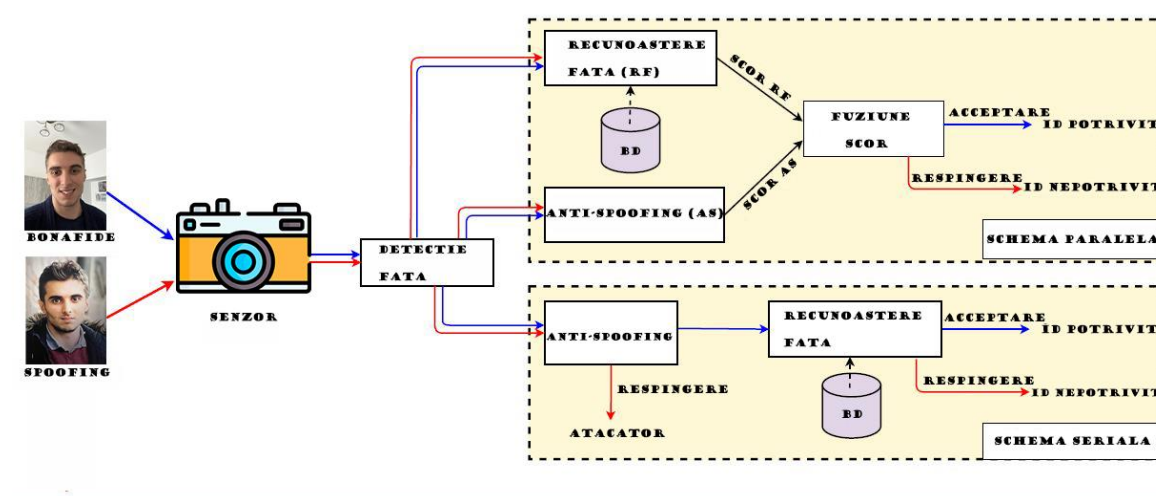


Fig. 3.1: topologia unei aplicații cu sistem anti-spoofing [10]

Se consideră tentativa de fraudare o imagine de tip *spoofing*, iar cea autentică de tip *bonafide*.

Se pot observa în figura de sus abordările posibile pentru o aplicație de autentificare facială, ce include un sistem anti-fraudă: o primă abordare este una paralelă, în care imaginea capturată de senzor, în cazul cel mai comun o cameră web, este trecută prin două căi, simultan făcându-se pe calea de sus recunoașterea facială din baza de date propusă, recunoaștere ce primește un scor RF, iar pe calea de jos se trece prin filtrul anti-spoofing, primindu-se un scor AS. Ulterior, scorurile AS și RF trec printr-un procedeu de fuziune, cel mai dezirabil este să se facă o simplă medie aritmetică, pentru ca mai apoi să se ia o decizie de acceptare sau respingere pe baza acestei valori, pentru a aduce la cunoștință utilizatorului că ID-ul celui autentificat este potrivit sau nu este potrivit. Cea de-a doua abordare este una serială, în care are loc o cascaderă a blocurilor, se trece prima dată prin blocul de anti-spoofing, de unde se ia o decizie timpurie în privința acceptării sau respingerii ID-ului de logare, după care se trece prin blocul de recunoaștere facială relativ la baza de date propusă, ID-ul fiind potrivit dacă și numai dacă s-a trecut cu succes prin filtru și apoi a existat potrivire facială cu un individ din baza de date.

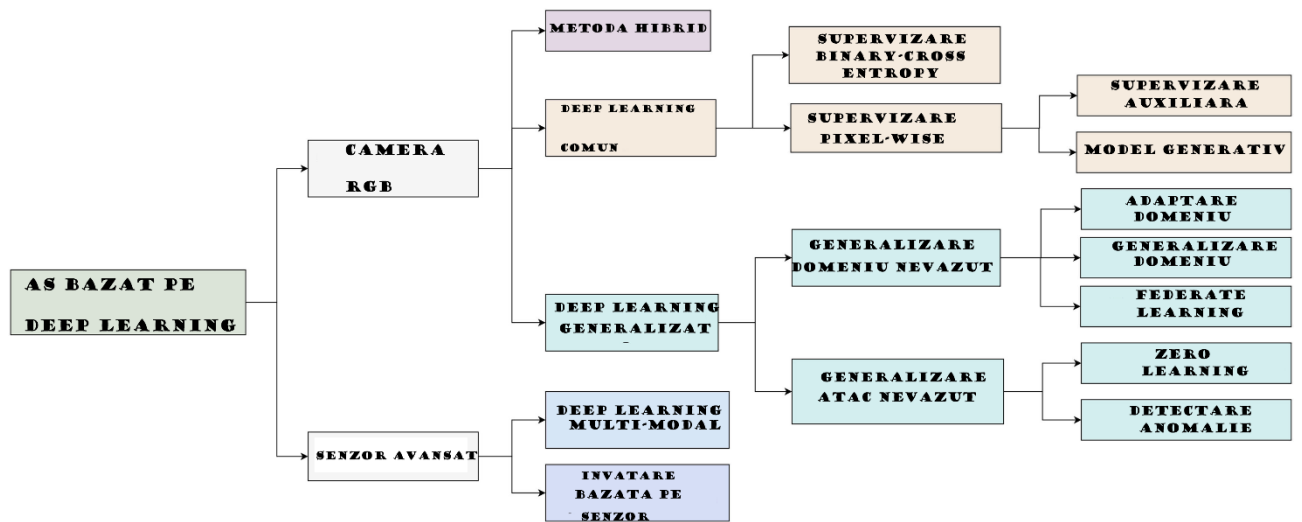


Fig. 3.2: topologie tehnologii Anti-Spoofing [10]

În cadrul unui sistem inteligent, deducem următoarea dihotomie: funcționare bazată strict pe o cameră comercială RGB sau funcționare bazată pe un senzor avansat.

În cazul unei simple camere RGB, se folosește cu precădere Deep Learning-ul comun, dar se aplică și o metodă hibrid. La Deep Learning-ul comun, supervizarea la nivel de pixel pentru sistemul anti-spoofing facial implică analiza detaliată a fiecărui pixel din imagine pentru a detecta eventualele manipulări sau diferențe semnificative între bonafide și spoof. Această abordare oferă o înțelegere mai bună a conținutului imaginii, permițând algoritmilor să identifice indicii subtile asociate cu tentativele de spoofing, cum ar fi textura pielii sau reflexiile neobișnuite.

Supervizarea binary-cross entropy este o metodă de antrenare a modelelor în care se urmărește minimizarea discrepanței între distribuția de probabilitate produsă de model și distribuția reală a datelor, instruindu-se modele pentru a face o distincție cât mai bună, modelul primește o imagine și produce o probabilitate asociată cu fiecare clasă posibilă: autentic sau spoof. Aceste probabilități sunt adesea obținute aplicând funcția sigmoidă, care restrânge rezultatele între 0 și 1, făcându-le interpretabile ca probabilități, apoi se compară cu etichete reale și se calculează pierderea pentru fiecare exemplu.

$$S(x) = \frac{1}{1+e^{-x}}, \text{ funcția sigmoidă}$$

O componentă esențială a deep-learning-ului modal constă în fuziunea și interacțiunea eficientă între datele din diferite modalități, concentrându-se pe integrarea și înțelegerea mai profundă a datelor modalităților multiple, cum ar fi imagini, texte sau sunete. Acest lucru se poate realiza printr-o varietate de tehnici, cum ar fi concatenarea caracteristicilor extrase din fiecare modalitate, aplicarea unor ponderi ori coeficienți pentru fiecare modalitate sau chiar utilizarea unor mecanisme de atenție precum Vision Transformer pentru a scoate în evidență informațiile relevante în fiecare modalitate.

3.2. Principii de funcționare

O metodă propune [11] extragerea de caracteristici LBP și descriptori locali Weber codificați cu CNN, care sunt combinate pentru a păstra informațiile locale privind intensitatea și orientarea marginilor. Cu toate acestea, se pierde detalii la nivel de pixeli, limitând astfel performanța modelului. Pe de altă parte, caracteristicile dinamice precum mișcarea, schimbările de iluminare sau semnalele fiziologice, prezente de-a lungul cadrelor temporale constituie de asemenea intrări CNN eficiente. Feng et al. [12] propun antrenarea unui perceptron multi-strat din mișcările dense bazate pe flux optic, care dezvăluie anomalii în atacurile cu spoof imprimat. Mai mult, Yu et al. [13] construiesc hărți PPG spațio-temporale din videoclipuri cu fețe și folosesc un Vision Transformer pentru a captura caracteristicile periodice ale pulsului cardiac pentru bonafide. Cu toate acestea, mișcările capului și semnalele PPG sunt ușor imitate în atacurile de retransmisie, făcând astfel indiciile dinamice mai puțin fiabile. Plecând de la faptul că atacurile de retransmisie au în mod obișnuit schimbări anormale de reflexie, alți cercetători din cadrul lucrării [14] propun să captureze aceste schimbări de iluminare folosind un CNN 1D cu intrări de histogramme ale diferenței de intensitate din imagini reflectante.

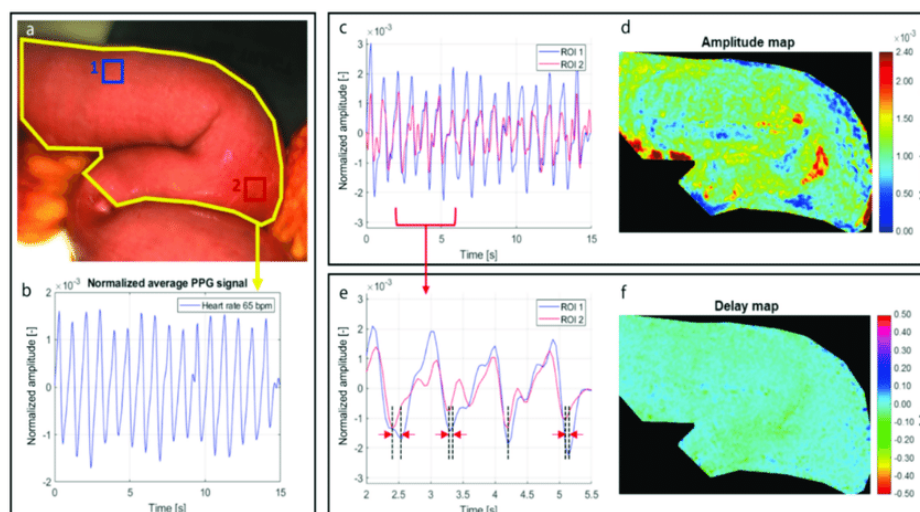


Fig. 3.3: semnal PPG și harta de amplitudine a acestuia pentru o zonă înregistrată a intestinului [15]

O altă metodă implică utilizarea altor dispozitive hardware în afară de senzorul optic de capturare a imaginii userului.

În cadrul unui smartphone precum cele de la Apple, încă din 2017, pentru sistemul de deblocare facială există un ansamblu de componente hardware ce lucrează cu o rețea neuronală adaptată pe un microprocesor precum A11 Bionic: camera frontală menită să captureze poze este însoțită de o cameră cu infraroșu, a cărei funcționare nu este sesizată de utilizator, datorită proprietății ochiului de a acționa drept un "filtru trece jos", lumina infraroșie acționând pe o bandă de lungimi de undă între 780 nm și 1 mm, apoi iluminatorul Flood este un iluminator auxiliar menit să intervină în situații în care lumina ambientală nu este suficientă, iar această intervenție este decisă în funcție de cantitatea de lumină recepționată de senzorul de lumină ambientală, iar cel mai important aspect al acestui sistem inteligent este proiectorul, menit să emită în jur de 30000 de puncte pe fața utilizatorului,

pentru a analiza adâncimea feței, creându-se o hartă a feței respective, ce este stocată sub formă de model matematic în telefon spre a fi comparată cu rezultatele obținute în urma procedurii de scanare a feței aferentă setării inițiale ale acestui mod de deblocare, numit FaceID. Altfel spus, se face o comparație între modelul matematic stocat inițial când s-a făcut implementarea de deblocare prin FaceID și cel obținut în momentul în care se dorește deblocarea telefonului, deci este vorba de un hardware puternic ce are un timp de răspuns de ordinul milisecundelor, cel puțin.

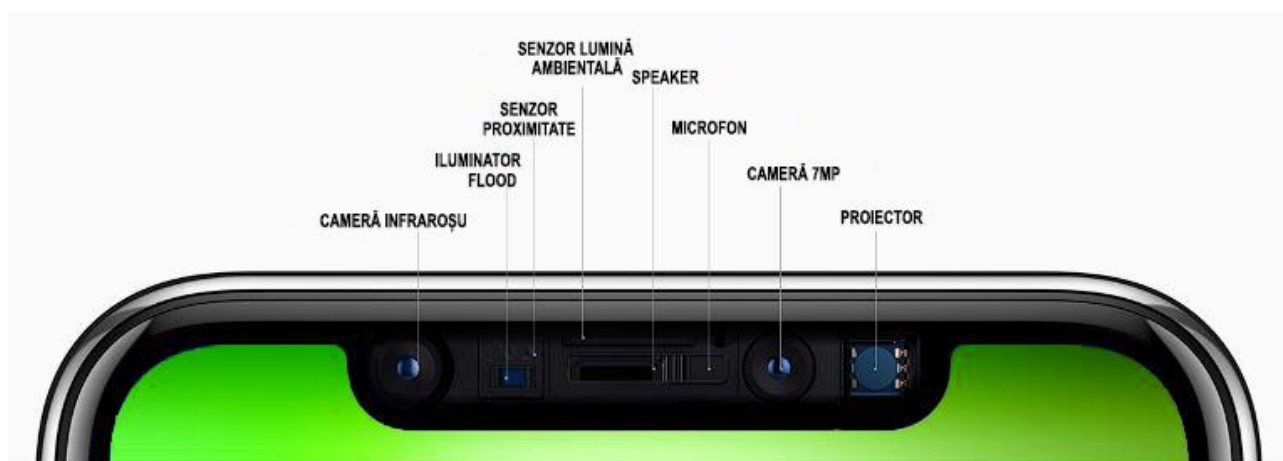


Fig. 3.4: componente sistem FaceID

Senzorul de proximitate este un senzor utilizat în sincron cu senzorul de lumină ambientală pentru a ajuta iluminatorul Flood. Acesta este un senzor ce funcționează pe principiul efectului Doppler și este menit să estimeze distanța dintre telefon și fața utilizatorului.

Efectul Doppler este fenomenul prin care frecvența unei unde, care poate fi sonoră sau luminoasă, este percepută diferit de un observator în mișcare față de sursa acelei unde. Cel mai simplu exemplu din viața de zi cu zi este percepția distinctă a sirenei unei mașini de intervenție de către 2 persoane aflate la distanțe diferite față de mașina respectivă, adică persoana care se află aproape va percepe o undă sonoră cu lungime de undă mică, dar de frecvență mare, iar cealaltă persoană va sesiza o undă sonoră cu lungime de undă mare, însă de frecvență mică. Așadar, estimarea frecvenței unei trimise este cheia funcționării acestui senzor de distanță.

Alternativ, FAS-SGTD este o structură de rețea utilizată pentru detectarea și prevenirea fraudei în sistemele de recunoaștere facială. Aceasta primește o serie de cadre consecutive de imagine și le procesează folosind un bloc special numit RSGB (Residual Spatial Gradient Block) pentru a genera o hartă aproximativă a adâncimii. Apoi, un modul numit STPM (Spatio-Temporal Propagation Module) este folosit pentru a analiza și ajusta adâncimea în funcție de schimbările temporale între cadre, ceea ce îmbunătățește precizia recunoașterii faciale și previne tentativele de fraudare.

Blocurile RSGB folosesc operații matematice pentru extragerea caracteristicilor, operații ce includ convoluții, activări non-liniare (cum ar fi funcțiile de activare ReLU sau sigmoid), normalizare a lotului și, uneori, alte operații aferente rețelelor neuronale, cum ar fi pooling-ul sau concatenarea de canale. Aceste operații matematice sunt aplicate pe regiuni mici ale imaginii pentru a evidenția caracteristici relevante pentru recunoașterea facială, cum ar fi marginile feței, textura pielii sau alte detalii distinctive.

Tehnica Optical Flow se regăsește în modulul STPM, urmărind mișcarea pixelilor între cadrele consecutive, permițând identificarea direcției și vitezei mișcării, iar o altă tehnică ar fi interpolarea temporală. Cu ajutorul acestor tehnici, se obține o pseudo-adâncime. [16]

3.3. State of The Art

Pe baza standardului ISO/IEC DIS 30107- 3:2017 menționat la referința [17], se disting următoarele metrici de bază ale unui sistem Anti-Spoofing:

False Rejection Rate (FRR), ce se referă la intrările respinse în mod eronat și False Acceptance Rate (FAR), ce se referă la intrările acceptate în mod eronat (adică o tentativă de fraudare este percepută drept un utilizator real). De asemenea, se folosesc Half Total Error Rate (HTER), Equal Error Rate (EER), Area Under The Curve (AUC), Attack Presentation Classification Error Rate (APCER), Bonafide Presentation Classification Error Rate (BPCER) și Average Classification Error Rate (ACER).

HTER este o metrică utilizată în evaluarea performanței unui sistem de recunoaștere facială cu sistem anti-spoofing. HTER este definit ca o medie între rata de acceptare a greșelilor (False Acceptance Rate - FAR) și rata de respingere a greșelilor (False Rejection Rate - FRR) la jumătate din rata de probabilitate a deciziei (Decision Rate), iar o valoare mai mică indicând o performanță cât mai bună.

EER este un indicator al echilibrului între securitate (evitarea acceptării greșite a unui spoof) și comoditate (evitarea respingerii greșite a unui bonafide), fiind punctul în care rata de acceptare a greșelilor (FAR) este egală cu rata de respingere a greșelilor (FRR). În general, un EER mai mic este considerat mai bun, deoarece implică un echilibru mai bun al sistemului între aceste două aspecte.

AUC (sau curba ROC) este o reprezentare grafică a performanței unui clasificator binar în funcție de sensibilitate (procentul de exemple pozitive corect clasificate) și specificitate (procentul de exemple negative clasificate corect). AUC este o măsură a cât de bine poate distinge clasificatorul între cele două clase (bonafide versus spoof).

APCER exprimă proporția de tentative de fraudare clasificate greșit ca fiind autentice în comparație cu numărul total de tentative de fraudare, iar cu cât este mai mică valoarea APCER, cu atât sistemul este mai eficient în detectarea și respingerea elementelor spoof.

La polul opus, Bonafide Presentation Classification Error Rate (BPCER), exprimă rata de eroare a clasificării prezentărilor bonafide ca fiind spoof.

Și nu în ultimul rând, ACER se definește ca media aritmetică dintre APCER și BPCER, obținându-se o imagine mai cuprinzătoare a performanței sistemului de recunoaștere facială, iar o valoare mai mică indicând o performanță mai bună a sistemului.

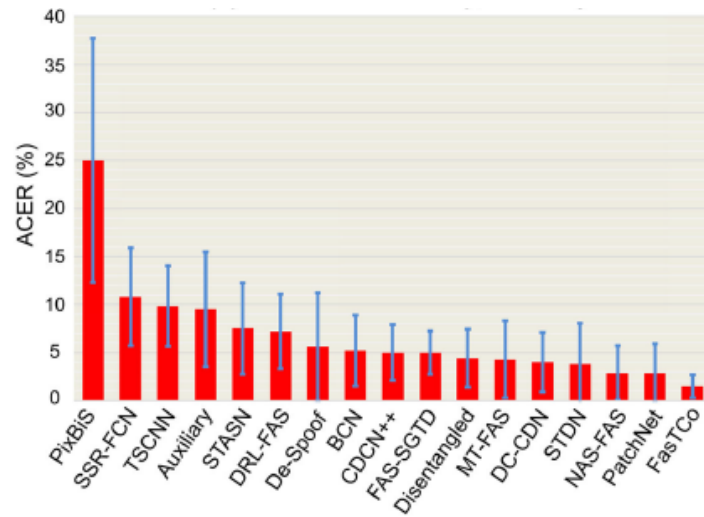


Fig. 3.5: performanțe ACER framework-uri actuale [10]

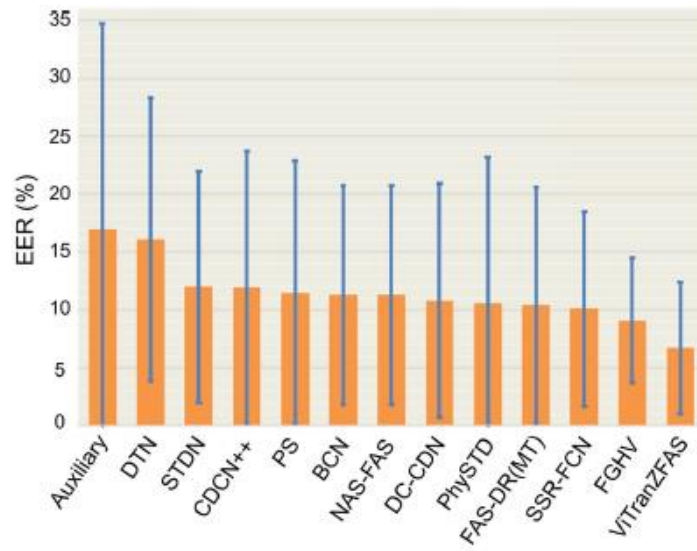


Fig. 3.6: performanțe EER framework-uri actuale [10]

Un dataset de înaltă calitate este crucial pentru un sistem Anti-Spoofing, deoarece oferă cât mai multe informații fundamentale pentru antrenarea și evaluarea precisă a algoritmilor de detectare a încercărilor de fraudare (spoof) din autentificarea pe bază de recunoaștere facială.

Tabelul 3.1: baze de date pentru antrenare Anti-Spoofing [10]

Dataset	An	Bonafide/Spoof	Nr.Subiecți	Tip atac
NUAA	2010	5105/7509(I)	15	imprimat
YALE_Recaptured	2011	640/1920(I)	10	imprimat
CASIA-MFSD	2012	150/450(V)	50	imprimat&replay
REPLAY-ATTACK	2012	200/1000(V)	50	imprimat&replay
Kose and Dugelay	2013	200/198(I)	20	mască din rășină
MSU-MFSD	2014	70/210(V)	35	imprimat&replay
UVAD	2015	808/16268(V)	404	replay
REPLAY-Mobile	2016	390/640(V)	40	imprimat&replay
HKBU-MARs V2	2016	504/504(V)	12	mască din rășină
MSU USSA	2017	1140/9120(I)	1140	imprimat&replay
SMAD	2017	65/65(V)	neștiut	mască din silicon
OULU-NPU	2018	720/2880(V)	55	imprimat&replay
Rose-Youtu	2018	500/2850(V)	20	imprimat, replay&mască
SiW	2019	1320/3300(V)	165	imprimat&replay
WFFD	2019	2300/2300(I)&140/145(V)	745	figuri de ceară
SiW-M	2020	660/968(V)	493	imprimat, replay, mască&machiaj
Swax	2020	1812(I) &110(V)	55	figuri de ceară
CelebA-Spoof	2020	156384/469153(I)	10177	imprimat, replay&mască
RECOD-Mtablet	2020	450/1800(V)	45	imprimat&replay
CASIA-SURF 3DMask	2020	288/864(V)	48	mască
HiFiMask	2021	13650/40950(V)	75	mască

În cea de-a treia coloană, I este abrevierea de la ”image”, iar V este abrevierea de la ”videoclip”, referindu-se la tipul de fișiere pe care le cuprinde fiecare set de date.

După cum se poate observa, baza de date cea mai voluminoasă din prezent este CelebA-Spoof, ce cuprinde o multitudine de fotografii ale persoanelor publice, ce au etichete asociate cu diverse atribute faciale.

SSAN (Spectral Spatial Attention Network) este o tehnică de Anti-Spoofing utilizată încă din 2023, axându-se pe detectarea încercărilor de fraudare folosind un model de rețea neurală ce integrează atenția spectrală și spațială pentru a extrage caracteristici semnificative din imagini și pentru a scoate în relief subtilitățile care fac diferența dintre bonafide și spoof. Toate acestea se datorează unor eforturi de peste 8 ani în cercetare, după cum se poate observa în figura de mai jos.

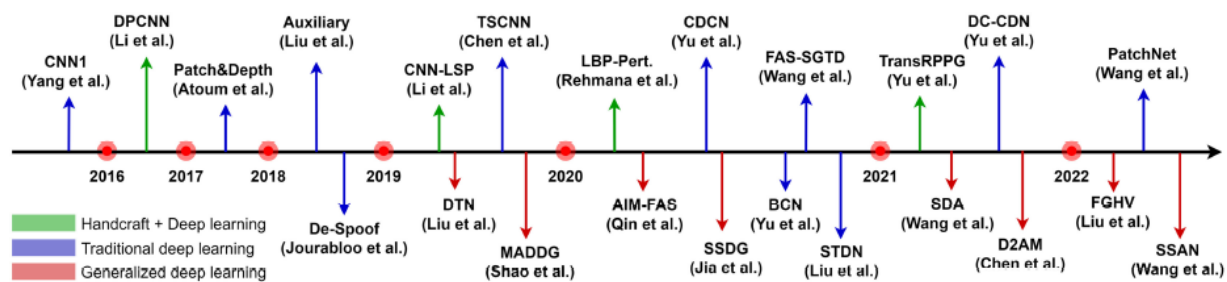


Fig. 3.7: evoluție metode deep-learning utilizate [10]

CAPITOLUL 4: REZULTATE PRACTICE

Pentru lucrarea de față, s-a propus realizarea unei aplicații de autentificare bazată pe algoritmi de recunoaștere facială cu sistem Anti-Spoofing în limbajul de programare Python, aplicație pe care s-au efectuat o serie de experimente. Cheia realizării acestui proiect a constat în utilizarea bibliotecii *face-recognition* și a unor tehnici prezentate în proiectele propuse de către cei de la *cvzone*, ce pot fi regăsite în punctul [18].

Aplicația respectă o topologie serială: se realizează recunoașterea facială, ulterior se trece printr-un filtru ce detectează apariția unui zâmbet, iar în final, se face dihotomia bonafide sau spoof, pe baza căreia se va primi o alertare pe email în cazul detecției unei încercări de înșelăciune.

Diagrama codului poate fi observată în figura 4.1.

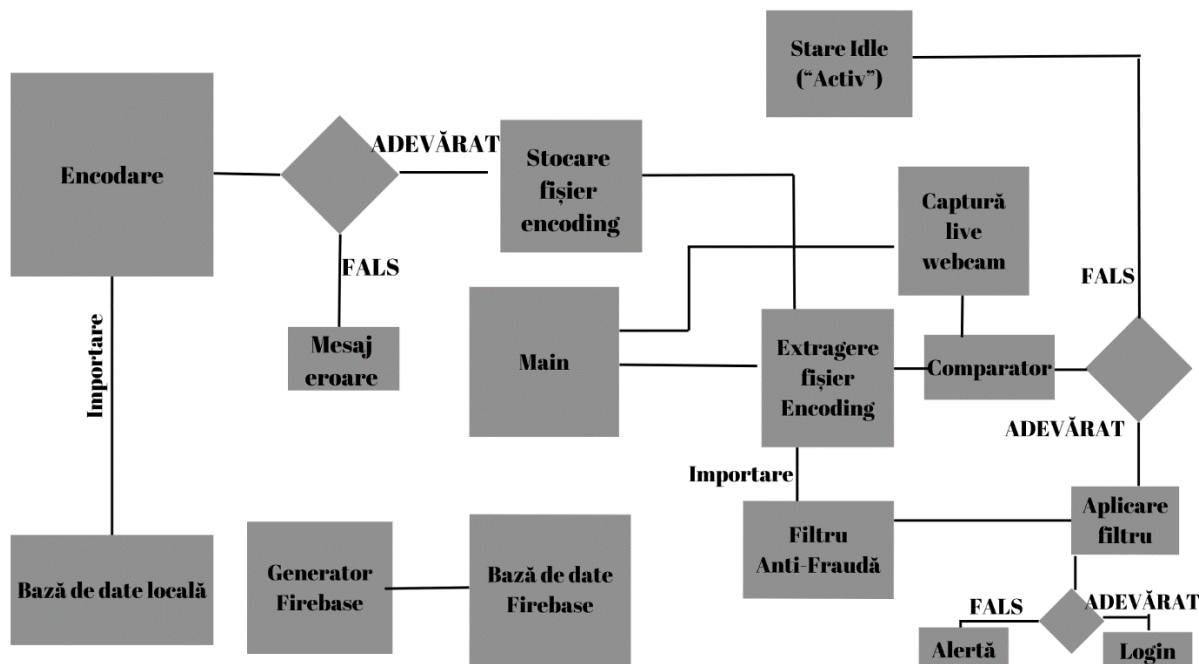


Fig. 4.1: diagramă cod sursă

Codul sursă este realizat prin intermediul unor sumedenii de biblioteci puse la dispoziție programatorilor, care nu necesită o simplă importare, ci au nevoie să fie instalate în IDE-ul programatorului, în cazul nostru, PyCharm Community Edition 2023.1.1, prin comanda *pip install*, după care se inserează numele bibliotecii dorite. Acest proces implică descărcarea pachetului în format *.tar.gz* sau *.whl*, apoi se dezarchivează conținutul și se instalează în directorul implicit de instalare sau cel selectat de către cel care a realizat proiectul, în cazul de față aflându-se în subdirectorul *venv* aferent locației proiectului.

Virtual environment (venv) este o metodă de a crea un mediu Python izolat și independent. Scopul este de a crea o separare față de celelalte proiecte în Python, cât și dependențele și configurațiile care vin cu acestea.

Avantajele utilizării unui mediu virtual sunt următoarele: dependențele sunt izolate, mediul de lucru se poate reproduce prin intermediul unui fișier *requirements.txt*, în care sunt listate toate dependențele proiectului, mediul de lucru este mai organizat, spațiul global de stocare nefiind afectat, iar performanța poate fi mai bună, scurtându-se drumul pentru căutarea resurselor necesare, ele fiind depozitate local, fiind o situație similară cu memoria Cache sau Direct Memory Access, în care sunt degrevate responsabilitățile unității principale de procesare (CPU).

Bibliotecile folosite pentru realizarea acestui software ocupă nu mai puțin de 1.82 GB pe discul calculatorului personal, așa cum se poate observa în figura 4.3.

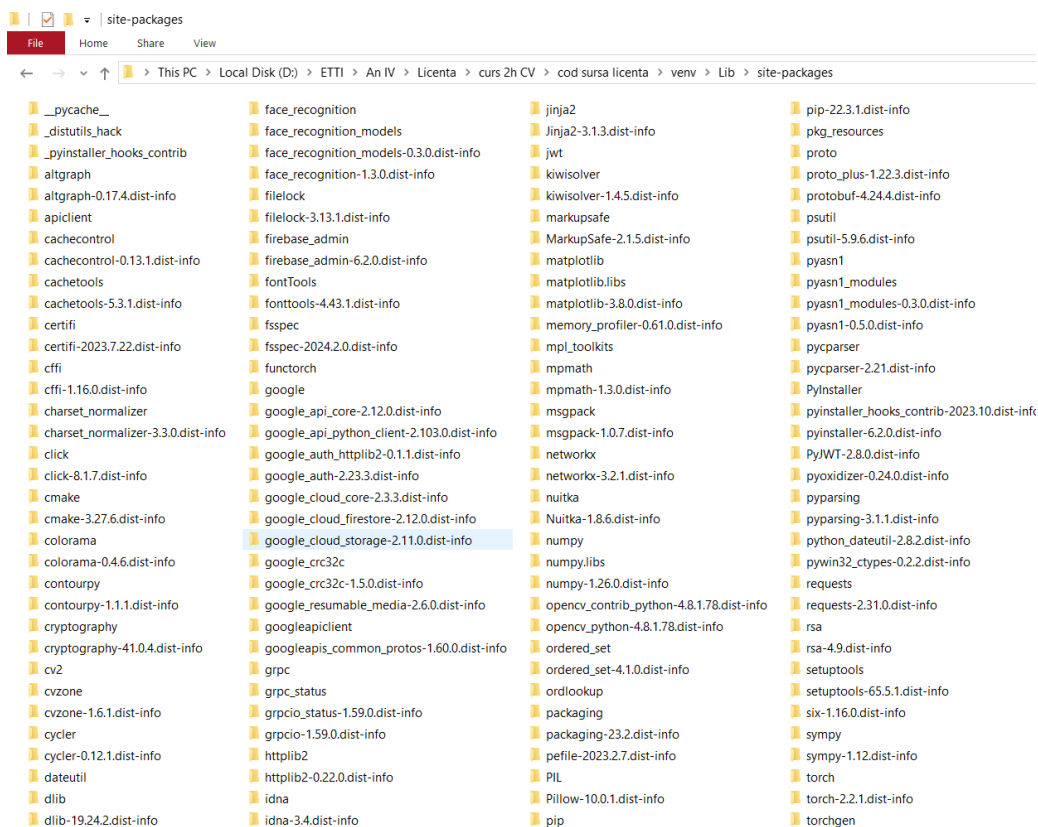


Fig. 4.2: subfoldere prezente în directorul cu bibliotecile

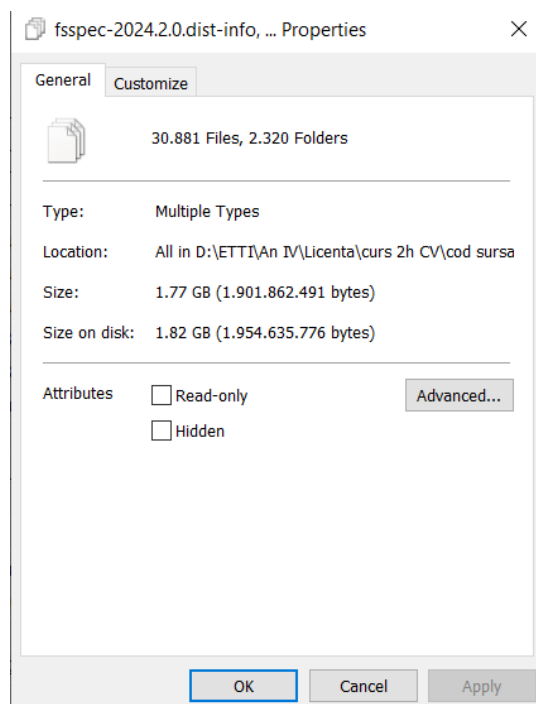
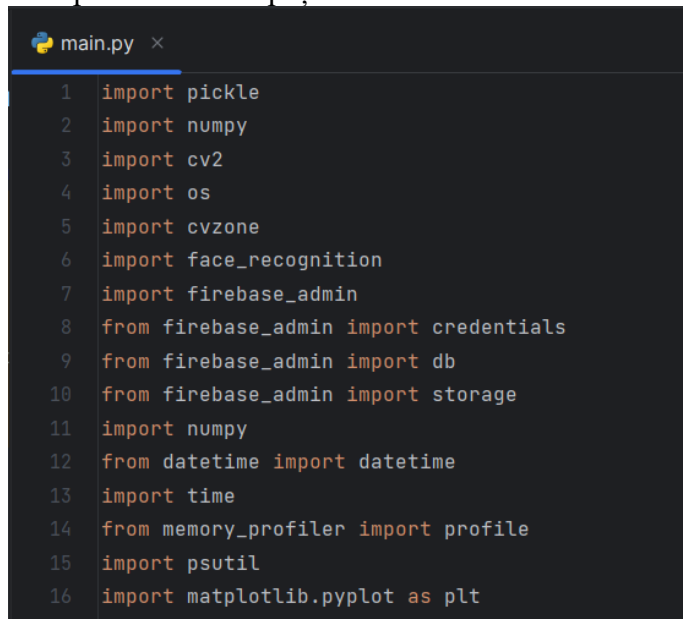


Fig. 4.3: spațiu ocupat de către biblioteci

Proiectul de față este compus din 3 fișiere .py ce asigură aplicația de recunoaștere facială ce detectează zâmbetul, cu interfață și contorizare logare, fiecare având câte un rol crucial: *main.py*, *AddDataToDatabase.py*, *EncodeGenerator.py*, iar cel de-al patrulea fișier este *test.py*, fiind responsabil pentru implementarea filtrului Anti-Spoofing. Pentru fișierul *main*, s-au importat nu mai puțin de 16 biblioteci.



```
main.py x
1 import pickle
2 import numpy
3 import cv2
4 import os
5 import cvzone
6 import face_recognition
7 import firebase_admin
8 from firebase_admin import credentials
9 from firebase_admin import db
10 from firebase_admin import storage
11 import numpy
12 from datetime import datetime
13 import time
14 from memory_profiler import profile
15 import psutil
16 import matplotlib.pyplot as plt
```

Fig. 4.4: biblioteci main.py

Biblioteca pickle este folosită pentru serializarea și deserializarea obiectelor Python, transformându-le într-un format care poate fi salvat pe disc sau trimis prin rețea, pentru ca apoi să fie refăcute în obiecte Python, lucru acesta este util, deoarece se va lucra cu baza de date Firebase.

Biblioteca numpy în Python este fundamentală pentru lucrul cu operații matematice și manipularea datelor regăsite sub formă matriceală, așa cum este în cazul imaginilor cu care se propune lucrul. În cadrul codului alte chestiuni utile constă în găsirea argumentului distanței minime dintre fețe pentru a putea crea variabila *matchIndex*, cât și aflarea valorii de minim a distanței dintre fețe.

Biblioteca cv2 este utilă pentru scrierea și citirea imaginilor, cât și a videoclipurilor, cum este în cazul capturii camerei web.

Biblioteca os asigură interacțiunea cu sistemul de operare, permițând manipularea căilor de fișiere.

Biblioteca cvzone oferă posibilitatea de a trasa forme geometrice peste captura video a camerei web, în cazul nostru un dreptunghi în jurul feței.

Biblioteca face-recognition oferă un set puternic de funcții pentru detectarea și recunoașterea fețelor capturate de camera web, permițând identificarea și localizarea fețelor, extragerea encodărilor acestora și compararea lor pentru recunoaștere.

Gestionarea bazei de date în timp real, stocarea pozelor de profil ale utilizatorilor, cât și ale datelor de logare ale acestora din cadrul aplicației este asigurată de către bibliotecă firebase_admin, cu derivatele acesteia.

Calculul diferenței dintre două date calendaristice pentru a afla timpul scurs de la ultima logare a unui utilizator este posibil cu ajutorul bibliotecilor datetime și time.

Și nu în ultimul rând, pentru a putea înregistra memoria utilizată de către calculatorul personal înainte de execuție, respectiv după execuție, s-au folosit psutil și profile, iar pentru a putea obține un grafic cu bare al acestei evoluții, este la îndemână matplotlib.pyplot.

De asemenea, pentru a putea avea o alertă trimisă pe email, s-a folosit bibliotecă smtplib, în tandem cu clasa EmailMessage.

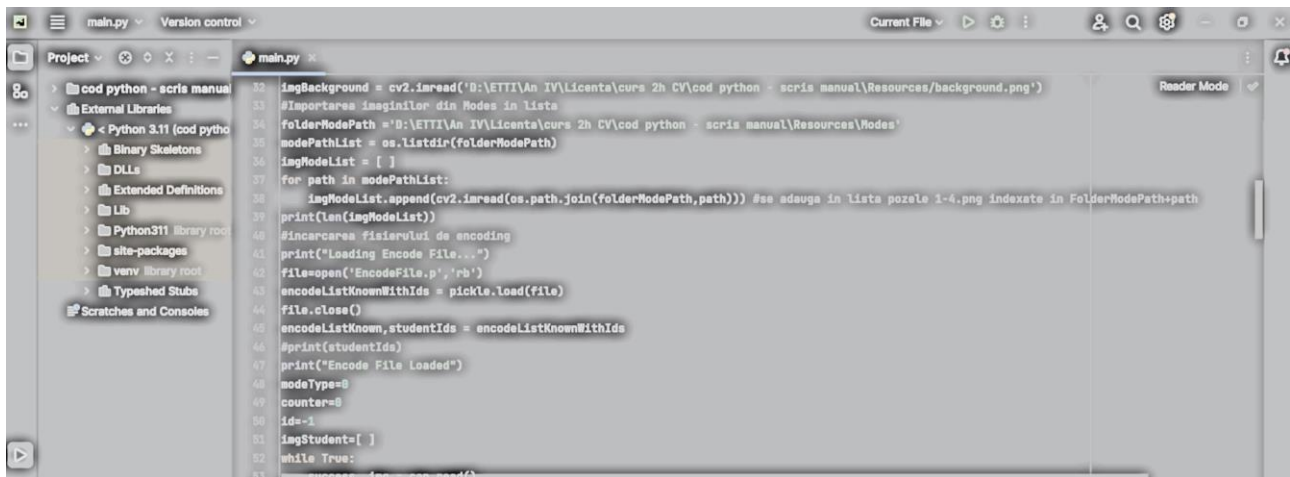


Fig. 4.5: porțiune cod main

Codul aferent fișierului main poate fi privit ca fiind format din 3 segmente principale, fiecare având funcționalitatea sa.

Primul segment începe de la linia 32 și se termină la linia 65, căci primele 32 linii definesc funcția aferentă trimiterii unui email către un destinatar. Rolul acestuia este de inițializare: se alege calea de directoare pentru datele de logare aferente serviciului Firebase, se stabilesc cheia în format *json*, *databaseURL* și *storageBucket*. De asemenea, se inițializează variabila pentru captură video *cap* și se fixează constrângerile legate de rezoluție, lucrând la rezoluția 640x480. Ulterior, se introduc clasificatorii de tip cascadă *face_cascade* și *smile_cascade*, ce folosesc o tehnică de detectare a obiectelor bazată pe aplicarea unui set de filtre pentru identificarea regiunilor de interes, prin prisma a două fișiere XML, conform unor tehnici relatate la referința [19]. Fișierele acestea includ parametri precum dimensiunea ferestrei de glisare, numărul de stadii în cascadă, valorile filtrelor Haar, cât și ale pragului.

Codul XML pentru detectarea unei fețe a fost realizat, prin colaborare cu cei de la Intel, de către Rainer Lienhart, iar cel pentru zâmbet, de către Oscar Deniz Suarez.

Următorul pas corespunzător primului segment este să se încarce elementele constitutive ale interfeței grafice, adică partea principală, ce prezintă o fereastră de tip green-screen, peste care va fi adăugată captura video, iar în partea dreaptă este o fereastră peste care vor fi adăugate alte elemente grafice (poze), numerotate de la 1 la 4, corespunzătoare modulei de lucru al aplicației.

Inițial, la pornire, aplicația este în modul "Activ". În cazul în care un utilizator a fost recunoscut, se face logarea, sunt prezentate datele utilizatorului, după aceea se trece în modul "Verificat", iar dacă se reîncearcă relogarea, adică utilizatorul se află în fața camerei la mai puțin de 20 secunde de logarea precedentă, se trece în modul "Marcat". În final, în acest segment, se extrage fișierul de encodare *EncodeFile.p* și se confirmă încărcarea acestuia cu succes printr-un mesaj transmis în consola IDE-ului, ca mai apoi să fie setate la valori de debut variabile de interes pentru desfășurarea aplicației.

În al doilea segment de cod, de la linia 65 până la linia 108, se citește o imagine de la cameră și se convertește în imagine alb-negru, apoi se detectează fețele din imaginea alb-negru folosind clasificatorul Haar pentru fețe, imaginea color se redimensionează și convertește în format RGB, pentru a se detecta localizările fețelor în aceasta, iar dacă sunt detectate fețe în imagine, se compară caracteristicile fețelor cu cele ale fețelor cunoscute, ulterior, dacă o față cunoscută este detectată, se identifică regiunile de interes printr-o structură repetitivă imbricată, în care avem o sumedenie de variabile de tip tuplu. Tuplurile sunt folosite pentru a stoca și a accesa coordonatele fețelor detectate și ale zâmbetelor într-un mod structurat. De exemplu, *faceLoc* este un tuplu care stochează coordonatele x și y ale colțurilor unei fețe detectate, iar *smile* stochează coordonatele zâmbetului în interiorul feței detectate. În final, variabila *bbox* este un tuplu care stochează coordonatele și dimensiunile unui chenar care înconjoară fețele detectate, însă înainte de a înconjura fața subiectului din fața webcam-ului cu acest chenar, se evaluează valoarea variabilei *label*, integrate sistemului

Anti-Spoofing cuprins. Dacă aceasta are valoarea 1, atunci se consideră subiectul bonafide, dacă nu, se consideră spoof, drept urmare, urmează o alertă.

În al treilea segment de cod, având variabila *counter* în prim-plan, prin verificarea dacă aceasta este diferită de zero, confirmăm dacă o față cunoscută a fost detectată și identificată. Dacă valoarea este 1, se preiau informațiile despre individ din baza de date Firebase folosind un identificator de tip id. Se obține și imaginea asociată cu utilizatorul identificat din bucket-ul Firebase, apoi se transformă imaginea descărcată într-un vector de numere întregi fără semn. Imaginea se decodează din formatul aferent Firebase (BGRA), în formatul folosit de OpenCV (BGR), iar apoi se convertește șirul de dată și oră din baza de date Firebase într-un obiect datetime Python, pentru a putea calcula ulterior numărul de secunde trecute de la ultima prezență a utilizatorului.

În Firebase, numărul total de prezențe ale utilizatorului este actualizat în momentul în care au trecut mai mult de 30 secunde de la ultima logare a acestuia, cât și data și ora ultimei logări.

Se verifică dacă variabila *counter* este între 10 și 20 și se setează *modeType* la 2, pentru a trece la noua stare în fereastra grafică. Pentru a urmări numărul de cadre procesate de la detectarea unei fețe cunoscute, se face o incrementare a acestei variabile. Dacă s-a depășit valoarea 20, înseamnă că s-au procesat suficiente cadre pentru a finaliza acțiunea și a pregăti aplicația pentru detectarea următoarei fețe, caz în care se resetează variabilele de inițializare aferente primului segment de cod.

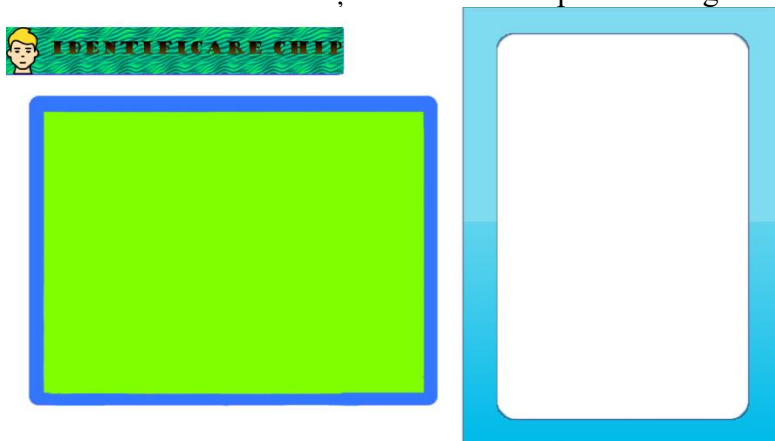


Fig. 4.6: fereastra principală GUI

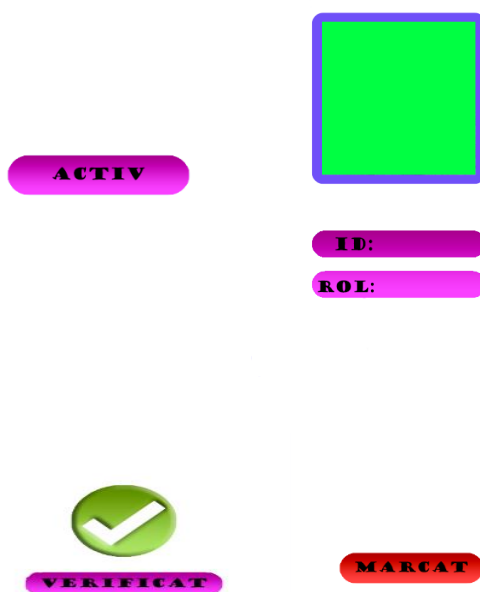


Fig. 4.7: moduri aplicație

În fișierul *EncodeGenerator.py*, imaginile sunt încărcate în directorul specific, iar ID-urile asociate cu acestea, necesare pentru popularea bazei de date Firebase, sunt extrase din numele fișierelor. Aceste fișiere trebuie să fie în format .png și să aibă rezoluția 216x216 pentru a asigura o encodare cu succes. Apoi, funcția *findEncodings* este definită pentru a encoda fețele din imaginile încărcate, lucru realizat utilizând biblioteca *face_recognition*, care extrage caracteristicile feței și le encodează într-un fișier cu extensia .p folosind modulul *pickle*.

De asemenea, se înregistrează parametri de performanță precum memoria înainte de execuție, prin amplasarea unei sintaxe specifice bibliotecilor *psutil* și *memory_profiler* la începutul codului, memoria după execuție, prin amplasarea altei sintaxe specifice bibliotecilor *psutil* și *memory_profiler*, iar nu în ultimul rând, se deduce timpul scurs prin aflarea diferenței dintre timpul marcat la sfârșitul codului și timpul marcat la începutul codului. Cu acești parametri de performanță se vor realiza, tot în cadrul acestui cod, pe lângă mesaje în consolă, grafice de tip bară, unul ce evidențiază evoluția memoriei executate, iar celălalt, evoluția timpului scurs. [20]

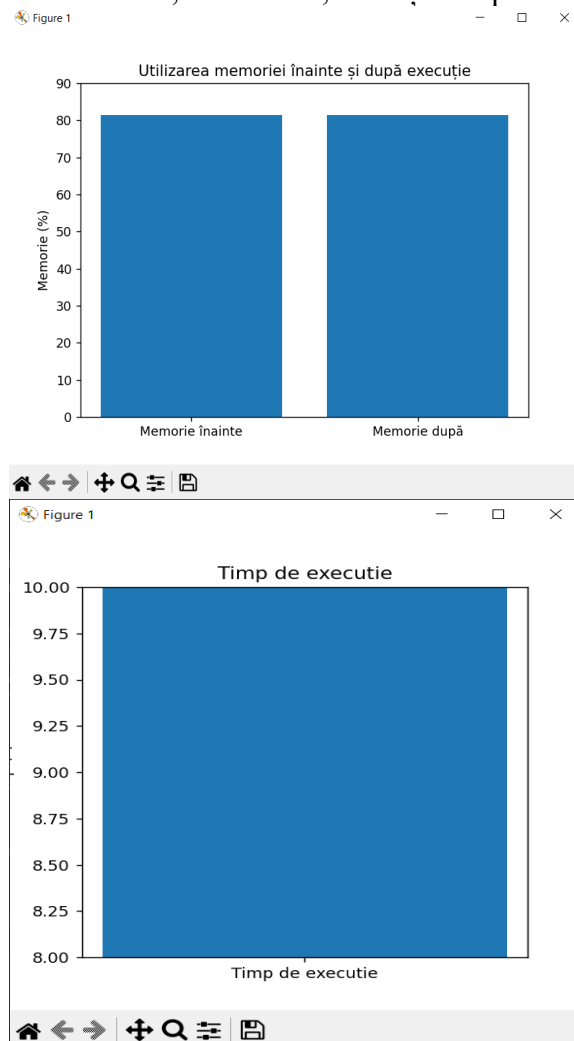


Fig. 4.8: grafic evoluție memorie utilizată

```
Timpul de execuție: 16.54 secunde
Memorie înainte de execuție: 81.40%
Memorie după execuție: 81.40%
```

Fig. 4.9: exemplu afișaj consolă

AddDataToDatabase se ocupă de integrarea mediului Python cu Firebase, inițializându-se conexiunea folosind fișierul JSON de autentificare furnizat, ca la celelalte fișiere componente aplicației. Aici sunt structurate datele despre utilizatori printr-un dicționar Python, cheile

dicționarului sunt ID-urile utilizatorilor, iar valorile reprezintă alte dicționare ce conțin date specifice fiecărui utilizator: numele și prenumele, domeniul în care activează, anul de începere a activității pe care o prestează, numărul de logări în aplicație, un scor privind poziția sa ierarhică în instituția în care activează, numărul de ani de când prestează activitatea, cât și data ultimei logări.

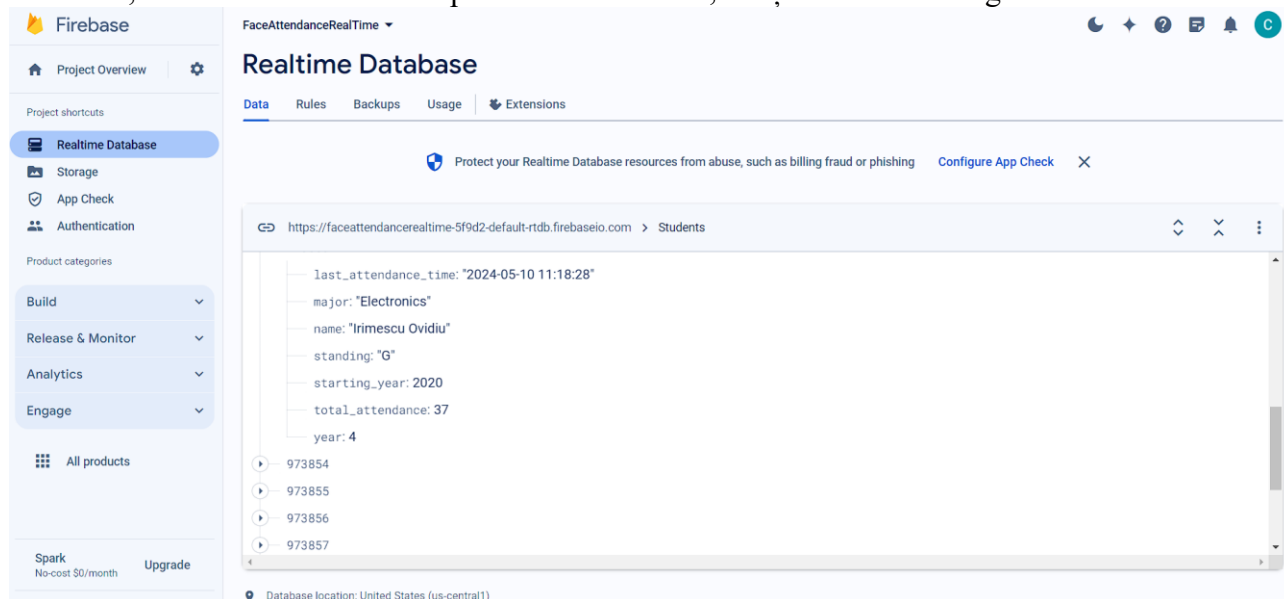


Fig. 4.10: ilustrare bază de date FireBase

Pentru a avea un filtru Anti-Spoofing, s-a efectuat o importare a unui set de algoritmi dezvoltati de către compania Minivision Technology. Acest filtru se bazează pe rețeaua neuronală MiniFASNet, care funcționează pe principiul transformatei Fourier. Spectrul transformatei Fourier poate discerne diferența dintre bonafide și spoof în domeniul frecvenței până la anumite limitări, deci arhitectura acestui model constă în supervizare auxiliară. Codul sursă pentru acest sistem anti-fraudă se află la referința [21].

Pentru a face această importare, având codul main deschis, în IDE s-a mers la File, s-a selectat Settings, apoi Project Structure, de unde se selectează folderul aferent algoritmilor Anti-Spoofing, folder ce trebuie să se regăsească în același director cu cel al proiectului principal, în care este implementată aplicația de autentificare facială. După ce s-a selectat acest director, la câmpul *Mark as*, selectăm Sources, după aceea se confirmă modificările prin apăsarea comenzii OK.

Ulterior, în codul main.py se aplică la începutul acestuia comanda *from test import test*, pentru a realiza conexiunea cu succes a acestui filtru.

Cheia funcționării filtrului constă în variabila *label* regăsită în codul din *main.py*. Aceasta este inițializată prin apelarea funcției *test*, în care se specifică 3 parametri: *image*, în care se selectează captura de webcam, *model_dir*, în care se specifică folderul cu resursele necesare pentru accesarea rețelei neuronale cu modelele de antrenare inerente și nu în ultimul rând, *device_id*, ce este setat la 0, referindu-se la unitatea grafică de procesare pe care dorim să o utilizăm.

This PC > Local Disk (D:) > ETTI > An IV > Licenta > curs 2h CV > cod python - scris manual > Silent-Face-Anti-Spoofing-master > resources > anti_spoof_models				
Name	Date modified	Type	Size	
2.7_80x80_MiniFASNetV2.pth	21.12.2022 16:08	PTH File	1.807 KB	
4.0_0_80x80_MiniFASNetV1SE.pth	21.12.2022 16:08	PTH File	1.813 KB	

Fig. 4.11: resursele necesare filtrului Anti-Spoofing

Aplicația finală permite așadar utilizatorului să se logheze, iar datele acestuia, numele și prenumele, ID-ul și rolul în companie, să fie afișate dacă și numai dacă persoana din fața webcam-ului este autentică și în același timp, zâmbește.

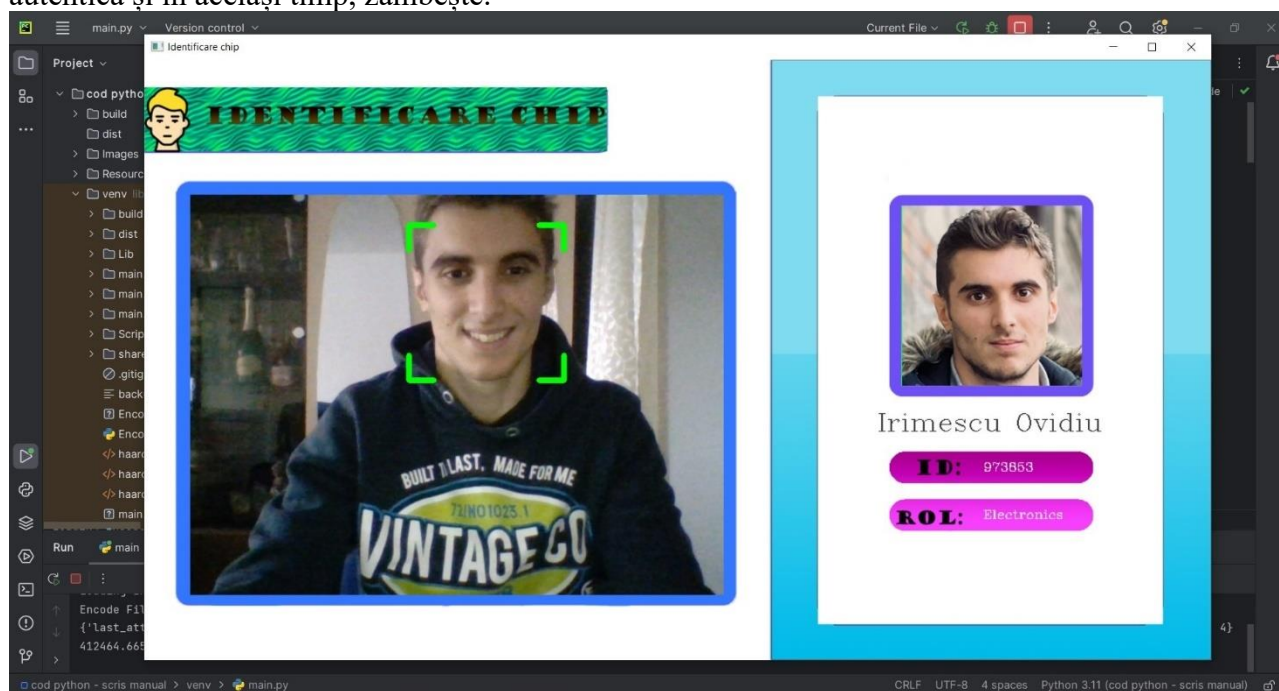


Fig. 4.12: rezultat practic aplicație cu detecție zâmbet

Pentru această aplicație s-au efectuat două teste principale: primul evidențiază persistența recunoașterii faciale cu trecerea timpului, iar cel de-al doilea ilustrează evoluția coeficientului decident pe parcursul anilor.

Vom numi primul test "analiza 1", iar cel de-al doilea "analiza 2". În cadrul analizei 1, se află un platou de recunoaștere pentru 8 subiecți. Pentru fiecare subiect s-au ales poze ce ilustrează înfățișarea individului de-a lungul timpului, având ca poze de referință capturi ale momentelor din tinerețe ale utilizatorilor propuși. Aceste poze de referință sunt cele prezente în directorul aferent codului responsabil pentru encodare, fiind în format png și obligatoriu de rezoluție 216x216. Pentru a face acest test posibil, s-a folosit baza de date *IMDB-WIKI*, motiv pentru care a fost necesar să se dezactiveze atât filtrul Anti-Spoofing, cât și cel de detecție zâmbet. În cazul în care nu s-ar fi făcut aceste modificări codului, toate încercările de a folosi poze cu persoane care nu se află la fața locului ar fi fost semnalate drept spoof, așa cum este și intenția programului final. De asemenea, baza de date utilizată cuprinde o colecție de poze ale persoanelor celebre ce activează în industria cinematografică, acestea nefiind de multe ori surprinse în ipostaza de a zâmbi, de aceea, s-a renunțat și la cel de-al doilea filtru. Baza de date este alcătuită din 10 subdirectoare ce ocupă în total în jur de 25.7 GB. Aproape fiecare poză cuprinde în denumirea sa anul de naștere al subiectului pozat și anul în care a fost pozat, iar în descrierea fișierului pozei este specificat numele subiectului.

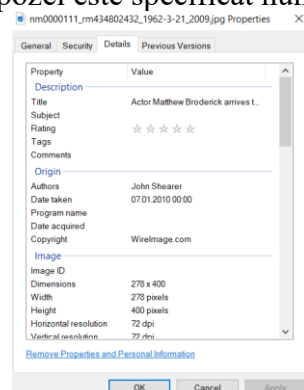


Fig 4.13: detalii fișier poză

Așadar, pentru consemnarea vârstei subiectului s-a efectuat diferența dintre anul în care a fost pozat subiectul și anul nașterii, ambele cuprinse în descriere. Trebuie specificat totuși că există neconcordanțe, datele nefiind pe deplin precise, lucru ce s-a putut observa pentru un subiect în urma comparării cu niște fotografii prezentate pe enciclopedia online Wikipedia. Într-o foaie de calcul Excel s-au notat sub forma unui tabel de 11 linii și 16 coloane vârsta și rezultatul corespunzător fiecărui subiect. Pentru a putea vizualiza mult mai facil și ordonat aceste date, s-a împărțit acest tabel în 2 tabele mai mici, ce prezintă informații pentru câte 4 subiecți. Coloana ”Rezultat” se interpretează astfel, pentru valoarea 1 se înțelege faptul că recunoașterea a avut loc, a fost posibilă într-un timp de expunere mai mic decât 10 secunde, iar valoarea 0 indică în majoritatea cazurilor faptul că algoritmul de recunoaștere facială nu mai identifică subiectul, iar în puține cazuri, această recunoaștere a avut loc la un timp de expunere mai mare decât 10 secunde.

Tabel 4.1: prima serie de subiecți

Subiect 1		Subiect 2		Subiect 3		Subiect 4	
Vârsta	Rezultat	Vârsta	Rezultat	Vârsta	Rezultat	Vârsta	Rezultat
21	1	28	1	30	1	23	1
26	0	31	1	35	1	26	0
26	1	35	1	37	1	29	1
27	1	42	1	40	1	33	1
30	1	43	1	41	1	34	1
31	1	45	1	42	1	46	0
32	1	46	1	52	1		
35	1	53	1				
42	0	57	1				

Tabel 4.2: a doua serie de subiecți

Subiect 5		Subiect 6		Subiect 7		Subiect 8	
Vârsta	Rezultat	Vârsta	Rezultat	Vârsta	Rezultat	Vârsta	Rezultat
29	1	32	1	22	1	33	1
32	1	33	1	27	1	42	1
34	1	36	1	30	1	49	0
36	1	48	1	33	1	57	1
39	1	55	1	35	1	60	1
42	1			43	1	62	1
53	1			48	0	64	1
						73	0

Procedura de captare a datelor a fost următoarea: s-au ordonat din baza de date poze cu fiecare persoană în parte, obținându-se câte un dosar pentru fiecare individ, iar aceste dosare au fost încărcate pe Google Drive, pentru a putea fi accesate de pe telefon. Pozele sunt deschise pe rând pe telefon, iar în fața webcam-ului se apropie dispozitivul mobil, astfel încât fața individului ce trebuie recunoscut să fie luată în prim-plan spre analiză.

Calitatea unora din poze nu este mereu foarte bună, unele poze prezintă o oarecare încețoșare, un nivel de blur, care, într-adevăr, poate afecta recunoașterea, motiv pentru care rezultatele nu pot fi considerate ca având o acuratețe de 100%. De asemenea, comparativ cu o testare efectuată cu foi imprimare, pot apărea reflexii ale ecranului dispozitivului mobil, cauzate de lumina solară, ce estompează vizibilitatea și captura camerei.

Din aceste tabele, s-au trasat grafice ce ilustrează pe axa Ox vârsta, iar pe axa Oy rezultatul, dacă recunoașterea are sau nu are loc. Forma graficului poate fi asemănată cu a unui platou.

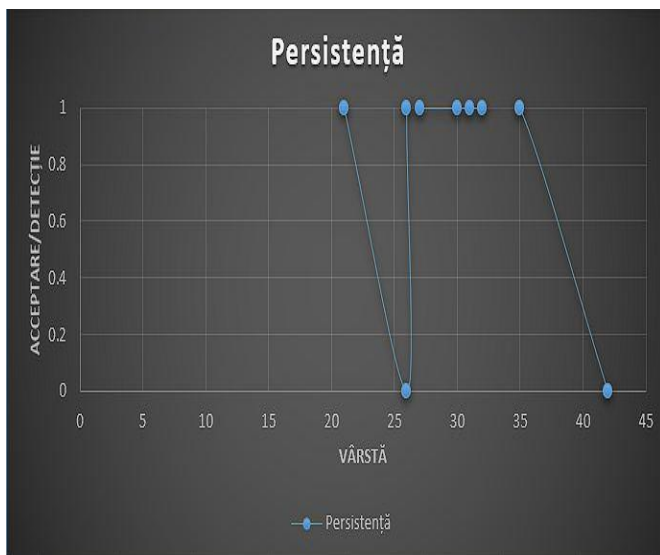


Fig. 4.14: Platou persistență subiect 1

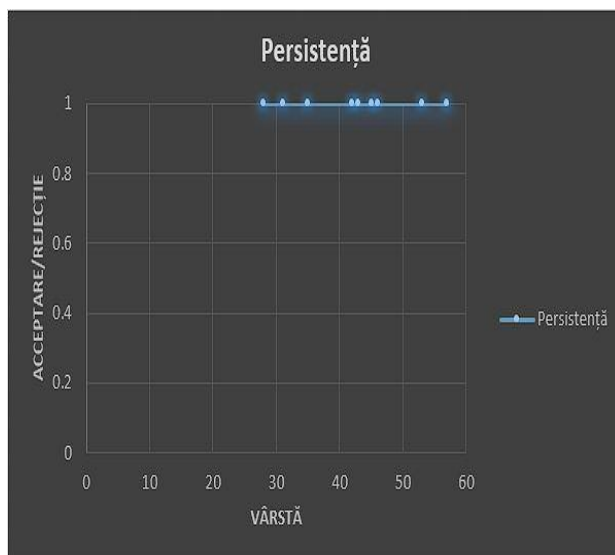


Fig. 4.15: Platou persistență subiect 2

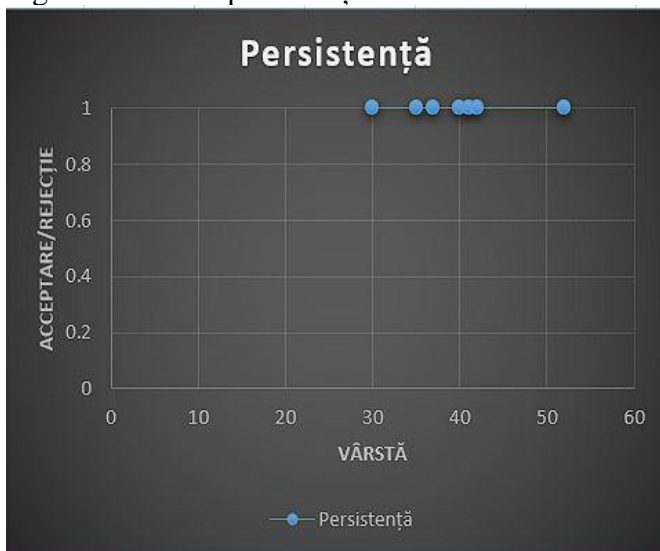


Fig. 4.16: Platou persistență subiect 3

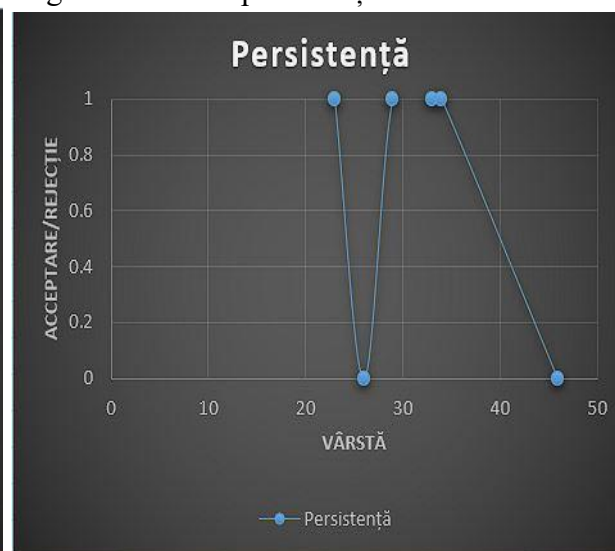


Fig. 4.17: Platou persistență subiect 4

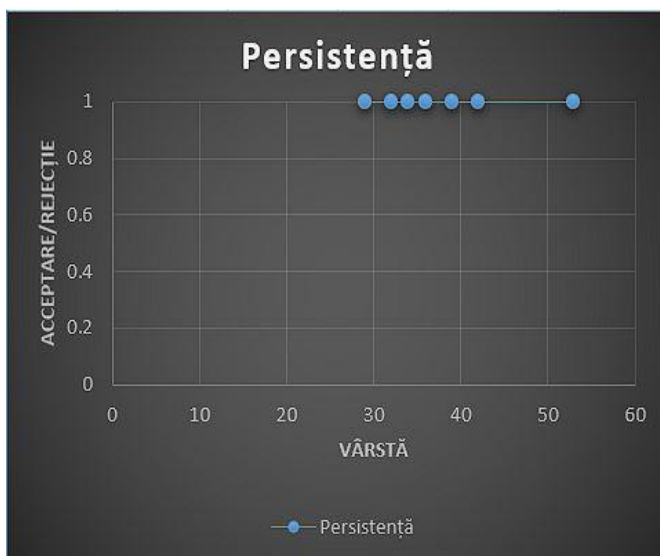


Fig. 4.18: Platou persistență subiect 5

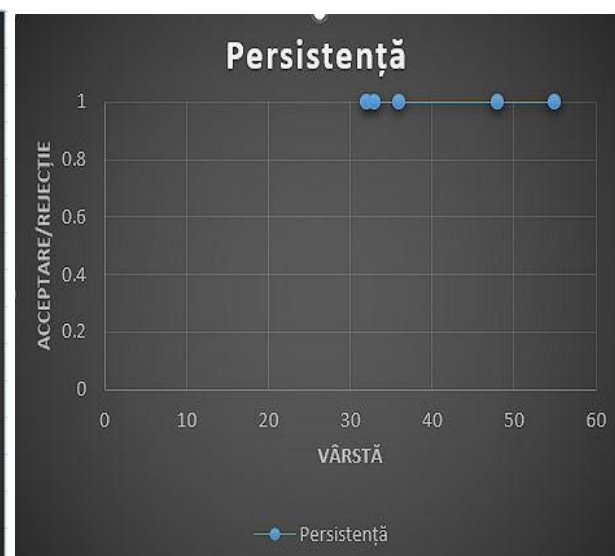


Fig. 4.19: Platou persistență subiect 6

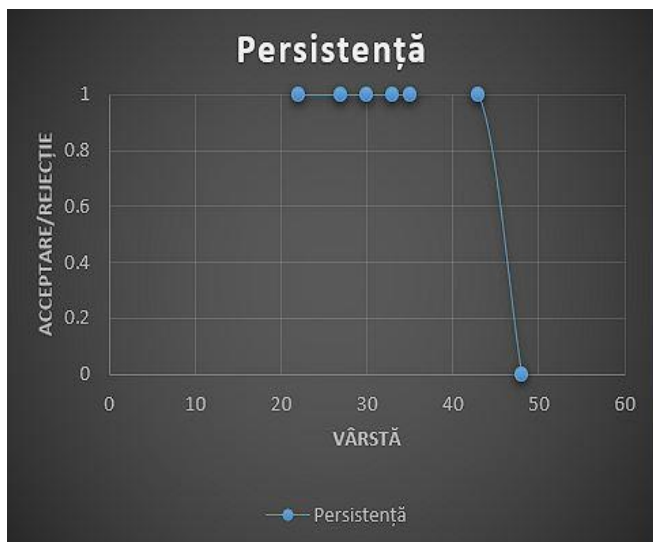


Fig. 4.20: Platou persistență subiect 7

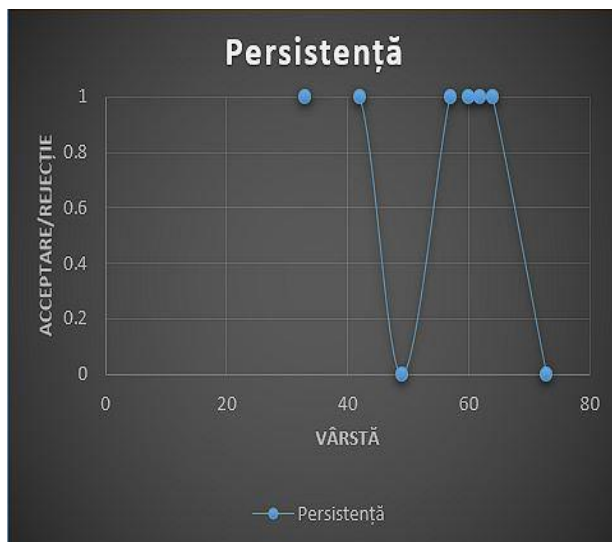


Fig. 4.21: Platou persistență subiect 8

Făcându-se diferența dintre vârsta de la care s-a oprit recunoașterea, din punctul din extrema dreaptă și vârsta de la care s-a început efectuarea recunoașterii, obținem durata platoului persistenței vârstei de recunoaștere facială, pentru care am efectuat un alt grafic de tip bară.

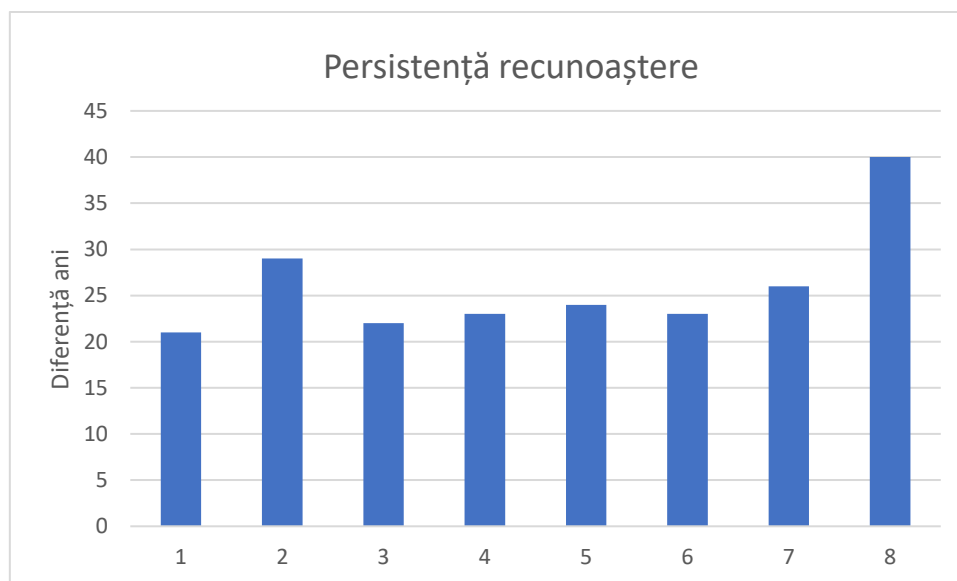


Fig. 4.22: Durate persistențe recunoaștere

Din figura de mai sus putem deduce că cea mai îndelungată persistență a avut loc pentru subiectul numărul 8, acesta având o recunoaștere pe o perioadă de 40 de ani, cu un hazard ce a avut loc la vârsta de 49 de ani, subiectul schimbându-și aparența prin abordarea unui stil cu păr facial, ceea ce a făcut recunoașterea să nu aibă loc. Efectuând media aritmetică a valorilor diferențelor de ani, obținem valoarea medie aferentă recunoașterii: 26 de ani.

În cadrul analizei 2, s-a dedus variația valorii coeficientului decident cu trecerea timpului pentru aceiași 8 subiecți. Coeficientul decident este o variabilă ce ia valori între 0 și 1, reprezentând procentul de potrivire a feței individului cu fața din poza de referință, prezentă în etapa de fetching sau encodare. Așadar, pentru următoarele 2 tabele, configurate asemănător cu cele de mai sus, în coloana "Rezultat", o valoare cât mai apropiată de 1 indică o potrivire cât mai bună.

Tabel 4.3:

Subiect 1		Subiect 2		Subiect 3		Subiect 4	
Vârsta	Rezultat	Vârsta	Rezultat	Vârsta	Rezultat	Vârsta	Rezultat
21	0.67	28	0.46	30	0.71	23	0.51
26	0.49	31	0.57	35	0.5	26	0.43
26	0.468	35	0.46	37	0.59	29	0.46
27	0.505	42	0.56	40	0.56	33	0.44
30	0.492	43	0.48	41	0.55	34	0.47
31	0.516	45	0.58	42	0.51	46	0.4
32	0.464	46	0.62	52	0.58		
35	0.443	53	0.55				
42	0.402	57	0.58				

Tabel 4.4:

Subiect 5		Subiect 6		Subiect 7		Subiect 8	
Vârsta	Rezultat	Vârsta	Rezultat	Vârsta	Rezultat	Vârsta	Rezultat
29	0.73	32	0.51	22	0.55	33	0.64
32	0.52	33	0.4	27	0.45	42	0.45
34	0.48	36	0.46	30	0.42	49	0.41
36	0.44	48	0.44	33	0.44	57	0.46
39	0.57	55	0.42	35	0.43	60	0.49
42	0.43			43	0.4	62	0.44
53	0.4			48	0.47	64	0.4
						73	0.43

Efectuând în aceeași manieră pentru tabelele 4.1 și 4.2, obținem forme de unde ce intuitiv ar fi respectat o monotonie descrescătoare, însă rezultatele sunt variate.

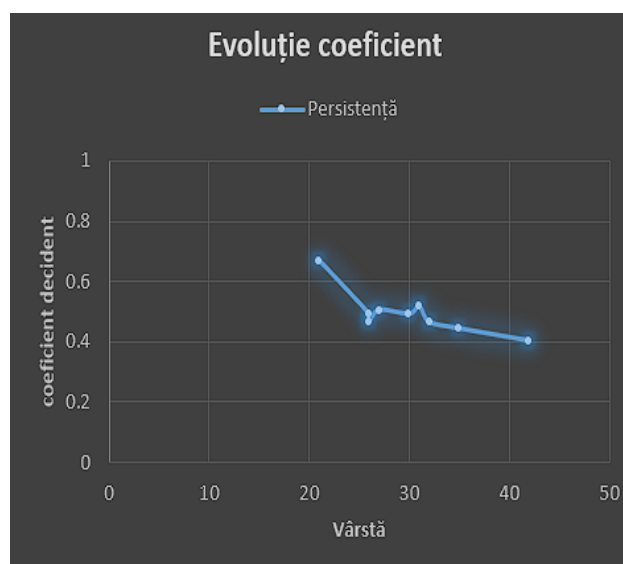


Fig. 4.23: evoluție coeficient subiect 1

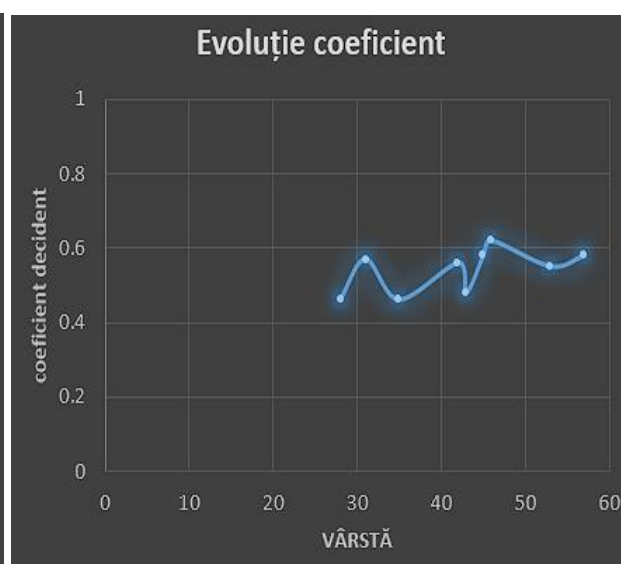


Fig. 4.24: evoluție coeficient subiect 2

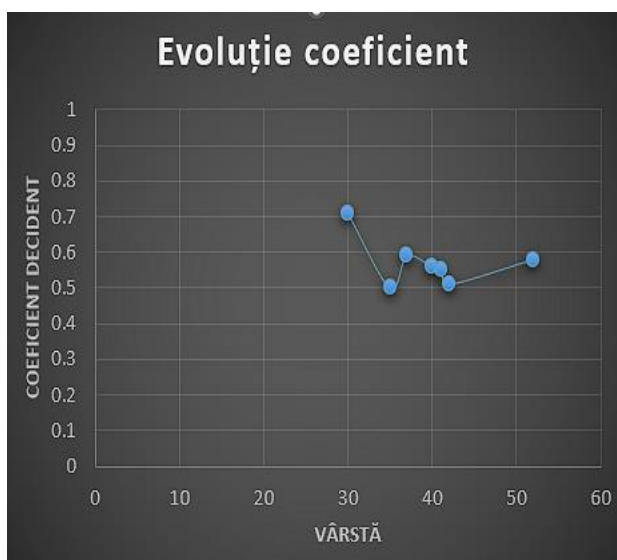


Fig. 4.25: evoluție coeficient subiect 3

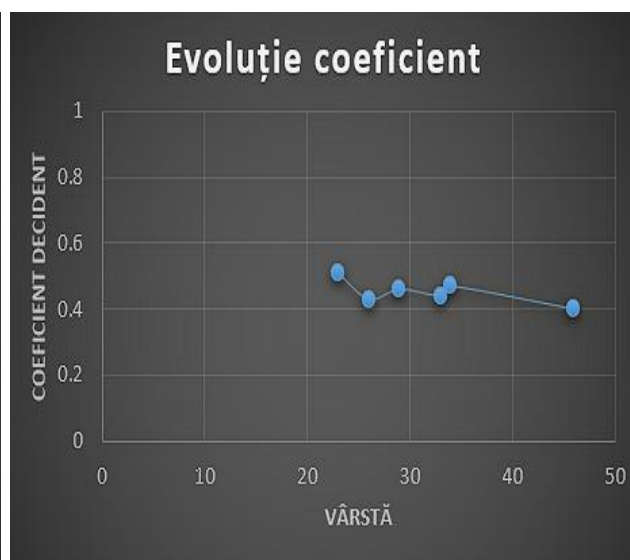


Fig. 4.26: evoluție coeficient subiect 4

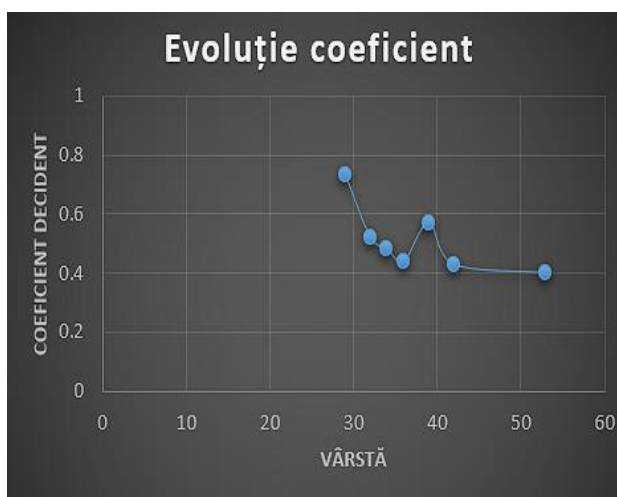


Fig. 4.27: evoluție coeficient subiect 5

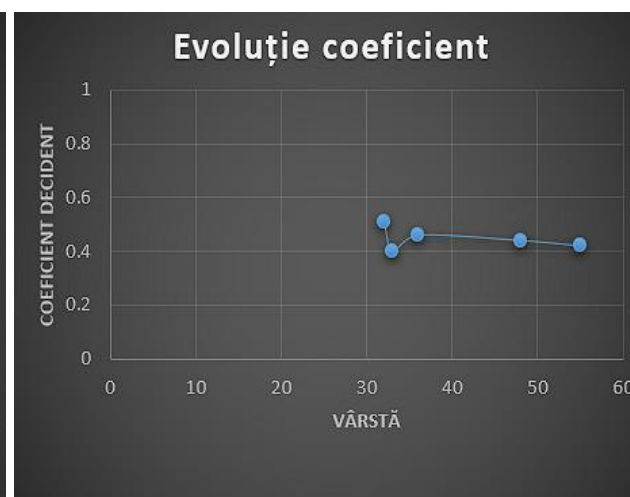


Fig. 4.28: evoluție coeficient subiect 6

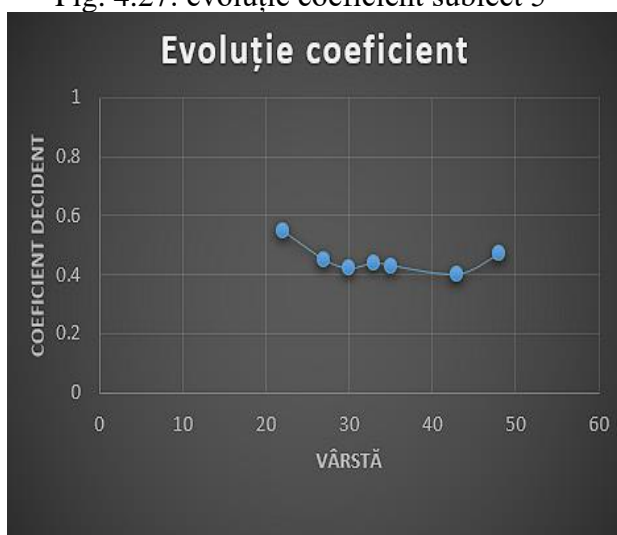


Fig. 4.29: evoluție coeficient subiect 7

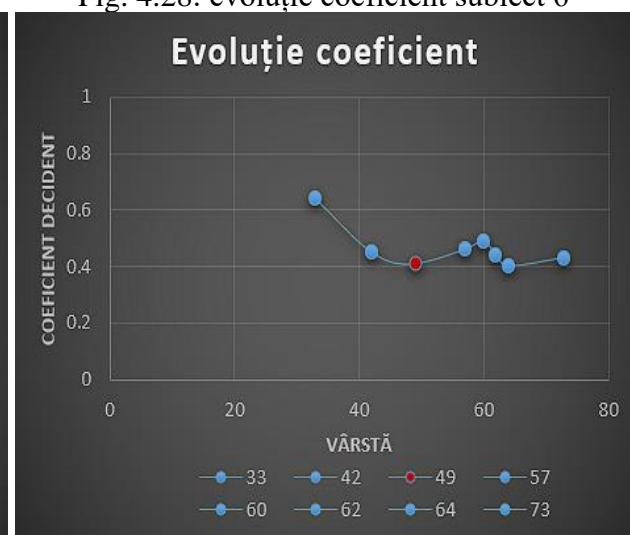


Fig. 4.30: evoluție coeficient subiect 8

În cadrul analizei efectuate pentru subiectul 8 (Bruce Boxleitner), a avut loc o anomalie: la vârsta de 49 de ani a avut loc o confuzie cu subiectul 6 (Kenneth Branagh), obținându-se valoarea

coeficientului de 0.41, de aceea, s-a marcat pe grafic (figura 4.30) cu roșu punctul corespunzător vârstei la care a apărut confuzia.

După cum se poate observa, trendul evoluției este descendent, existând în cadrul fiecărei figuri cel puțin două pante negative, ba chiar și 4 astfel de pante, cum se poate observa la al cincelea subiect, însă nu este o funcție strict monotonă, existând pe alocuri și creșteri ale coeficientului cu până la 20%. Cea mai mare creștere a coeficientului a fost de 33% și a fost găsită la subiectul 5, raportându-se la vârsta de referință de 29 de ani și vârsta de 53 ani, așadar pe o perioadă de 24 de ani, această valoare a scăzut cu 0.33, de la 0.73 la 0.4.

O altă analiză pe care am efectuat-o a constat în aflarea evoluției timpului de encodare pe măsură ce baza de date a aplicației de recunoaștere a fost populată cu utilizatori, implicit cu cât mai multe poze. De asemenea, s-a notat și procentul de memorie rulată după efectuarea acestui proces de fetching. Această lucrare a avut loc pe un sistem de calcul (laptop) cu următoarele componente: Procesor (CPU) cu 4 nuclee Amd Ryzen 5 3550H, tactat la 2.1 GHz, Memorie RAM 8 GB dispusă pe un singur canal, Placă Video nVidia GTX 1650 cu 4 GB DRAM și memorie de stocare valabilă de 68 GB. În momentul de față, acest sistem de calcul se plasează în gama mid-range, de aceea valorile pot varia și nu prezintă o precizie bine determinată, în funcție de sistemul de calcul pe care se va propune să se implementeze o astfel de aplicație similară de recunoaștere facială.

Tabel 4.5:

Nr poze	Timp encodare	Memorie rulată
6	9.8s	0.10%
7	10.5s	0.10%
8	10.8s	0.20%
9	14s	0.40%
10	18.2s	1%
11	15.25s	0.20%
12	15.16s	0.70%
13	16.76s	0.20%
14	27.17s	2.50%

În ceea ce privește următorul tabel, acesta are rol predictiv pentru a estima cât de fiabil este un astfel de sistem pe măsură ce încărcăm baza de date, lucru extrem de util în cazul în care dorim să vedem ce cerințe trebuie respectate și ce hardware trebuie să avem pentru a îndeplini astfel de sarcini pentru o companie mai mare, sau de ce nu, pentru un oraș cu o populație mai mare de 1000 de locuitori.

Aflarea acestor valori a fost posibilă prin folosirea bibliotecilor *pandas* și *scikit-learn* din limbajul de programare python, cum se relevă în documentația [22]. S-au întocmit două fișiere .csv, fiecare având 2 coloane, în primul se află coloana "Nr poze" alături de coloana "Timp encodare", iar în cel de-al doilea avem o structură identică, doar că avem coloana "Memorie rulată". Pe aceste fișiere s-a aplicat pe rând un algoritm de regresie liniară, inspirat din referința [23], cu scopul de a face o predicție.


```

predictie.py x
1 import pandas
2 import matplotlib.pyplot as plt
3 from sklearn.linear_model import LinearRegression
4 data = pandas.read_csv('encodare.csv')
5 plt.scatter(data['NrPoze'], data['Timp'])
6 plt.show()
7 model = LinearRegression()
8 model.fit(data[['NrPoze']], data[['Timp']])
9 print(model.predict([[30]]))

```

Fig. 4.31: Algoritm Regresie Liniară

Tabel 4.6:

Nr poze	Timp encodare	Memorie rulată
6	9.8s	0.10%
7	10.5s	0.10%
8	10.8s	0.20%
9	14s	0.40%
10	18.2s	1%
11	15.25s	0.20%
12	15.16s	0.70%
13	16.76s	0.20%
14	27.17s	2.50%
20	31.12s	2.38%
25	39.25s	3275%
30	48.03s	4.16%
35	56.22s	5.05%
40	64.4s	5.95%
50	80.78s	7.73%
60	97.15s	9.51%
70	113.52s	11.30%
80	129.9s	13.08%
90	146.26s	14.86%
100	162.63s	16.65%
1000	1636.08s	177.15%

Putem spune despre acest tabel faptul că pe măsură ce baza de date are mai mulți utilizatori, intervine o îngreunare a procesului de execuție, de exemplu, dacă vorbim de o entitate juridică ce are 1000 de angajați, vor fi necesare aproximativ 1636 de secunde pentru a procesa atâția utilizatori în momentul în care se va realiza aplicația, adică în jur de 27 de minute, iar dacă dorim să extindem ordinul de mărime, la 100000 de utilizatori, un număr egal cu populația unui oraș de populație medie precum Râmnicu Vâlcea, avem nevoie de 45 de ore la dispoziție, iar în privința memoriei, aceasta este cu mult depășită de la 1000 de utilizatori, fiind obligatoriu nevoie de un calculator mai performant, ceea ce desigur, crește costurile.

CONCLUZII

Scopul acestei lucrări de licență este de a afla și cerceta impactul recunoașterii faciale pentru activitățile de autentificare și autorizare din viața de zi cu zi, în contextul evoluției grandioase a cercetării în domeniul inteligenței artificiale. De asemenea, se remarcă o înclinație spre tehnica Single Sign-On, o metodă de autentificare care permite utilizatorilor să se autentifice o singură dată cu scopul de a accesa mai multe aplicații ori resurse. Odată ce un utilizator s-a autentificat cu succes la o aplicație sau un portal centralizat, este automat autentificat la toate celelalte aplicații care acceptă SSO, fără a se repeta procesul de reintroducere a datelor de logare, care acestea pot fi biometrice, de aceea, dat fiind faptul că accesul la aceste aplicații sau resurse implică intersecția cu date sensibile și confidențiale, în special există o apetență pentru implementarea acestei metode în cadrul aplicațiilor bancare, este imperativ să existe un sistem care să elimine orice fel de amenințări al căror scop este să inducă în eroare beneficiarii primari ai aplicației și să capteze informații sensibile ce pun în pericol binele comunității.

Cercetarea ulterioară poate consta în: găsirea unei metode de a îmbunătăți rata de cadre pe secunde aferentă capturii video și foto a utilizatorului, reducerea spațiului de stocare necesar implementării aplicației, cât și identificarea unei tehnici de optimizare a timpului de execuție necesar etapei de fetching, pentru a se găsi o soluție pentru implementarea unei astfel de aplicații pe o bază de date foarte voluminoasă, cu un număr echivalent populației unei metropole europene.

Bibliografie

- [1] R. Kirsch/NIST, <https://www.nist.gov/mathematics-statistics/first-digital-image> (accesat la data de 20.04.2024).
- [2] Starmer, Josh. "The Statquest illustrated guide to machine learning!!!: master the concepts, one full-color picture at a time, from the basics all the way to neural networks. BAM!." (No Title) (2022).
- [3] Zhang, Dengsheng, and Dengsheng Zhang. "Wavelet transform." *Fundamentals of image data mining: Analysis, Features, Classification and Retrieval* (2019): 35-44.
- [4] Parida et al. , *Wavelet based transition region extraction for image segmentation. Future Computing and Informatics Journal*, 2017.
- [5] Wasilewski, Filip. "Pywavelets documentation." (2012).
- [6] I. Ciocoiu, *curs CIPS*, <http://scs.etc.tuiasi.ro/iciocoiu/courses/CIPS/course8/Capitolul1.pdf> (accesat la data de 20.04.2024).
- [7] Florin Leon, *curs ML*, http://florinleon.byethost24.com/Curs_ML/ML01_Clusterizare.pdf?i=1 (accesat la data de 21.04.2024).
- [8] Matthew Turk, Alex Pentland, *Eigenfaces for recognition*, 1991.
- [9] Andreas Tilevik - Tilestats, *Multivariate statistics - a full course*, 2022.
- [10] Z. Yu, Y. Qin, X. Li, C. Zhao, Z. Lei and G. Zhao, "Deep Learning for Face Anti-Spoofing: A Survey," în *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 5, pp. 5609-5631, 2023.
- [11] M. Khammari, "Robust face anti-spoofing using CNN with LBP and WLD," *IET Image Process.*, vol. 13, pp. 1880–1884, 2019
- [12] L. Feng et al., "Integration of image quality and motion cues for face anti-spoofing: A neural network approach," *J. Vis. Commun. Image Representation*, vol. 38, pp. 451–460, 2016.
- [13] Z. Yu, X. Li, P.Wang, and G. Zhao, "TransRPPG: Remote photoplethysmography transformer for 3D mask face presentation attack detection" *IEEE Signal Process. Lett.*, vol. 28, pp. 1290–1294, 2021.
- [14] L. Li, Z. Xia, X. Jiang, Y. Ma, F. Roli, and X. Feng, "3D face mask presentation attack detection based on intrinsic image analysis" *IET Biometrics*, vol. 9, pp. 100–108, 2020.
- [15] Lai, M.; van der Stel, S.D.; Groen, H.C.; van Gastel, M.; Kuhlmann, K.F.D.; Ruers, T.J.M.; Hendriks, B.H.W. *Imaging PPG for In Vivo Human Tissue Perfusion Assessment during Surgery. J. Imaging*, 2022, <https://doi.org/10.3390/jimaging8040094> (accesat la data de 22.04.2024).
- [16] Z. Wang et al., "Deep spatial gradient and temporal depth learning for face anti-spoofing," în *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 5041–5050, 2020.
- [17] I. J. S. Biometrics, "Information technology–biometric presentation attack detection–Part 3: Testing and reporting" *Standard, International Organization for Standardization, Geneva, Switzerland*, 2017.
- [18] CV Zone, *Face Attendance Course*, <https://www.computervision.zone/courses/face-attendance/> (accesat la 23.02.2024).
- [19] Python | Smile detection using OpenCV, <https://www.geeksforgeeks.org/python-smile-detection-using-opencv/> (accesat la 25.11.2023)
- [20] Nedelcu, Dorian, and Tihomir Latinovic. "PyChart–A Python module for analysis and visual view of 2D/3D Charts." *Journal of Physics: Conference Series*. Vol. 1781. No. 1. IOP Publishing, 2021.
- [21] Silent-Face-Anti-Spoofing, https://github.com/minivision-ai/Silent-Face-Anti-Spoofing/blob/master/README_EN.md, accesat la 21.03.2024

- [22] Pedregosa, Fabian, et al. "*Scikit-learn: Machine learning in Python.*" the Journal of machine Learning research 12 (2011): 2825-2830
- [23] Jolly, Kevin. *Machine learning with scikit-learn quick start guide: classification, regression, and clustering techniques in Python*. Packt Publishing Ltd, 2018.

Anexa 1 – Codul sursă al aplicației

Main.py

```
import pickle
import cv2
import os
import cvzone
import face_recognition
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db
from firebase_admin import storage
import numpy
from datetime import datetime
import time
from memory_profiler import profile
import psutil
import matplotlib.pyplot as plt
from test import test
import smtplib
from email.message import EmailMessage

def send_email(receiver, subject, message):
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.starttls()
    # Make sure to give app access in your Google account
    server.login('crispopesco25@gmail.com', 'tzai ttvs hhse mmcu')
    email = EmailMessage()
    email['From'] = 'crispopesco25@gmail.com'
    email['To'] = receiver
    email['Subject'] = subject
    email.set_content(message)
    server.send_message(email)

cred = credentials.Certificate("D:\ETTI\An IV\Licenta\curs 2h CV\cod python - scris manual\serviceAccountKey.json")
firebase_admin.initialize_app(cred, {
    'databaseURL': "https://faceattendancerealtime-5f9d2-default-rtdb.firebaseio.com/",
    'storageBucket': "faceattendancerealtime-5f9d2.appspot.com"
})
bucket=storage.bucket()
cap=cv2.VideoCapture(0)
cap.set(3,640)
cap.set(4,480)
#capDetectie = cv2.VideoCapture(0)
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
smile_cascade = cv2.CascadeClassifier('haarcascade_smile.xml')
#face=face_cascade.detectMultiScale(cap,1.3,5)
imgBackground = cv2.imread('D:\ETTI\An IV\Licenta\curs 2h CV\cod python - scris manual\Resources\background.png')
#Importarea imaginilor din Modes in lista
folderModePath = 'D:\ETTI\An IV\Licenta\curs 2h CV\cod python - scris manual\Resources\Modes'
modePathList = os.listdir(folderModePath)
imgModelList = [ ]
for path in modePathList:
    imgModelList.append(cv2.imread(os.path.join(folderModePath,path))) #se adauga in lista
#pozele 1-4.png indexate in FolderModePath+path
print(len(imgModelList))
#incarcarea fisierului de encoding
print("Loading Encode File...")
file=open('EncodeFile.p', 'rb')
encodeListKnownWithIds = pickle.load(file)
file.close()
encodeListKnown,studentIds = encodeListKnownWithIds
#print(studentIds)
print("Encode File Loaded")
modeType=0
counter=0
id=-1
ok=0
imgStudent=[ ]
while True:
    #_, frame = capDetectie.read()
    success, img = cap.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

```

face = face_cascade.detectMultiScale(gray, 1.3, 5)
imgS = cv2.resize(img, (640, 480))
imgColor = imgS # backup imagine color
imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)
faceCurFrame = face_recognition.face_locations(imgS)
encodeCurFrame = face_recognition.face_encodings(imgS, faceCurFrame)
imgBackground[162:162+480, 55:55+640] = img
imgBackground[44:44+633, 808:808 + 414] = imgModeList[modeType]
if faceCurFrame: #daca se detecteaza o fata cunoscuta
    for encodeFace, faceLoc in zip(encodeCurFrame, faceCurFrame):
        matches=face_recognition.compare_faces(encodeListKnown, encodeFace)
        faceDis = face_recognition.face_distance(encodeListKnown, encodeFace)
        #print("matches", matches)
        #print("faceDis", faceDis)
        matchIndex=numpy.argmin(faceDis) #se extrage indexul cu valoarea minima, in cazul
nostru ce fata se potriveste
        if matches[matchIndex]:
            #for x,y,w,h in face:
            #print("Known face detected")
            #print(studentIds[matchIndex])
            y1,x2,y2,x1 = faceLoc
            for c1, c2, c3, c4 in face:
                cv2.rectangle(img, (c1, c2), (c1 + c3, c2 + c4), (0, 255, 255), 2)
                face_roi = img[c2:c2 + c4, c1:c1 + c3]
                gray_roi = gray[c2:c2 + c4, c1:c1 + c3]
                smile = smile_cascade.detectMultiScale(face_roi, 1.3, 25)
                for s1, s2, s3, s4 in smile:
                    cv2.rectangle(face_roi, (s1, s2), (s1 + s3, s2 + s4), (0, 0, 255), 2)
            #y1, x2, y2, x1 = y1*4, x2*4, y2*4, x1*4
            bbox= 55+x1, 162+y1, x2-x1, y2-y1 # la sfarsit avem width and height
            label=test(image=imgS,model_dir='D:\\ETTI\\An IV\\Licenta\\curs 2h
CV\\cod python - scris manual\\Silent-Face-Anti-Spoofing-
master\\resources\\anti_spoof_models',device_id=0)
            if label == 1:
                imgBackground = cvzone.cornerRect(imgBackground, bbox, rt=0) #
dreptunghi in jurul fetei de grosime=0
                id=studentIds[matchIndex]
                if counter == 0:
                    cvzone.putTextRect(imgBackground, "Procesare", (275, 400))
                    cv2.imshow("Identificare chip", imgBackground)
                    cv2.waitKey(1)
                    counter=1
                    modeType=1
                else: ok=ok+1
                if ok==1:
                    send_email('cristianirimescu25@gmail.com', 'TENTATIVA FRAUDARE',
'ATENTIE, se incearca o fraudare!')

            if counter!=0:
                #preia datele despre individ
                if counter==1:
                    studentInfo=db.reference(f'Students/{id}').get()
                    print(studentInfo)
                #preia imaginea individului
                blob=bucket.get_blob(f'D:\\ETTI\\An IV\\Licenta\\curs 2h CV\\cod python - scris
manual\\Images/{id}.png')
                array=numpy.frombuffer(blob.download_as_string(), numpy.uint8)
                imgStudent=cv2.imdecode(array, cv2.COLOR_BGRA2BGR) #se descarca imaginea prin
decodificare in format BGR
                #actualizeaza datele despre prezenta
                datetimeObject=datetime.strptime(studentInfo['last_attendance_time'],
"%Y-%m-%d %H:%M:%S")
                secondsElapsed=(datetime.now()-datetimeObject).total_seconds()
                print(secondsElapsed)
                if secondsElapsed>30:
                    ref=db.reference(f'Students/{id}')
                    studentInfo['total_attendance']=studentInfo['total_attendance']+1
#actualizam numar prezente
                    ref.child('total_attendance').set(studentInfo['total_attendance'])
                    ref.child('last_attendance_time').set(datetime.now().strftime("%Y-%m-%d
%H:%M:%S")) #actualizam data noua de prezenta
                else:
                    modeType=3
                    counter=0
                    imgBackground[44:44 + 633, 808:808 + 414] = imgModeList[modeType]
                if modeType!=3:
                    if 10<counter<20:

```



```

        modeType=2
        imgBackground[44:44 + 633, 808:808 + 414] = imgModeList[modeType]
        if counter<=10:
            cv2.putText(imgBackground, str(studentInfo['total attendance']), (861,125),
                        cv2.FONT_HERSHEY_COMPLEX, 1, (255,255,255), 1)
#pozitie, font, culoare, grosime pentru a afisa numarul de prezente
            cv2.putText(imgBackground, str(studentInfo['major']), (1006, 550),
                        cv2.FONT_HERSHEY_COMPLEX, 0.5, (255, 255, 255), 1)
            cv2.putText(imgBackground, str(id), (1006, 493),
                        cv2.FONT_HERSHEY_COMPLEX, 0.5, (255, 255, 255), 1)

            (w,h),_ =cv2.getTextSize(studentInfo['name'],cv2.FONT_HERSHEY_COMPLEX,1,1)
            offset=(414-w)//2
            cv2.putText(imgBackground, str(studentInfo['name']), (808+offset, 445),
                        cv2.FONT_HERSHEY_COMPLEX, 1, (50, 50, 50), 1)
            imgBackground[175:175+216,909:909+216]=imgStudent
            counter=counter+1
        if counter>=20:
            counter=0
            modeType=0
            studentInfo=[]
            imgStudent=[]
            imgBackground[44:44+633 , 808:808 + 414] = imgModeList[modeType]
        else:
            modeType=0
            counter=0
            #cv2.imshow("Webcam", img)
            cv2.imshow("Identificare chip", imgBackground)
            cv2.imwrite('ultima_poza.jpg', imgColor)
            if cv2.waitKey(10) == ord('q'):
                break

```

EncodeGenerator.py

```

import cv2
import face_recognition
import pickle
import os
import time
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db
from firebase_admin import storage
import psutil
import matplotlib.pyplot as plt
from memory_profiler import profile
# Înregistrați memoria utilizată înainte de execuție
memory_before = psutil.virtual_memory().percent
# Înregistrați timpul de început
start_time = time.time()
cred = credentials.Certificate("D:\ETTI\An IV\Licenta\curs 2h CV\cod python - scris manual/serviceAccountKey.json")
firebase_admin.initialize_app(cred,{
    'databaseURL':"https://faceattendancerealttime-5f9d2-default-rtdb.firebaseio.com/",
    'storageBucket':"faceattendancerealttime-5f9d2.appspot.com"
})
#Importarea imaginilor din Images in lista
folderPath = 'D:\ETTI\An IV\Licenta\curs 2h CV\cod python - scris manual\Images'
pathList = os.listdir(folderPath)
print(pathList)
imgList = [ ]
studentIds=[ ]
for path in pathList:
    imgList.append(cv2.imread(os.path.join(folderPath,path)))
    studentIds.append(os.path.splitext(path)[0])
    fileName=f'{folderPath}/{path}'
    bucket=storage.bucket()
    blob=bucket.blob(fileName)
    blob.upload_from_filename(fileName)
    #print(path)
    #print(os.path.splitext(path)[0])
print(studentIds)

def findEncodings(imagesList):

```

```

encodeList=[ ]
for img in imgList:
    img=cv2.cvtColor(img,cv2.COLOR_BGR2RGB) #conversie imagine din format BGR in format
RGB suportat
    encode=face_recognition.face_encodings(img)[0] #variabila in care se petrece encodarea
    encodeList.append(encode)

return encodeList
print("Encoding started...")
encodeListKnown = findEncodings(imgList)
encodeListKnownWithIds = [encodeListKnown,studentIds]
#print(encodeListKnown)
print("Encoding Complete")
file=open("EncodeFile.p", 'wb')
pickle.dump(encodeListKnownWithIds, file)
file.close()
print("File saved")
end_time = time.time()
elapsed_time = end_time - start_time
memory_after = psutil.virtual_memory().percent
print(f"Timpul de execuție: {elapsed_time:.2f} secunde")
print(f"Memorie înainte de execuție: {memory_before:.2f}%")
print(f"Memorie după execuție: {memory_after:.2f}%")
labels = ['Memorie înainte', 'Memorie după']
values = [memory_before, memory_after]

plt.bar(labels, values)
plt.ylim(0, 90) # Seteaza limita pentru axa Y între 0 și 90 (procente)
plt.title('Utilizarea memoriei înainte și după execuție')
plt.ylabel('Memorie (%)')
plt.show()
labels2 = ['Timp de executie']
values2=elapsed_time
plt.bar(labels2, values2)
plt.ylim(8,10 )
plt.title('Timp de executie')
plt.ylabel('Timp (s)')
plt.show()

```

AddDataToDatabase.py

```

import firebase_admin
from firebase_admin import credentials
from firebase_admin import db
cred = credentials.Certificate("D:\ETTI\An IV\Licenta\curs 2h CV\cod python - scris
manual/serviceAccountKey.json")
firebase_admin.initialize_app(cred,{
    'databaseURL':"https://faceattendancerealtime-5f9d2-default-rtdb.firebaseio.com/"
})

ref=db.reference('Students')
data = {
    "852741": #cheia
        { #valoarea - compusa din toate campurile de jos
            "name": "Emily Blunt",
            "major": "Economics",
            "starting_year":2018,
            "total_attendance":12,
            "standing":"B",
            "year":2,
            "last_attendance_time":"2022-12-11 00:54:34"
        },
    "963852": #cheia
        { #valoarea - compusa din toate campurile de jos
            "name": "Elon Musk",
            "major": "Physics",
            "starting_year":2020,
            "total_attendance":7,
            "standing":"G",
            "year":2,
            "last_attendance_time":"2022-12-11 00:54:34"
        },
    "973853": #cheia
        { #valoarea - compusa din toate campurile de jos
            "name": "Irimescu Ovidiu",

```

```

        "major": "Electronics",
        "starting_year": 2020,
        "total_attendance": 5,
        "standing": "G",
        "year": 4,
        "last_attendance_time": "2021-12-9 00:54:34"
    },
    "973854": { #cheia
        { #valoarea - compusa din toate campurile de jos
            "name": "Irimescu Teodora",
            "major": "Pupil",
            "starting_year": 2020,
            "total_attendance": 20,
            "standing": "G",
            "year": 4,
            "last_attendance_time": "2021-12-9 00:54:34"
        }
    },
    "973855": { #cheia
        { #valoarea - compusa din toate campurile de jos
            "name": "Irimescu Ionut",
            "major": "Mechanics",
            "starting_year": 2020,
            "total_attendance": 5,
            "standing": "D",
            "year": 4,
            "last_attendance_time": "2021-12-9 00:54:34"
        }
    },
    "973856": { #cheia
        { #valoarea - compusa din toate campurile de jos
            "name": "Irimescu Ion",
            "major": "Management",
            "starting_year": 2005,
            "total_attendance": 20,
            "standing": "D",
            "year": "F",
            "last_attendance_time": "2021-12-9 00:54:34"
        }
    },
    "973857": {
        "name": "Ashton Kutcher",
        "major": "Management",
        "starting_year": 2005,
        "total_attendance": 20,
        "standing": "D",
        "year": "F",
        "last_attendance_time": "2021-12-9 00:54:34"
    },
    "973858": {
        "name": "David Schwimmer",
        "major": "Management",
        "starting_year": 2005,
        "total_attendance": 20,
        "standing": "D",
        "year": "F",
        "last_attendance_time": "2021-12-9 00:54:34"
    },
    "973859": {
        "name": "Michael C Hall",
        "major": "Management",
        "starting_year": 2005,
        "total_attendance": 20,
        "standing": "D",
        "year": "F",
        "last_attendance_time": "2021-12-9 00:54:34"
    },
    "9738510": {
        "name": "Topher Grace",
        "major": "Management",
        "starting_year": 2005,
        "total_attendance": 20,
        "standing": "D",
        "year": "F",
        "last_attendance_time": "2021-12-9 00:54:34"
    },
    "9738511": {
        "name": "Elizabeth Mitchell",
        "major": "Management",
        "starting_year": 2005,

```

```

        "total_attendance": 20,
        "standing": "D",
        "year": "F",
        "last_attendance_time": "2021-12-9 00:54:34"
    },
    "9738512":
    { "name": "Kenneth Branagh",
      "major": "Management",
      "starting_year": 2005,
      "total_attendance": 20,
      "standing": "D",
      "year": "F",
      "last_attendance_time": "2021-12-9 00:54:34"
    },
    "9738513":
    { "name": "Julia Roberts",
      "major": "Management",
      "starting_year": 2005,
      "total_attendance": 20,
      "standing": "D",
      "year": "F",
      "last_attendance_time": "2021-12-9 00:54:34"
    },
    "9738514":
    { "name": "Bruce Boxleitner",
      "major": "Management",
      "starting_year": 2005,
      "total_attendance": 20,
      "standing": "D",
      "year": "F",
      "last_attendance_time": "2021-12-9 00:54:34"
    }
}
for key, value in data.items():
    ref.child(key).set(value)

```