UNIVERSITATEA BABEŞ-BOLYAI CLUJ-NAPOCA
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ
SPECIALIZAREA INFORMATICĂ

LUCRARE DE LICENŢĂ

Gestionarea prezențelor cu ajutorul recunoașterii faciale

Conducător ştiinţific
Profesor Universitar, Dr. Pop F. Horia

Absolvent
Pop Adrian Cristian

2022

ABSTRACT

In these days keeping maintaining the attendance records can be a challenging task especially when working with a large volume of people. There are a lot of methods for the management of attendances, the most common being calling the name of the person, manually marking the name of a person in a table or RFID cards. These classical methods are time consuming and are prone to error or in the case of the RFID cards, they can be lost.

The purpose of this thesis is find the optimum solution for the problem of attendance management through the use of facial recognition. During the next pages, the most popular facial recognition algorithms will be described and compared in terms of accuracy. Then, the algorithm with the best results will be used in a face recognition attendance management system.

The first chapter represents a short introduction. The second chapter goes through the history of face recognition and talks about some of the most important moments for the development of this technology. Then, the third chapter will analyze how is facial recognition working, and describe some of the most popular face recognition algorithms. The fourth chapter will analyze and compare the performance of the previously described algorithms. The next chapter will talk about different attendance management techniques. In chapter number 6 the IntAttend face recognition attendance management system will be presented. And the last chapter represents the conclusions of this thesis.

Cuprins

Capitolul 1

Introducere

Cu toate că tehnologia de recunoașterea facială a fost prezentă într-o formă sau alta începând cu anii 1960, aceasta a căpătat popularitate doar în ultimii ani. Dacă acum câțiva ani părea de domeniul științifico fantasticului, acum există foarte multe utilizări ale recunoașterii faciale, identificarea criminalilor, găsirea persoanelor dispărute, deblocarea telefoanelor mobile și multe altele.

În această lucrare vom studia folosirea recunoașterii faciale pentru gestiunea prezențelor într-o organizație. Gestionarea prezențelor este o sarcină importantă în orice instituție. Există numeroase metode de marcare a prezențelor, cele mai răspândite fiind folosirea cartelelor de pontaj sau marcarea manuală. Aceste metode sunt consumatoare de timp și există riscul de a apărea erori.

Folosirea recunoașterii faciale pentru gestiunea prezențelor este una dintre cele mai eficiente metode. Este cea mai rapidă metodă și riscul apariției erorilor este mai scăzut decât în cazul celorlalte metode.

Această lucrare are ca scop studierea și compararea mai multor algoritmi de recunoaștere facială din punct de vedere al timpului de execuție și al acurateței și alegerea celei mai bune soluții pentru problema gestionării prezențelor. Atenția va cădea pe algoritmii Eigenfaces, Fisherfaces, metoda histogramelor cu modele binare locale (LBPH) și rețelele neuronale convoluționale (CNN).

În al doilea capitol vom prezenta o scurtă istorie a recunoașterii faciale începând cu primele apariții ale acesteia în anii 1960 și până în zilele noastre. Printre cele mai semnificative evenimente din istoria recunoașterii faciale se enumeră, inventarea primului sistem de recunoaștere facială manuală de către Woodrow Bledsoe, dezvoltarea abordării Eigenfaces cu ajutorul algebrei liniare, testarea algoritmului la evenimentul Super Bowl din anul 2022 și răspândirea recunoașterii faciale cu ajutorul social media.

Capitolul al treilea va explica cum funcționează sistemele de recunoaștere facială. Recunoașterea facială este definită ca un proces în patru etape: detectarea feței, procesarea imaginilor, extragerea trăsăturilor și identificarea subiectului. Fie-

care dintre aceste etape va fi prezentată. Se vor prezenta două abordări diferite pentru etapa detectării faciale, Clasificatorul Haar-Cascade și Histograma Gradienților Orientați. În cadrul etapei de identificare a persoanei se va explica modul de funcționare a algoritmilor Eigenfaces, Fisherfaces, LBPH și CNN. Algoritmii discutați anterior vor fi testați și comparați din punct de vedere al timpului de execuție și al corectitudinii rezultatelor. Rezultatul acestei comparații va fi folosit pentru a decide care este cel mai bun algoritm pentru implementarea unui sistem de gestiune a prezențelor cu ajutorul recunoașterii faciale.

Ultimul capitol al lucrării va prezenta sistemul IntAttend - un sistem inteligent de gestionare a prezențelor cu ajutorul recunoașterii faciale. Vom avea în vedere arhitectura, funcționalitățile, implementarea și testarea acestui sistem.

Contribuția personală asupra acestei lucrări a fost adusă prin studierea modului de funcționare a celor mai populari algoritmi de recunoaștere facială și testarea acestora. În urma testării s-au obșinut niște rezultate personale care au fost folosite pentru compararea algoritmilor. Studiul comparaiv al algoritmilor a condus la stabilirea algoritmului optim pentru soluționarea problemei gestionării prezențelor. Acest algoritm a fost folosit pentru implementarea sistemului IntAttend - o soluție unică pentru gestionarea prezențelor cu ajutorul recunoașterii faciale.

Capitolul 2

Istoria recunoașterii faciale

Recunoașterea facială este o metoda de identificare a unui individ cu ajutorul imaginii feței acestuia. Imaginile pot să provină fie din poze, fie din videoclipuri.

Ființele umane recunosc fețe în fiecare zi, în mod automat și fară nici un efort. Acest lucru pare foarte simplu pentru oameni, dar pentru calculatorare este o sarcină complexă care implică multe variablie care pot afecta rezultatul.

Până nu de mult, tehnologia de recunoaștere facială părea de domeniul științifico fantasticului, dar în ultimii ani această tehnologie revoluționară a devenit nu numai posibilă, ci și foarte răspândită.

Ar putea părea că tehnologia de recunoaștere facială a apărut peste noapte, dar de fapt se lucreaz de mult timp asupra ei.

2.1 Măsurătorile manuale ale lui Bledsoe

Istoria recunoașterii faciale începe în anii 60 când matematicianul Woodrow Wilson Bledsoe a dezvoltat un sistem care putea clasifica fețele oamenilor folosind un dispozitiv denumit tableta RAND. Tableta RAND permitea introducerea coordonatelor pe o grilă folosind un stylus care emitea unde electromagnetice.

Cu ajutorul acestui dispozitiv, Bledsoe înregistra coordonatele trasăturilor feței unei persoane cum ar fi ochii sau gura apoi le insera într-o bază de date. Prin introducerea unei noi fotografii cu fața unei persoane în sistem, acesta era capabil să găsească cea mai asemănătoare poză existentă deja in baza de date.

Cu toate că sistemul lui Bledsoe era destul de limitat, acesta a reprezentat un important prim pas în dezvoltarea recunoașterii faciale.

2.2 Progrese asupra preciziei recunoașterii faciale

Pornind de la munca lui Bledsoe, în anii 70, Goldstein, Harmon și Lesk au reușit să crească precizia recunoașterii faciale. Aceștia au folosit 21 de markeri specifici, cum ar fi colțurile gurii și al nasului, culoarea părului și grosimea buzelor. Cu toate că precizia a evolut, ca și în cazul sisemului lui Bledsoe, datele biometrice încă trebuiau caluclate manual.

2.3 Algebra liniară în recunoașterea facială

La sfârșitul anilor 80 am observat noi progrese asupra recunoașterii faciale. În anul 1988, Sirovich și Kirby au început să folosească algebra liniară la problema recunoașterii faciale.

Această abordare a căpătat a devenit cunoscută sub numele de abordarea Eigenface și a arătat că analiza caracteristicilor principale ar putea fi utilizată pe o colecție de imagini faciale pentru a forma un set de caracteristici de bază.

În 1991, Turk și Pentland au continuat munca lui Sirovich și Kirby. Aceștia au dezvoltat abordarea Eigenfaces. Cu ajutorul acestei metode au reușit să descopere cum să detecteze fețele în imagini. În acest fel a apărut primul caz de rucunoaștere facială automată. Această descoperire semnificativă a deschis calea pentru viitoare dezvoltări în tehnologia recunoașterii faciale[FAC20].

2.4 Programul FERET

La începutul anilor 90, Agenția pentru Proiecte de Cercetare Avansată a Apărării (DARPA) și Institutul Național de Standarde și Tehnologie au lansa programul FERET (Face Recognition Technology) pentru a încuraja piața comercială a recunoașterii faciale. Proiectul presupunea creearea unei baze de date cu imagini faciale. În setul de testare au fost incluse 2413 imagini faciale reprezentând 856 de persoane. Baza de date a fost actualizată în anul 2003 pentru a include versiuni color de înaltă calitate ale imaginilor.

Speranța a fost că o bază de date mare de imagini de testare pentru recunoașterea facială va fi capabilă să inspire inovație, care ar putea duce la o tehnologie de recunoaștere facială mai puternică[SAAW21].

2.5 Super Bowl XXXV (2002)

La evenimentul Super Bowl din 2002, recunoașterea facială a fost testată. Cu toate că oficialii au rapotat că mai mulți infractori au fost identificați, testul a fost declarat un eșec. Răspunsurile fals-pozitive și criticile aduse în urma testului au arătat că tehnologia de recunoaștere facială nu era încă pregătită.

La acea vreme, recunoașterea facială nu funcționa încă bine în mulțimile mari de persoane, această funcționalitate fiind esențială pentru asigurarea securității evenimentului.

2.6 Social Media

Începând cu 2010 Facebook a început să implementeze o funcționalitate bazată pe recunoașterea facială care ajută la identificarea și etichetarea persoanelor ale căror fețe apar în fotografii. Cu toate că au aparut numeroase discuții legate de confidențialitate, utilizatorii Facebook nu au fost deranjați de această funcționalitate. Drept urmare, în zilele noastre, peste 350 de milioane de foografii sunt încărcate și etichetate zilnic cu ajutorul recunoașterii faciale.

2.7 Identificarea lui Osama Bin Laden (2011)

Recunoașterea faciala a început sa fie din ce în ce mai folosită de către forțele de ordine și profesioniștii militari. În prezent este cea mai eficientă metodă de identificare a cadavrelor. După ce Osama Bin Laden a fost ucis într-un raid din SUA, identitatea lui a fost confirmată cu ajutorul recunoașterii faciale.

2.8 iPhone X (2017)

Începând cu 2010, recunoașterea faciala a avansat rapid. 12 septembrie a fost o dată seminficativă pentru integrarea recunoașterii faciale în viața noastră de zi cu zi. Aceasta a fost data la care Apple a lansat iPhone X. Una dintre caracteristicile principale ale acestui dispozitiv este utlizarea recunoașterii faciale pentru securitate. Faptul că iPhone X s-a vândut aproape instant demonstrează că recunoașterea facială este acceptată de către utilizatori ca standard pentru securitatea dispozitivelor.

Capitolul 3

Cum funcționează recunoașterea facială?

Ființele umane recunosc fețele în mod automat, în fiecare zi. Probabil este ușor să identificăm fața unei persoane cunoscute. Identificăm chipul unei persoane prin, particularitățile aceseia, ochii, nasul, gura.

Sistemele de recunoaștere facială funcționează la fel, dar la un nivel algoritmic. Fiecare sistem de recunoaștere facială identifică trăsăturile faciale a unei persoane dintr-o fotografie, convertește aceste trăsături în date și le compară cu datele deja existente într-o bază de date.

3.1 Procesul de recunoaștere facială

Sistemele de recunoaștere facială pot varia, dar în general tind să funcționeze după cum urmează.3.1.

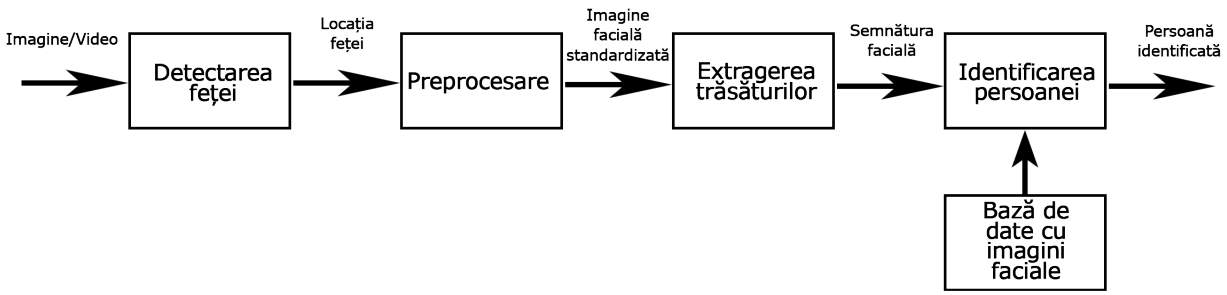

Figura 3.1: Procesul de recunoaștere facială

Un sistem de recunoaștere facială primește mereu ca intrare o imagine sau un video și întoarce ca rezulta identificarea unuia sau a mai multor indivizi dint-o fotografie sau videoclip. Un sistem de recunoaștere facială este definit ca un proces în patru etape: detectarea feței, preprocesarea imaginii, extragerea trasăturilor și identificarea subiectului.

Detectarea facială este definită ca etapa de extragere a unei fețe dintr-o scenă, fie ea fotografie sau videoclip. În această etapă sistemul identifica o anumită regiune a unei scene ca fiind o față.

Următorul pas - preprocesarea imaginii implică normalizarea imaginilor obținute anterior. În această fază imaginile vor suferi anumite modificări pentru a ajunge la un standard. Printre aceste modificări regăsim, redimensionarea imaginii și centrarea feței.

Extragerea trăsăturilor presupune obținerea trăsăturilor principale din fotografia unei fețe, de exemplu poziția ochilor și a nasului, distanța dintre ochi, distanța de la frunte la bărbie.

Ultimul pas, recunoașterea individului, presupune identificarea subiectului.

3.2 Detectarea feței

Detectarea feței este un prim pas necesar în sistemele de recunoaștere facială având ca scop localizarea și extragerea regiunii feței dintr-o imagine. Detectarea feței este o sarcină pe care oamenii o pot face fară efort, dar pentru calculatoare nu este o sarcină tocmai ușoară. Soluția problemei de detectare facială implică segmentarea, extragerea și verificarea fețelor dintr-o fotografie. Un sistem de detectare facială ar trebui să poată îndeplini sarcina indiferent de iluminare, orientare sau distanță[HL01]. Există numeroși algoritmi de detectare facială. în continuare vom analiza doi dintre ei.

3.2.1 Clasificatorul Haar-Cascade (HCC)

Clasificatorul Haar-Cascade este o abordare eficientă de detectare a obiectelor. Aceasta a fost propusă în anul 2001 de către Paul Viola și Michael Jones în lucrarea lor, "Rapid Object Detection using a Boosted Cascade of Simple Features [VJ01]". Metoda constă în împărțirea imaginii în subregiuni și calcularea trăsăturilor de tip Haar pe fiecare subregiune folosind imaginea integrală. Carcateristicile Haar sunt reprezentate de dreptunghiuri împărțite în două, trei sau patru alte dreptunghiuri. Urmatoarea imagine prezintă diferitele caracteristici de tip Haar 3.2.

Fiecare caracteristică este reprezentată printr-o singură valoare obținută prin calcularea diferenței dintre suma intensității pixelilor din regiunea albă și suma intensității pixelilor din regiunea neagră. Scopul acestei abordări este de a găsi o

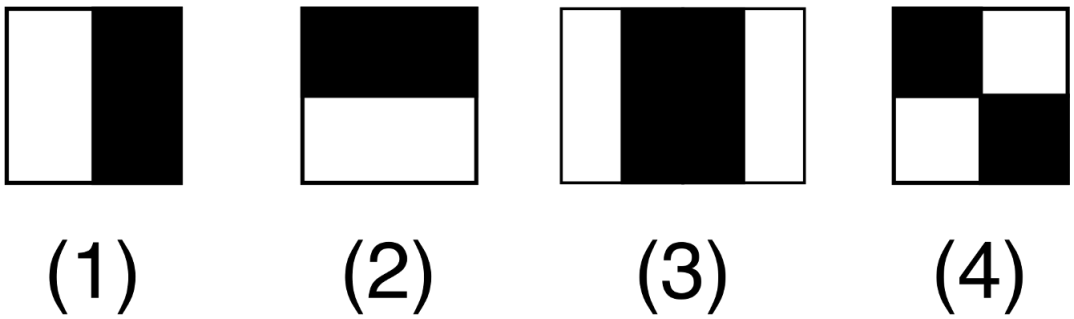

Figura 3.2: Caracteristici de tip Haar

combinație de trăsături Haar care reprezintă o față 3.3.

Figura 3.3: Exemplu de combinație de trăsături Haar

3.2.2 Histograma Gradienților Orientați (HOG)

Histograma Gradienților Orientați (HOG) este o altă metodă de detectare a obiectelor care poate fi folosită și pentru detectarea facială. Această metodă a fost propusă de Navneet Dalal și Bill Triggs, în lucrarea "Histograms of Oriented Gradients for Human Detection [DT05]", 2005. Această metodă necesită imagini în tonuri de gri. Fiecare pixel al imaginii este reprezentat de un număr întreg.

Metoda HOG compară fiecare pixel cu vecinii săi și găsește direcția în care imaginea se întunecă. O săgeată albă va fi folosită pentru a reprezenta această schimbare. Scopul acestei metode este de a deteta orice față, dar dacă păstrăm prea multe detalii, atunci doar fețele pe care aplicăm această metodă vor putea fi detectate. Așadar, doar direcțiile relevante vor fi păstrate.

Următorul pas este împărțirea imaginii în blocuri de 16x16 pixeli. Numărăm de câte ori este identificată o nouă direcție, iar în fiecare bloc se trasează o singură linie pentru direcția care a fost identificată de cele mai multe ori. Modelul se va antrena cu un număr cât mai mare de imagini frontale cu fețe pentru a obține cele mai bune rezultate.

Dupa ce algoritmul a fost antrenat este pregătit pentru utilizare. Pentru detectarea unei fețe, algoritmul va trece prin imagine, va calcula schimbările de direcție și va verifica dacă au fost identificate fețe.

Această metodă nu este afectată de schimbările luminozității imaginii. Daca imaginea este mai întunecată, atunci toți pixelii vor fi mai întunecați. Ar putea apărea probleme atunci când doar o parte a imaginii este afectată de lumină, de exemplu din cauza unui bec [Del17].

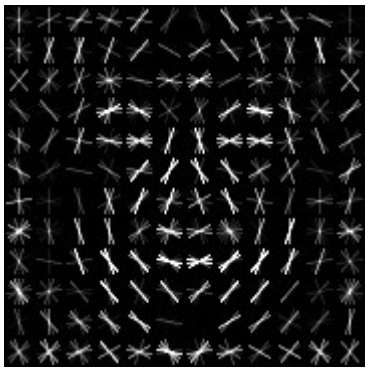

Figura 3.4: Exemplu de detectare facială cu metoda HOG

3.3 Preprocesarea imaginilor

Oricare dintre metodele discutate anterior pot fi folosite pentru a detecta fețe în imagini. Următorul pas în procesul de recunoaștere facială este preprocesarea fețelor detectate anterior pentru a îmbunătăți șansele de a recunoaște în mod corect o persoană. Nu toate imaginile vor avea același zoom, dimensiune sau centrare. Este important ca aceste imagini să fie normalizate.

De obicei algoritmii de recunoaștere facială necesită ca fețele din imagini să fie centrate în același mod. Scopul este ca nivelul ochilor și poziția nasului sa fie la fel în toate imaginile. Pentru a face acest lucru este nevoie să folosim un algoritm de recunoaștere a trăsăturilor feței pentru a găsi nasul și gura. Această sarcină poate fi îndeplinită de clasificatorul Haar, sau orice alt detector de trăsături umane.

Dupa centrarea imaginilor, acestea trebuie aduse la aceeași dimensiune și decupate astfel încât să cuprindă doar fețele. Este recomandată o dimensiune mai mică pentru imagini, deoarece în acest fel acestea sunt mai ușor de procesat de către algoritmul de recunoaștere facială. Este important să găsim dimensiunea optimă pentru imagini, ca să îmbunătățim cât mai mult rezultatele și timpul recunoașterii faciale, dar să nu pierdem informații importante.

3.4 Extragerea trăsăturilor și identificarea persoanei

Următorul pas în procesul de recunoaștere facială este extragerea trăsăturilor și în sfârșit identificarea persoanei. Acești doi pași funcționează împreună, așadar vom vorbi despre ei într-un singur capitol.

Pentru oameni este o sarcină ușoară să recunoască fețele persoanelor cunoscute. Nu este foarte dificil să identificăm un cunoscut chiar dacă acesta poartă ochelari sau mască. Toate aceste sarcini sunt triviale pentru oameni, dar reprezintă o adevărată provocare pentru calculatoare. Principala problemă a recunoașterii faciale este extragerea de informații dintr-o fotografie. Aceste informații trebuie să fie precise deoarece sunt folosite pentru identificarea unui subiect.

Procesul de extragere a trăsăturilor funcționează în felul următor. În primul rând, un algoritm va exrage trăsăturile dintr-o imagine, transformă aceste trăsături în date relevante pentru recunoașterea facială, alege cele mai bune trăsături și renunță la trăsăturile nerelevante.

De îndată ce trăsăturile sunt extrase și selectate, următorul pas este clasificiarea imaginii, adica identificarea propziu zisă. Sistemul compară trăsăturile obținute anterior cu o bază de date și identifică persoana pe baza acestor trăsături.

Unele sisteme de recunoaștere facială sunt concepute penru a calcula un scor de potrivire între persoana necunoscută și semnăturile faciale stocate în baza de date. Aceste sisteme vor oferi mai multe potriviri potențiale, clasate în ordinea probabilităților, în loc să returneze un singur rezultat.

Sistemele de recunoaștere facială diferă din punct de vedere al capacității de a identifica persoane în condiții dificile, cum ar fi imagini blurate, lumină slabă, rezoluția imaginii de calitate scăzută.

Când vine vorba de erori, avem doua situații pe care trebuie să le avem în vedere: rezultatele fals pozitive și rezultatele fals negative.

Rezultatele fals pozitive sunt întâlnite în situațiile în care sistemul de recunoaștere facială, potrivește în mod eronat fața unei persoane cu o imagine din baza de date.

Întâlnim un rezultat fals negativ atunci când sistemul nu reușeste să găsească o potrivire pentru chipul unei persoane cu o imagine din baza de date chiar dacă persoana respectivă este conținută în baza de date. Cu alte cuvinte, sistemul va returna în mod eronat un răspuns negativ.

Este important ca atunci când cercetăm un algoritm de recunoaștere facială să fim atenți la rata de răspunsuri fals pozitive și fals negative ale acestuia. De exemplu, atunci când un algoritm de recunoaștere facială este utilizat pentru a debloca telefonul este mai bine ca algoritmul să întoarcă răspunsuri fals negative, decat răspunsuri fals pozitive. Cu alte cuvinte este mai bine ca algoritmul să nu identifice proprietarul telefonului de cateva ori, decât să deblocheze telefonul pentru o altă persoană. Pe de

altă parte, dacă un algoritm de recunoaștere facială este folosit pentru identificarea infractorilor, ne dorim ca acesta să întoarcă cât mai puține rezultate fals poztive, în caz contrar, o persoană nevinovată ar putea ajunge la închisoare.

În continuare vom analiza unii dintre cei mai cunoscuți algoritmi de recunoaștere facială.

3.4.1 Eigenfaces

Metoda Eigenfaces, folosită prima dată de Turk si Pentland în 1991 [TP91], este o metodă de recunoaștere facială bazată pe o abordare statistică. Această tehnică are ca scop extragerea componentelor principale care afectează cel mai mult variația imaginilor. Algoritmul folosește imagini în tonuri de gri pentru antrenare. Metoda Eigenfaces este bazată pe analiza componenelor principale pentru a reduce numărul dimensiunilor în timp ce păstrează informațiile esențiale. O dimensiune este reprezentată de un pixel din imagine.

Antrenarea algoritmului Eigenfaces

În faza de antrenare a algoritmului identificăm următorii pași:

1. Citirea tuturor imaginilor de antrenament
2. Transformarea imaginilor în matrice

Pixelii imaginilor au ca și corespondenți valori numerice, așadar putem să convertim foarte ușor aceste imagini în matrici unde fiecare element este reprezentat de un pixel.

3. Transformarea matricilor în vectori

O matrice ocupă mai mult spațiu comparativ cu un vector. Așadar matricile obținute anterior vor fi transformate în vectori. Pentru a face acest lucru, fiecare rând al unei matrici va fi concatenat și apoi transpus pentru a forma un vector imagine.

4. Calcularea vectorului medie

Se calculează suma tuturor vectorilor imagine, apoi această suma este împărțită la numărul de vectori.

5. Scăderea vectorului medie din fiecare vector imagine
6. Calcularea matricei de covarianță

Se calculează cu următoarea formulă:

$$C = \frac{1}{M} \sum_{n=1}^{M} \Phi_n \Phi_n^T = AA^T$$

Unde C reprezintă matricea de covarianță, M este numărul de vectori, iar Φ reprezintă un vector imagine din care s-a scăzut media vectorilor Vectorii Φ grupați reprezintă matricea A, iar matricea A^T reprezintă transpusa acesteia.

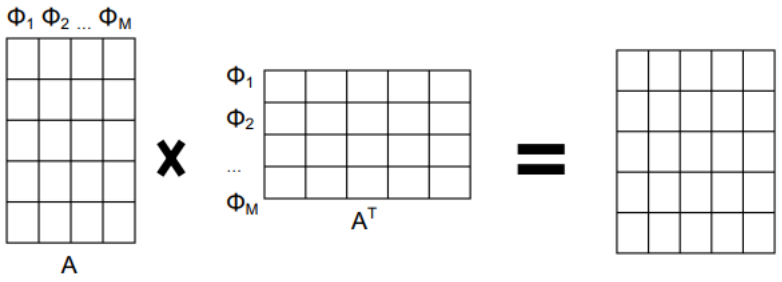

Figura 3.5: Calcularea matricei de covarianță

7. Calcularea valorilor proprii a matricei de covarinață

Valorile proprii se calculează cu următoarea formulă:

$$det(C - \lambda I_n) = 0$$

Unde I_n reprezintă matricea identitae corespunzătoare lui C.

După ce se efecuează calculele, se obțin mai multe valori proprii. Fiecare valoare proprie va avea un vector propriu corespunzător.

8. Calcularea vectorilor proprii

Pentru fiecare valoare proproe obținută anterior, vom aplica formula:

$$C\vec{v} = \lambda\vec{v}$$

Unde $\vec{v}$ este vectorul propriu corespunzător valorii proprii. λ

Vectorii proprii trebuie normalizați astfel încât norma lor să fie egală cu 1.

9. Alegem cei mai buni K vectori proprii

Valoarea K este aleasă arbitrar. Ordonăm descrescător valorile proprii și vectorii lor corespunzători, și îi alegem pe primii K. Fiecare imagine de antrenament poate fi reprezentată folosindu-ne de cei K vectori proprii.

$$\Phi_i = \sum_{j=1}^{K} w_j^i u_j, (w_j^i = u_j^T \Phi_i)$$

u_j = vectorul propriu de la indexul j

Φ_i = vectorul imaginii i - media vectorilor imagine

w_j^i = proiectia imaginii i cu vectorul imagine u_j

Fiecare imagine de antrenament va fi reprezentată folosindu-ne de proiecții.

$$\Omega_i = \begin{vmatrix} w_1^i \\ w_2^i \\ \dots \\ w_3^i \end{vmatrix}$$

Testarea algoritmului Eigenfaces

Pentru a aplica recunoașterea facială pe o imagine necunoscută, întâi o convertim în vector, apoi scădem din el vectorul medie calculat în faza de testare. În cele din urmă calculăm Ω pentru această imagine și căutăm printre imaginile de test, imaginea al cărei Ω_i este cel mai apropiat de imaginea necunoscută, calculând distanța dintre cei doi $\left\|\Omega - \Omega_i\right\|$.

3.4.2 Fisherfaces

O altă soluție pentru problema recunoașterii faciale este o alta metoda statistică denumită Fisherfaces. Această metodă a fost propusă pentru prima dată de Belhumeur, Hespanha și Kriegman în anul 1997, în lucrarea "Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection [BHK97]". Acest algoritm este o variantă modificată a metodei Eigenfaces. Principala diferență între cele doua este că Eigenfaces nu face nici o diferență între două imagini din clase diferite, pe când Fisherfaces folosește imagini etichetate, de exemplu cu numele persoanelor și folosește metoda analizei liniare a discriminantului pentru a diferenția imagini din clase diferite.

Fisherfaces funcționează asemănător cu Eigenfaces. Imaginile sunt convertite în vectori. Spre deosebire de metoda Eigenfaces, metoda Fisherfaces folosește pe lângă vectorul medie total, câte un alt vector medie pentru fiecare clasă. Din fiecare vector se scade vactorul medie al clasei de care aparține.

Pasul următor este calcularea matricei de dispersie dintre clase, notată cu S_b și a matricei de dispersie din clasă, notată cu S_w.

$$S_w = \sum_{i=1}^{c} \sum_{\Gamma_k \in c_i} (\Gamma_k - \Psi_{c_i})(\Gamma_k - \Psi_{c_i})^T$$

$$S_b = \sum_{i=1}^{c} q_i (\Psi_{c_i} - \Psi)(\Psi_{c_i} - \Psi)^T$$

c = numărul de clase
Γ = vector imagine
Ψ = vector medie
Ψ_{c_i} = vector medie al clasei c_i
Vectorii proprii se calculează în felul următor:

$$S_b w_i = \lambda S_w w_i, i = 1, 2, ..m$$

Apoi procesul este la fel ca și în cazul metodei Eigenfaces, proiecția imaginii de test va fi comparată cu proiecțiile imaginilor de antrenament, și clasa imaginii cu distanța cea mai mică va fi predicția algoritmului.

3.4.3 Histograme cu modele binare locale (LBPH)

LBPH este o altă abordare pentru recunoașterea facială diferită de cele discutate până acum. Aceasta nu este o metodă holistică. LBPH este una dintre cele mai simple metode de recunoaștere facială.

În primul rând, imaginea este împărțită în blocuri de 3x3 pixeli. Pe rând, pentru fiecare bloc valoarea centrală este comparată cu valorile adiacente. Dacă valoarea unui vecin este mai mică decât valoarea centrală, acesta se înlocuiește cu 0, altfel se înlocuiește cu 1. Următorul pas, după ce fiecare vecin a fost comparat cu valoarea din mijloc și înlocuit corespunzător, începând de la vecinul din partea stângă, sus, în sensul acelor de ceasornic, vom concatena toți vecinii pentru o forma un număr binar. Numărul binar obținut se transformă într-un număr decimal. Valoarea din mijlocul blocului se înlocuiește cu numărul decimal obținut.

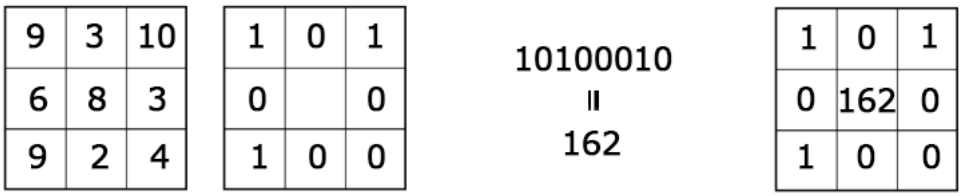

Figura 3.6: Exemplu de aplicare a algoritmului LBPH

După efectuarea acestui proces pe fiecare pătrat 3x3 al imaginii, se extrage histograma imaginii. Pentru a face acest lucru, se contorizează fiecare apariție a valorii din mijloc a fiecărui bloc 3x3. Toate aceste contorizări se rețin într-un vector, astfel obținându-se histograma imaginii.

Pentru a recunoaște o față, se calculează histograma imaginii acestei fețe și se compară cu histogramele imaginilor de antrenament. Cea mai apropiată histograma reprezintă predicția algoritmului.

Ca și în cazul algoritmului HOG, rezultatul acestei metode nu este influențat de variația luminozității imaginilor.

3.4.4 Rețele Neuronale Convoluționale (CNN)

Utilizarea rețelelor neuronale convoluționale reprezintă o altă abordare pentru recunoașerea facială. Conceptul de Rețea neuronală convoluțională a fost introdus de Yann LeCun și Yoshua Benigo în anul 1995 [KMT14].

Rețelele neuronale convoluționale sunt o clasă de rețele neuronale artificiale specializate în analiza imaginilor. Acestea sunt formate din straturi de intrare, straturi de ieșire și mai multe straturi ascunse. Fiecare strat este compus din mai mulți neuroni. Fiecare neuron primește date de intrare, le procesează și întoarce un rezultat. În cazul recunoașterii faciale, primul strat al CNN va primi o imagine a unei persoane, iar ultimul strat va întoarce rezultatul identificării individului.

De obicei rețelele neuronale convoluționale sunt compuse din 3 tipuri de straturi, straturile convoluționale, straturile de pooling și straturile fully connected.

Stratul de convoluție

Stratul de convoluție este elementul de bază al CNN. În cadrul acestui strat se aplică unul sau mai multe filtre asupra imaginii. Filtrele sunt reprezentate de matrici nxn de dimensiuni mici. Fiecare filtru va trece peste fiecare porțiune a imaginii cu un pas stabilit anterior. Dacă suprapunerea ar trebui evitată la compararea imaginii cu filtrul, atunci pasul trebuie să fie ales astfel încât să fie mai mare sau egal decât lungimea filtrului. De exemplu, pentru un filtru 3x3, pasul ar trebui să fie mai mare sau egal cu 3. De asemenea este posibil să adaugăm în jurul imaginii niște pixeli suplimentari cu valoarea 0.

Straturile de convoluție vor avea ca rezultat ceea ce poartă denumirea de feature maps, imagini care conțin trăsăturile caracteristice ale imaginii.

În general, straturile convoluționlale sunt urmate de o funcție de activare. Cele mai folosite funcții de activare sunt funcțiile Sigmoid, Than și ReLU. Funcția Sigmoid primește un număr real pe care îl mapează pe intervalul 0,1. În cazul funcției Than, se face maparea pe intervalul [0,1]. Funcția ReLU (Rectified Linear Unit) a devenit foarte populară în ultimii ani. Aceasta primește un număr real k și calculează funcția f(k) = max(0,k). Cu alte cuvinte întoarce ca rezultat maximul dintre 0 și numărul real.

Stratul de pooling

Stratul de pooling înlocuiește valoarea unei zone din rețea cu o statistică a acesteia. Acest lucru ajută la reducerea spațiului reprezentării ceea ce rezultă în reducerea numărului calculelor necesare. Operația de pooling se aplică pe toate porțiunile reprezentării.

Există mai multe funcții de pooling, dar cea mai populară dintre acestea este max pooling. Aceasta înlocuiește o porțiune a reprezentării cu maximul tuturor valorilor acelei porțiuni.0

Stratul fully connected

Acest strat conectează fiecare neuron dintr-un strat cu fiecare neuron din alt strat. Pentru a recunoaște facial o persoană, vom aplica acești pași asupra imaginii faciale ale acesteia, și vom compara, rezultatul cu rezultele imaginilor de test. Un exemplu de algoritm de recunoaștere facială folosind CNN este implementat în librăria dlib.

3.5 Studiu comparativ al algoritmilor de recunoaștere facială

După cum am observat în subcapitolul anterior, există numeroase metode de recunoaștere facială, fiecare având diferite abordări și rezultate. În continuare vom analiza algoritmii discutați anterior și vom face o comparație din punct de vedere al timpului de execuție și al preciziei acestora. Pentru acest lucru vom rula diferite teste și vom analiza rezultatele acestora.

3.5.1 Testarea algoritmilor de recunoaștere facială

Pentru a compara algoritmii de recunoaștere facială, vor fi rulate mai multe teste, cu date diferite. Pentru fiecare test în parte, datele de testare vor suferi modificări precum numărul indivizilor din baza de date, numărul de imagini de antrenament pentru fiecare individ și luminozitatea imaginilor.

Algoritmii folosiți pentru test vor fi Eigenfaces, Fisherfaces, metoda histogramelor cu modele binare locale (LBPH) și metoda recunoașterii faciale cu ajutorul rețelelor neuronale convoluționale. Pentru a implementa algoritmii Eigenfaces, Fisherfaces și LBPH vom folosi librăria OpenCV, iar pentru CNN vom folosi dlib. Imaginile de antrenament și imaginile de test vor proveni din baza de date Labeled Faces in the Wild.

Vom avea 3 categorii de teste. Pentru prima categorie de test, vom stoca în baza de date imaginile faciale a 5 indivizi diferiți. Toate imaginile acestora vor avea aceeași luminozitate. Pentru a doua categorie de test, vom stoca în baza de date tot imagini faciale pentru 5 indivizi diferiți, dar de această dată imaginile de test ale acestora vor suferi modificări din punct de vedere al luminozității. Pentru a treia categorie, numărul indivizilor pentru care se va efectua testul crește la 10, iar la fel ca și în cazul categoriei precedente, luminozitățile imaginilor de test vor suferi modificări.

Fiecare categorie de test va fi rulată în trei scenarii diferite, respectiv cu câte o imagine de antrenament/individ, 5 imagini de antrenament și 20 de imagini de antrenament.

Pentru fiecare test în parte, vom alege în mod aleator 20 de imagini de test ale indivizilor din baza de date pe care vom aplica algoritmii de recunoaștere facială și vom păstra rezultatele într-un tabel. Rezultatele din tabel reprezintă numărul de identificări corecte ale indivizilor din imagini, numărul de identificări greșite și rata de succes sau acuratețea algoritmului.

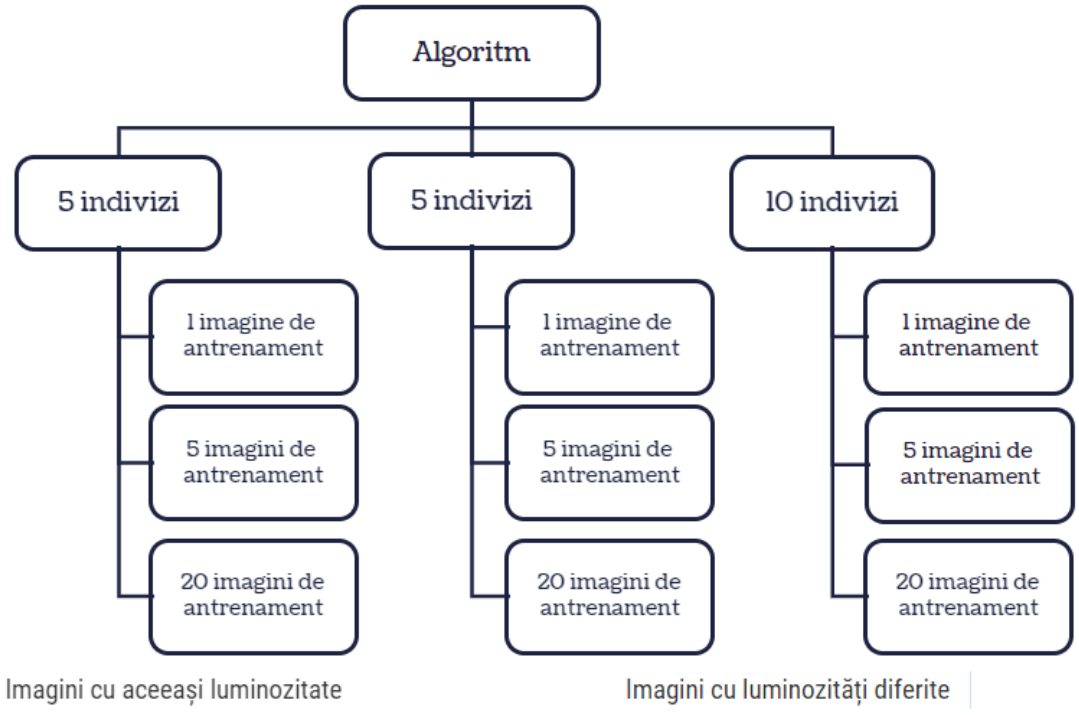

Figura 3.7: Reprezentare a testelor planificate

Pentru rularea testelor au fost îndepliniți următirii pași: În primul rând, s-au ales imaginile de antrenament și imaginile de test ale indivizilor din baza de date Labeled Faces in The Wild. Pentru testele care necesită modificarea luminozității imaginilor de antrenament, s-au selectat aleator câteva imagini pentru fiecare individ. Pe aceste imagini s-au aplicat diferite filtre folosind un program de editare foto. S-au creeat două directoare, test_data și training_data. În ambele directoare, s-a creeat câte un folder pentru fiecare individ. Aceste foldere au fost populate cu imaginile indivizilor.

Următorul pas a fost antrenarea algoritmului. Pentru acest lucru s-au folosit toate imaginile de antrenament din training_data. Ultimul pas al testului a fost recunoașterea facială propriu zisă. S-au ales imagini random din setul de imagini de test și s-a aplicat asupra lor algoritmul de recunoaștere facială. S-a afișat pe ecran predicția algoritmului care reprezintă un nume de persoană și denumirea directorului în care se află imaginia de test, care reprezintă numele real al persoanei din imagine. Numele prezis de algoritm și numele real al persoanei au fost comparate,

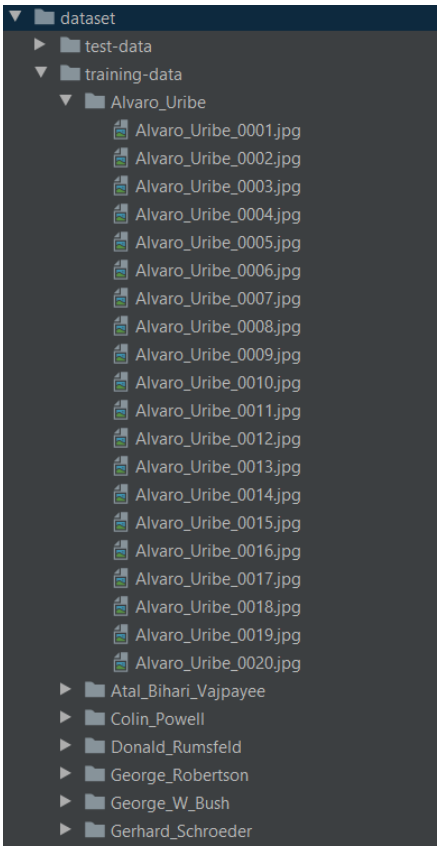

Figura 3.8: Directory tree-ul imaginilor de antrenament și imagiilor de test

iar pentru fiecare algoritm s-a marcat în tabel numărul de predicții corecte, numărul de predicții greșite și s-a calculat acuratețea cu ajutorul formulei:

$$PC/(PC + PG) * 100$$

PC - numărul predicțiilor corecte

PG - numărul predicțiilor greșite

Pentru calcularea timpului necesar rulării algoritmului de recunoaștere facială, s-a salvat întro variabilă timpul curent înainte de rularea algoritmului, iar în altă variabilă timpul curent după rularea algoritmului. Timpul necesar rulării reprezintă diferența dintre dimpul curent după rulare și timpul curent înainte de rulare.

3.5.2 Analiza rezultatelor testelor

Imaginea 3.9 reprezintă rezultatele obținute în prima fază a testului - 5 indivizi cu imaginile de test și imaginile de antrenament provenite din același mediu (cu aceleași luminozități). Tabelul arată rata de identificare corectă a unui individ. După cum putem observa, pentru fiecare algoritm acuratețea crește o dată cu creșterea

	1 imagine de antrenament			5 imagini de antrenament			20 de imagini de antrenament		
Algoritm	Corect	Eroare	Rata	Corect	Eroare	Rata	Corect	Eroare	Rata
Eigenfaces	7	13	35%	9	11	45%	19	1	95%
Fisherfaces	7	13	35%	10	10	50%	19	1	95%
LBPH	10	10	50%	12	8	60%	20	0	100%
Dlib CNN	17	3	85%	20	0	100%	20	0	100%

Figura 3.9: Rezultatele obținute în urma rularii testelor pentru 5 indivizi unde imaginile de antrenament și imaginile de test au aceeași luminozitate

	1 imagine de antrenament			5 imagini de antrenament			20 de imagini de antrenament		
Algoritm	Corect	Eroare	Rata	Corect	Eroare	Rata	Corect	Eroare	Rata
Eigenfaces	7	13	35%	6	14	30%	13	7	65%
Fisherfaces	7	13	35%	8	12	40%	12	8	60%
LBPH	9	11	45%	11	9	55%	9	11	45%
Dlib CNN	16	4	80%	18	2	90%	20	0	100%

Figura 3.10: Rezultatele obținute în urma rularii testelor pentru 5 indivizi unde imaginile de antrenament și imaginile de test au luminozități diferite

	1 imagine de antrenament			5 imagini de antrenament			20 de imagini de antrenament		
Algoritm	Corect	Eroare	Rata	Corect	Eroare	Rata	Corect	Eroare	Rata
Eigenfaces	5	15	25%	6	14	30%	13	7	65%
Fisherfaces	4	16	20%	10	10	50%	18	2	90%
LBPH	8	12	40%	15	5	75%	18	2	90%
Dlib CNN	14	6	70%	17	3	85%	20	0	100%

Figura 3.11: Rezultatele obținute în urma rularii testelor pentru 10 indivizi unde imaginile de antrenament și imaginile de test au luminozități diferite

numărului de imagini de antrenament. Pentru 20 de imagini de antrenament, toți algoritmii au identificat aproape perfect indivizii. Pentru o singură imagine de antrenament, algoritmii statistici Eigenfaces și Fisherfaces au avut cele mai slabe rezultate. Cele mai bune rezultate au fost obținute de CNN cu Dlib. Această metodă a avut o acuratețe de 85% chiar și cu o singură imagine de antrenament.

În a doua fază testele s-au rulat pentru 5 indivizi cu imaginile de test și imaginile de antrenament provenite din medii diferite 3.10. În acest caz, acuratețea tuturor algoritmilor a scăzut față de testul trecut. Din nou, cele mai bune rezultate au fost obținute de CNN. Pentru o imagine de antrenament și 5 imagini de antrenament, algoritmul LBPH a avut a doua cea mai bună acuratețe, dar pentru 20 de imagini de antrenament, acuratețea acestui algoritm a fost cea mai mică.

În a treia fază a testului - 10 indivizi cu imagini de antrenament și imagini de test provenite din medii diferite 3.11, acuratețea algoritmilor pentru o imagine de antrenament a fost cea mai mică dintre toate testele. Față de testul trecut, în afară de algoritmul Eigenfaces, acuratețea celorlalți algoritmi s-a îmbunătățit pentru 5 și 20 de imagini de antrenament.

Din punct de vedere al timpului de execuție pentru fiecare algoritm s-au obținut rezultate bune. Toți algoritmii au reușit să identifice persoanele din imagine în maximum 2 secunde.

În concluzie, acuratețea tuturor algoritmilor a fost afectată în mod negativ cand imaginile de test și imaginile de antrenament au provenit din medii diferite. Pentru un număr mic de imagini de antrenament, algoritmii Eigenfaces și Fisherfaces au obținut cele mai slabe rezultate, dar pentru cel puțin 20 de imagini de antrenament, acuratețea tuturor algoritmilor a crescut considerabil. Cele mai bune rezultate au fost obținute de CNN. Această metodă a avut o acuratețe de cel puțin 70%, pentru o imagine de antrenament. Pentru 2 imagini de antrenament, acuratețea a crescut pentru toate testele. Pentru 20 de imagini de antrenament, acest algoritm a identificat perfect toate imaginile, având o acuratețe de 100%.

Capitolul 4

IntAttend - gestionarea prezențelor cu ajutorul recunoașterii faciale

IntAttend este un sistem inteligent de gestionare a prezențelor cu ajutorul recunoașterii faciale. Scopul acestui sistem este de a reduce erorile care vin o dată cu metodele convenționale de gestionare a prezențelor precum deteriorarea sau pierderea datelor sau timpul îndelungat de lucru.

IntAttend funcționează după următorul scenariu: un administrator se loghează în aplicație și activează funcția de marcare a prezențelor cu ajutorul recunoașterii faciale. Administratorul poziționează dispozitivul Android care rulează IntAttend la intrarea în clădire. Utilizatorii își marchează intrarea sau ieșirea din clădire pe dispozitivul Android plasat de administrator. Pentru a-și verifica prezențele, utilizatorii pot să instaleze aplicația IntAttend pe dispozitivul Android personal. Utilizatorii obișnuiți nu au acces la funcția de marcare a prezențelor pe dispozitivele personale. Administratorul poate să vizualizeze toți angajații și prezențele acestora.

4.1 Motivație

În multe organizații și instituții, prezența este o normă foarte importantă care este folosită în diferite scopuri. Printre aceste scopuri se enumeră menținerea disciplinei în organizații și școli, oferirea educației de calitate în școli, pedepsirea abaterilor de la standardele cerute, calcularea salariilor angajaților.

Există numeroase metode de gestionare a prezențelor, cea mai folosită fiind înregistrarea manuală a numelor elevilor sau a angajaților pe foi de prezență. Pentru un număr mare de elevi sau angajați, această metodă necesită mult timp, iar foile de prezență se pot pierde sau deteriora în orice moment. Deseori este nevoie de numărul de prezențe al unei persoane, de exemplu pentru a stabili dacă un student este eligibil pentru a intra în examen. Filtrarea și calcularea numărului de prezențe de pe foile

este o sarcină obositoare.

O altă metodă populară de gestionare a prezențelor este folosirea cartelelor RFID. Această metodă rezolvă unele probleme ale marcării manuale a prezențelor, dar cu toate acestea nu este cea mai bună metodă. Cartelele de pontaj se pot deteriora sau pierde, acest lucru necesitând înlocuirea lor. O altă mare problemă a acestei metode este imposibilitatea de a se verifica dacă persoana care folosește cartela este chiar posesoarea acesteia.

Toate aceste probleme ar putea fi rezolvate cu ajutorul sistemelor automate de gestionare a prezențelor. Aceste sisteme folosesc date biometrice pentru a verifica identitatea persoanelor și marchează prezențele în mod automat. Printre metodele automate de gestionare a prezențelor se enumeră identificarea amprentei, recunoașterea facială, recunoașterea irisului, scanarea geometriei mâinii.

Toate aceste metode sunt mult mai eficiente decât metodele convenționale. Prezențele sunt înregistrate în mod automat într-o bază de date, datele putând fi procesate mult mai usor mai departe. Riscul de a pierde datele referitoare la prezențe este eliminat, iar timpul necesar procesului de marcare a prezențelor este redus seminficativ. Sistemul IntAttend reprezintă o soluție de gestionare a prezențelor cu ajutorul recunoașterii faciale.

4.2 Arhitectură

În imaginea de mai jos este prezentată arhitectura sistemului de gestionare a prezențelor, IntAttend 4.1.

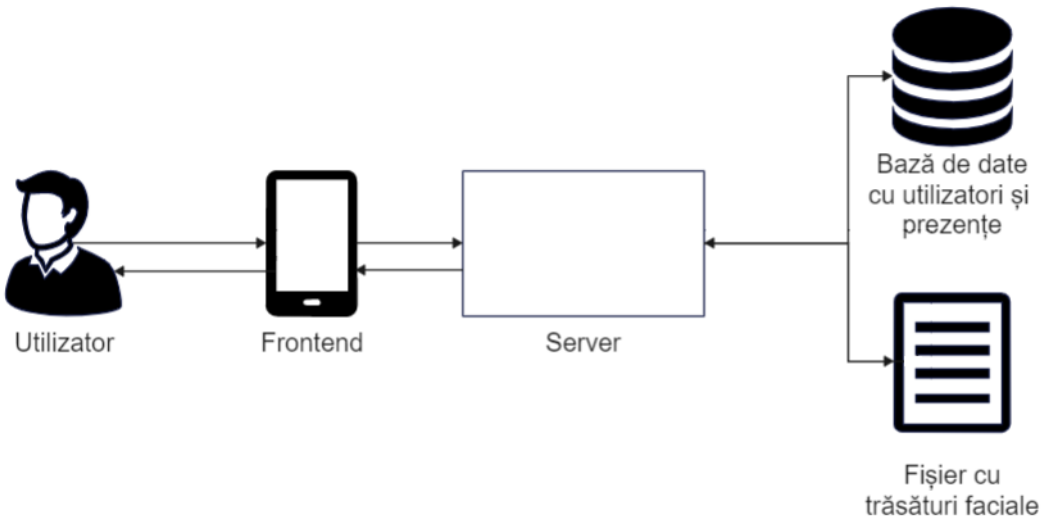

Figura 4.1: Arhitectura sistemului IntAttend

Sistemul IntAttend este reprezentat de o arhitectură compusă din frontend, server, bază de date cu utilizatori și prezențele acestora și un fișier cu trăsăturile faciale ale fiecărui utilizator.

Utliziatorul comunică cu frontend-ul sistemului prin intermediul unui dispozitiv android. Acesta poate să introducă sau să ceară date prin intermediul dispozitivului android.

Partea de frontend a aplicației reprezintă interfața grafigă a sistemului. GUI-ul a fost dezvoltat în limbajul Flutter, după modelul MVC, astfel încât să ruleze pe dispozitive Android. Frontent-ul sistemului are ca rol colectarea de informații de la utilizator și transmiterea lor către server și livrarea informațiilor de interes de la server către client.

Utilizatorul are posibilitatea de a se înregistra în aplicație. Pentru a face acest lucru este nevoie de introducerea numelui și prenumelui, a adresei de email, a unei parole și a unei imagini faciale. Daele colectate sunt validate, și în caz de eroare sunt afișate mesaje corespunzătoare. Pentru afișarea mesajelor de eroare se folosește librăria Flushbar din Flutter. Înainte ca parola să fie transmisă către server, aceasta este criptată cu ajutorul algoritmului SHA256. Pentru capturarea imaginilor faciale este folosită librăria Camera.

La logarea în aplicație, în funcție de rolul utilizatorului, acesta poate obține sau transmite diferite informații de la sau către server. Există doua roluri diferite pentru utilizator USER și ADMIN. Un USER poate să vizualizeze datele legate de prezențele acestuia. Aceste date sunt, data și ora și tipul înregistrării, CHECK IN sau CHECK OUT.

Un ADMIN, pe lângă vizlualizarea prezențelor sale, poate vizualiza prezențele fiecărui angajat. Pentru îregistrarea prezențelor există un meniu special, pe care doar ADMIN-ul îl poate accesa.

Pentru comunicarea cu serverul este folosită librăria http. Datele primite și trimise către server sunt reprezentate prin formatul JSON. Pentru interpretarea răspunsurilor de la server există o clasă specială, Response. Informațiile acestei clase sunt tipul răspunsului și datele adiționale primite de la server. Aceste date adiționale pot fi inclusiv nule. Datele adiționale pot fi reprezentate în format CSV. Pentru aceste cazuri sunt folosite funcții adiționale care transformă datele CSV în obiectele corespunzătoare.

Serverul sistemului IntAttend a fost dezvoltat în limbajul de programare Python. Serverul este alcătuit din mai multe componente diferite. Asigurarea comunicării cu partea de prontend se realizează printr-un RESTful API dezvoltat în Flusk. Endpoint-urile acestui API sunt, sign-up, sign-in, check-in, check-out, attendance, users și delete-employee.

Pentru procesarea cererilor făcute de către client, serverul folosește un service.

Acesta este responsabil cu asigurarea comunicării tuturor componentelor din server, respectiv cu validatorul, repository-urile și componenta de recunoaștere facială. După procesarea datelor, service-ul creează niște răspunsuri pe care le trimite către API, componenta responsabilă cu comunicarea client-server. Aceste răspunsuri sunt convertite în format JSON și transmise către client unde urmează să fie decodate.

Componenta de recunoaștere facială se bazează pe două librării importante, OpenCV și FaceRecognition. Librăria OpenCV este folosită pentru citirea și procesarea imaginilor. Librăria FaceRecognition este folosită pentru a implementa algoritmul de recunoaștere facială Dlib. Această metodă presupune folosirea unei rețele neuronale convoluționale și a fost discutată într-un capitol anterior.

Comunicarea cu baza de date este implementată în componenta Repository a serverului. Aceasta folosește ORM-ul SQLAlchemy pentru implementarea operațiilor CRUD. Citirea și scrierea în fișierul cu trăsături faciale se realizează în componenta de recunoaștere facială.

Utilizatorii și prezențele acestora sunt stocate într-o bază de date relațională MySQL. Baza de date este compusă din două tabela, attendance și user. Pentru stocarea trăsăturilor faciale ale fiecărui utilizator s-a folosit un fișier csv cu trei coloane. Prima coloană reprezintă id-ul utilizatorului, a doua coloană reprezintă numele și prenumele utilizatorului, iar ultima coloană conține un vector cu trăsăturile faciale ale utilizatorului.

4.3 Funcționalități

Printre funcționalitățile sistemului IntAttend, se enumeră următoarele:

1. Logare - aplicația permite logarea utilizatorilor înregistrați prin introducerea adresei de email și a parolei. Aceste câmpuri sunt introduse în două casete de text specifice fiecăreia. Parola va apărea sub formă de steluțe. Logarea se efectuează prin apăsarea buonului Login. În cazul în care adresa de email și/sau parola nu sunt introduse, va apărea mesajul de eroare ”Please enter your email and password!”. În cazul în care adresa de email și/sau parola sunt greșite va apărea mesajul de eroare ”Invalid email or password!”. În cazul în care utilizatorul se loghează cu succes, acesta va fi redirecționat pe o pagină care va conține un meniu. Acest meniu va fi diferit în funcție de rolul utilizatorului, ADMIN sau USER.

2. Înregistrare - această funcție poate fi acesată prin apăsarea butonului Register din pagina de logare. Prin această funcționalitate, IntAttend permite înregistrarea utilizatorilor noi. Acest lucru se face prin introducerea în casete text a numelui și prenumelui, adresei de email, parolei și a confirmării parolei și prin capturarea unei imagini faciale prin acționarea butonului Face Photo. În cazul în care una dintre aceste informații lipsește se va afișa un mesaj de eroare. Numele trebuie să fie de

forma ”Nume Prenume”, în caz contrar se va afișa un mesaj de eroare. Adresa de email trebuie să fie validă, iar parola tebuie să conțină cel puțin 8 caractere și să fie identică cu confirmarea parolei. Procesul de înregistrare se finalizează prin acționarea butonului Register. Dacă există deja un cont cu aceeași adresă de email, utilizatorul va fi atenționat. Există și cazul când pot apărea erori necunoscute de la server. în acest caz, utilizatorul va fi atenționant să reîncerce peste câteva momente.

3. Funcția de vizualizare a prezențelor personale poate fi accesată de orice utilizator prin apăsarea butonului My Working Time din meniu 4.2. Acest buton deschide o nouă fereastră unde utilizatorul poate să vadă toate prezențele sale. Pentru vizualizarea prezențelor, utilizatorul alege data din calendar. Toate evenimentele de tipul CHECK IN sau CHECK OUT din acea dată vor apărea sub formă de tabel. Pentru fiecare eveniment se afișează tipul, data și ora.

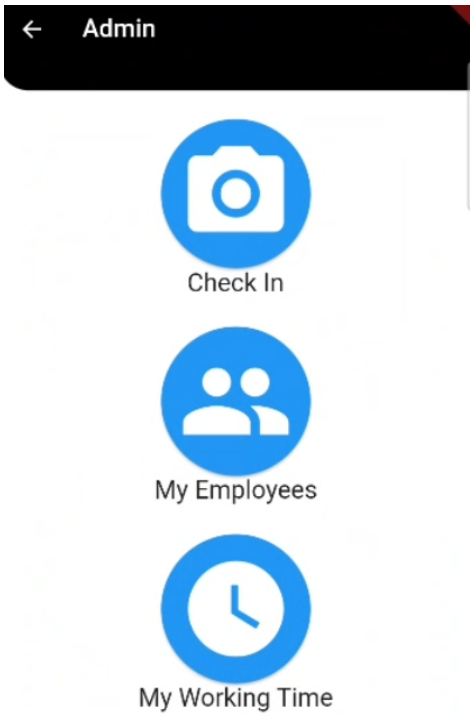

Figura 4.2: Meniul specific utilizatorilor de tip ADMIN

4. Vizualizarea listei angajaților este o funcție specifică admin-ului. Aceasta poate fi accesată prin apăsarea butonului My Employees. Lista angajaților va apărea sub formă de tabel, iar pentru fiecare angajat se va afișa numele complet și adresa de email. În dreapta fiecărui angajat va apărea o iconiță cu un coș de gunoi.

5. Vizualizarea prezențelor unui angajat este o altă funcție specifică utilizatorilor de tip ADMIN. Pentru a vizualiza prezențele unui angajat, adminul selectează acel angajat din lista de angajați, apoi se va deschide fereastra de vizualizare a

prezențelor pentru angajatul respectiv. Această fereastră are același comportament ca și funcționalitatea discutată la punctul 4.

6. Ștergerea unui angajat - această funcție poate fi realizată doar de un admin. Admin-ul deschide fereastra de vizualizare a listei angajaților. În dreptul fiecărui angajat va apărea o iconiță cu un coș de gunoi.

7. Funcția de marcare a prezențelor poate fi activată doar de un admin. Administratorul va activa această funcție și va plasa dispozitivul android într-un loc de interes, de exemplu la intrarea în clădire, astfel încât utilizatorii să își marcheze sosirea și plecarea. Această funcție permite utilizatorului să aleagă tipul de eveniment prin butoane specifice, respectiv CHECK IN care reprezintă sosirea și CHECK OUT care reprezintă plecarea. După selectarea tipului de eveniment, utilizatorul se va poziționa asfel încât fața acestuia să se vadă cât mai clar și va apăsa butonul de capturare a imaginii. Sistemul va încerca să identifice persoana din imagine. Dacă persoana este identificată cu succes, se introduce în baza de date o înregistrare pentru prezență și se afisează un mesaj cu numele utilizatorului. În cazul în care persoana nu se poate identifica, se afisează un mesaj de eroare.

4.4 Implementare

Sistemul IntAttend este alcătuit din client și server. Cele două componente au fost implementate separat, în limbaje de programare diferite. Următoarea diagramă de clase reprezintă structura serverului sistemului 4.3.

Serverul aplicației se rulează din main. În main se instanțiază clasa Recognition, folosită pentru recunoașterea facială. Se creează două instanțe ale clasei DatabaseRepository, respectiv userRepository și attendanceRepository, folosite pentru salvarea în baza de date a utilizatorilor și ale prezențelor acestora. De asemenea se instanțiază clasele UserValidator, folosiă pentru validarea utilizatorilor, Service și NetworkController, folosită pentru comunicarea cu frontend-ul sistemului. După instanțierea tuturor claselor necesare, se execută funcția run din NetworkController. Aceasta este funcția propriu zisă de rulare a serverului.

Clasa NetworkController este responsabilă pentru comunicarea între client și server. Această clasă este implementată cu ajutorul librăriei Flusk. Serverul are nevoie de un canal compus din IP și PORT pentru a putea primi cererile trimise de client. Aceste informații sunt preluate de NetworkController din fișierul config. În interiorul clasei sunt declarate endpoint-urile care pot fi apelate de client. Aceste endpoint-uri sunt sign-up, sign-in, check-in, check-out, attendance, users și delete-employee. Fiecare endpoint funcționeaza asemănător, primește de la client un request de tipul GET sau POST împreună cu un set de date, apelează metoda corespunzătoare din Service și primește un răspuns de la acesta, întoarce către client

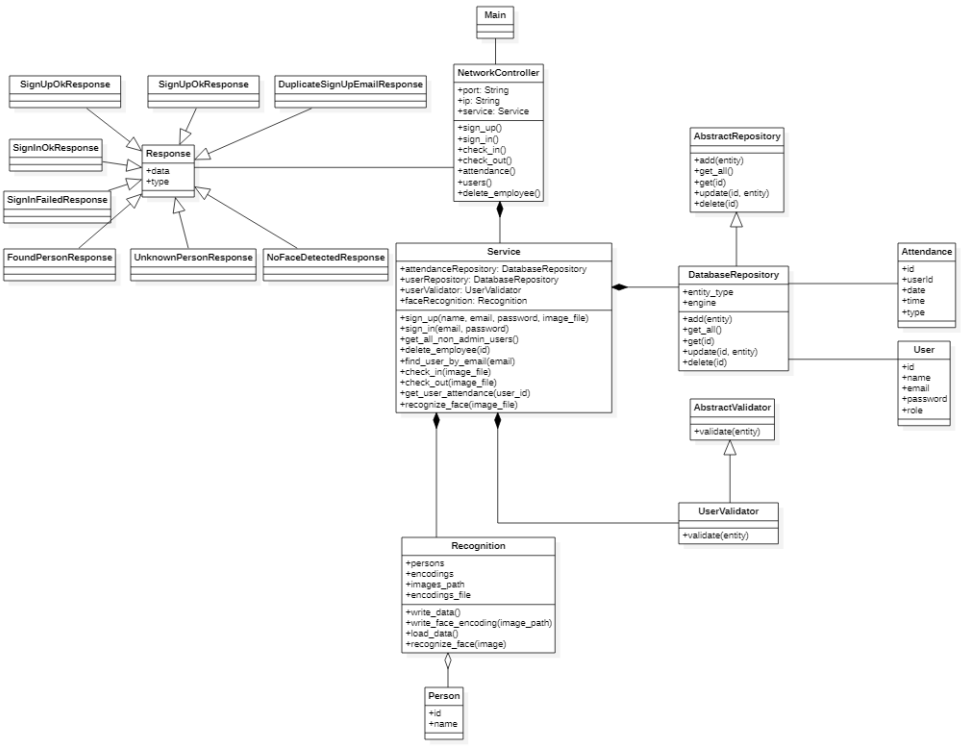

Figura 4.3: Diagrama de clase corespunzătoare serverului

un răspuns de tip JSON.

Endpoint-ul sign-up este folosit pentru înregistrarea noilor utilizatori. Acesta primește de la client un request de tip POST care conține numele, emailul, parola și imaginea facială a noului utilizator, apelează metoda cu același nume din Service și întoarce un răspuns clientului.

Sign-in este apelat pentru logarea utilizatorilor. Acest endpoint primește request-uri POST împreună cu emailul și parola utilizatorului. Metoda din Service poartă același nume.

Check-in și check-out sunt folosite pentru marcarea sosirii (CHECK IN) și părăsirii (CHECK OUT) clădirii. Aceste metode funcționează cu request-uri de tip POST care conțin imaginea facială a angajatului.

Users primește un request GET și întoarce toți utilizatorii care nu sunt administratori.

Delete-employee primește un request POST împreună cu id-ul utilizatorului. Este folosit pentru a șterge un utilizator. Acest endpoint nu întoarce nici un răspuns.

Clasa Service execută cererile trimise de client prin endpoint-uri și întoarce răspunsuri către NetworkController. Există mai multe tipuri de răspuns posibile. Toate aceste răspunsuri moștenesc clasa de bază Response. Un Response conține tipul răspunsului și datele adiționale corespunzătoare. Metoda sign_up primește ca input numele, adresa de email, parola și o imagine facială a unui utilizator nou. Cu aceste câmpuri

creează o instanță a clasei User și validează această instanță cu ajutorul clasei UserValidator. Dacă utilizatorul este valid, încearcă adăugarea acestuia în baza de date cu ajutorul instanței userRepository. În cazul în care există deja un utilizator cu aceeași adresă de email se întoarce un răspuns de tipul DuplicateSignUpEmailResponse. În cazul în care apar orice alt fel de erori se returnează un SignUpFailedResponse. În cazul în care utilizatorul este adăugat cu succes la baza de date se trimite către componenta Recognition imaginea facială a utilizatorului și se returnează SignUpOkResponse.

Metoda sign_in primește o adresă de email și o parolă, caută în baza de date utilizatorul carre corespunde acestor date. În cazul în care se găsește un utilizator, se returnează un răspuns de tipul SignInOkResponse, iar în caz negativ SignInFailedResponse.

Metoda get_all_non_admin_users extrage din baza de date toți utilizatorii cu ajutorul metodei get_all din userRepository și îi returnează doar pe cei care au rolul USER. Metoda delete_employee primește id-ul unui utilizator și apelează metoda delete din userRepository. După ștergerea utilizatorului, parcurge toate prezențele din baza de date și le șterge pe toate acelea care au userId-ul utilizatorului șters. find_user_by_email apelează metoda get_all din userRepository și caută utilizatorul cu emailul dat.

Metodele check_in și check_out funcționează la fel, primesc imaginea facială a unei persoane și apelează metoda recognize_face din clasa Recognition. Dacă persoana din imagine este recunoscută se creează un nou obiect de tipul Attendance cu userId-ul persoanei din imagine, data și ora curentă și tipul CHECK IN sau CHECK OUT și se adaugă obiectul Attendance în baza de date prin intermediul attendanceRepository. Metoda returnează răspunsurile primite de la metoda din Recognition.

Metoda get_user_attendance primește id-ul unui utilizator, și returnează toate prezențele acelui utilizator.

Clasa DatabaseRepository moșteneste clasa Repository. Aceasta este o clasă generică implementată cu ajutorul ORM-ului SQLAlchemy, care primește tipul unei entități pentru care se vor efectua operații CRUD. Celălalt câmp al clasei, pe lângă tipul entității este un engine necesar pentru asigurarea legăturii cu baza de date. Pentru creearea engine-ului este necesar un connection string, care conține dialectul, driver-ul, numele de utilizator, parola, adresa ip și portul și denumirea bazei de date. Toate aceste informații se află în fișierul config.

Baza de date a fost creeată cu ajutorul MySQL și conține două tabele, user și attendance. Tabela user este alcătuită din 5 coloane, id - INT, Primary Key, Not Null, Unique, Auto Incremental; name - VARCHAR(45), Not Null; email - VARCHAR(100), Not Null, Unique; password - VARCHAR(255), Not Null; role - VARCHAR(45), Not Null. Tabela attendance are următoarele coloane: id - INT, Primary

Key, Not Null, Unique, Auto Incremental; userId - INT, Not Null; date - VARCHAR(100), Not Null; time - VARCHAR(45), Not Null; type - VARCHAR(25), Not Null.

În clasa Recognition se întâmplă procesul de recunoaștere facială. Metodele din această clasă sunt implementate cu ajutorul librăriilor OpenCv - folosită pentru procesarea imaginilor și Dlib - folosită pentru recunoașterea facială cu ajutorul unei rețele neuronale convoluționale. Câmpurile acestei metode sunt un vector de persoane, un vector cu codările trăsăturilor faciale ale fiecărei persoane, locația imaginilor faciale și locația fișierului cu trăsăturile faciale.

Trăsăturile faciale sunt salvate într-un fișier de tipt txt. Fișierul este organizat pe trei coloane, despărțite prin virgulă. Coloanele reprezintă id-ul persoanei, numele și prenumele presoanei și un vector cu trăsături faciale.

În momentul initializării clasei, se execută metoda load_data. Această metodă extrage fiecare rând din fișierul txt, creează cu ajutorul id-ului și numele persoanei instanțe ale clasei Person pe care le introduce în vectorul de persoane și adaugă trăsăturile fiecărei persoane în vectorul de trăsături.

Metoda write_face_encoding primește locația unei imagini. Cu ajutorul librăriei OpenCV citește și convertește imaginea în format RGB. Cu ajutorul librăriei FaceRecognition, care implementează recunoașterea facială din Dlib, convertește imaginea într-un vector de trăsături. În cele din urmă, scrie în fișierul de trăsături faciale codarea imaginii respective, precum și numele și id-ul persoanei.

Ultima metodă, recognize_face efectuează procesul de recunoaștere facială. Aceasta funcționează în felul următor, procesează imaginea cu ajutorul librăriei OpenCv, cu ajutorul librăriei FaceRecognition detectează fața. Dacă algoritmul nu reușește să identifice nici o față, returnează răspunsul NoFaceDetectedResponse. În continuare, codează imaginea cușsi compară această codare cu codările din fișierul de trăsături faciale. Algoritmul consideră cea mai bună potrivire ca fiind persoana din imagine. Dacă algoritmul reușește să identifice o persoană returnează un răspuns de tipul FoundPersonResponse, altfel returnează UnknownPersonResponse.

Cea de-a doua componentă a sistemului IntAttend este clientul implementat în limbajul Flutter după modelul MVC. Această componentă este împărțită în mai multe ecrane, fiecare reprezentând o pagină din interfața grafică a aplicației. WelcomeScreen - ecranul de pornire cu logo-ul IntAttend, AdminScreen reprezintă meniul utilizatorilor de tip ADMIN, UserScreen, meniul utilizatorilo de tip USER, CameraScreen, ecranul unde are loc fotografierea utilzatorilor pentru marcarea prezențelor, LoginScreen - pagina de logare, RegisterScreen - pagina de înregistrare a noilor utilizatori, RegisterPhotoScreen - ecranul unde se realizează fotografia noilor utilizatori, MyWorkingTimeScreen - pagina unde sunt afișate prezențele unui utilizator, MyEmployeesScreen - pagina unde sunt afișați toți angajații.

```python
def recognize_face(self, image):
    logging.info("Trying to recognize face")
    rgb_image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    face_location = face_recognition.face_locations(rgb_image)
    if len(face_location) == 0:
        logging.info("No face detected")
        return NoFaceDetectedResponse()

    face_encoding = face_recognition.face_encodings(rgb_image, face_location)[0]
    matches = face_recognition.compare_faces(self.encodings, face_encoding)
    face_distances = face_recognition.face_distance(self.encodings, face_encoding)
    best_match_index = np.argmin(face_distances)
    if matches[best_match_index]:
        logging.info("Found person: {}".format(self.persons[best_match_index]))
        return FoundPersonResponse(self.persons[best_match_index])
    logging.info("Unknown person")
    return UnknownPersonResponse()
```

Figura 4.4: Algoritmul de recunoaștere facială

Fiecare dintre acest ecrane a fost implementat cu ajutorul StatefulWidget-urilor și au în componență mai multe elemente numite Widget-uri. Ecranul de logare conține câte un LoginTextInputWidget folosit pentru introducerea datelor de logare, email și parolă. Butonul de logare este implementat în LoginButtonWidget, iar butonul text care duce la ecranul de înregistrare este implementat în NoAccountWidget. Ecranul de înregistrare are în componență 4 LoginTextInputWidget-uri, pentru nume, email, parolă și confirmarea parolei. Pentru capturarea imaginii faciale a noului uilizator, se folosește TakePhotoWidget, iar butonul text care duce la pagina de logare este implementat în AlreadyHaveAccountWidget.

Fiecare buton din meniul de navigare corespunde câte unui widget. Butonul de navigare la fereastra de marcare a prezențelor este implementat în CameraNavigationWidget. Pentru a naviga la ecranul MyEmployeesScreen, se folosește butonul implementat în MyEmployeesNavigationWidget. Butonul de navigare la ecranul de vizaulizare a prezențelor este implementat în MyWorkingTimeNavigationWidget.

Ecranul de marcare a prezențelor este alcătui din alte două widget-uri. Pentru previzualizarea imaginii camerei foto se folosește CameraPerviewWidget, iar pentru realizarea fotografiei este folosit TakePhotoWidget. Acest widget cuprinde trei butoane, două butoane pentru schimbarea tipului prezenței, respectiv CHECK IN și CHECK OUT și butonul folosit pentru realizarea fotografiei.

Pentru afișarea datelor în ecranele MyWorkingTimeScreen și MyEmployeesScreen, respectiv prezențele și lista angajaților, se folosesc liste de tipul ListView.builder. Pentru a ne asigura că datele afișate sunt actualizate în conformitate cu datele din baza de date, aceste liste au fost puse în interiorul unor structuri de tipul StreamBuilder. StreamBuilder trimite periodic cereri către server pentru a actualiza datele.

Fiecare ecran corespunde câte unui controller. Pe lângă controllere-le de ecran mai există și un NetworkController folosit pentru asigurarea comunicării între client și server. Trimiterea cererilor și primirea răspunsurilor se realizează cu ajutorul

librăriei http di Flutter. Fiecare metodă din NetworkController corespunde și poartă aceeași denumire cu căte un endpoint din server. Fiecare metodă este implementată în același fel, primește datele de la controllere-le de ecran, trimite un request către server și primește un răspund sub formă de JSON. Pentru interpretarea datelor de tip JSON se folosește clasa Response. Această clasă cuprinde tipul răspunsului și datele adiționale ale răspunsului. Tipurile de răspuns sunt la fel ca și cele din server.

Un controller de ecran este CameraController cu metodele checkIn și checkOut. Aceste metode sunt folosite pentru trimiterea către NetworkController a imaginilor faciale din ecranul CameraScreen, apoi în funcție de tipul răspunsului primit, afișează un mesaj. Fiecare mesaj afișat în aplicație are căte o metodă specifică. Mesajele au fost implementate cu ajutorul librăriei Flushbar din Flutter.

Un alt controller corespunzător ecranului de logare este LoginController. Aici se realizează trimiterea datelor de logare și de înregistrare către NetworkController, validarea datelor de înregistrare, cu ajutorul clasei UserValidator, criptarea parolei cu ajutorul algoritmului SHA256 și afișarea mesajelor conform răspunsurilor primite de la server.

Controllere-le EmployeesController și WorkingTimeController sunt folosite pentru a prelua de la NetworkController datele necesare, prezențele și angajații. Listele de prezențe și de angajați sunt transmise de către server în formatul CSV. Pentru a mapa prezențele din tipul CSV în Attendance este folosită metoda AttendanceFromCVS. Pentru maparea angajaților avem metoda userFromCSV.

Pentru asigurarea conexiunii cu serverul avem nevoie de adresa IP și de port. Acestea pot fi configurate în fișierul config.dart.

4.5 Testare

Pentru testarea sistemului IntAttend s-a folosit atât testarea automată cât și testarea manuală. Testarea automată s-a realizazt folosind metoda Unit Testing cu ajutorul librăriei unittest din Python. Această formă de testare s-a aplicat doar asupra backend-ului sistemului. Componentele testate cu această metodă au fost DatabaseRepository, UserValidator și Service. Pentru a testa metodele acestor clase, pentru fiecare s-a creat câte o clasă specifică de test, iar pentru fiecare metodă s-a creat câte o funcție care să o testeze. Pentru fiecare metodă testată, au fost folosite mai multe date de test astfel încăt să fie acoperite toate rezultatele posibile. Toate testele au fost rulate cu succes din fișierul tests_main.py.

Pentru testarea manuală, mai mulți utilizatori au fost înregistrați în aplicație, atăt cu rolul de USER cât și cu rolul de ADMIN. Aplicația a fost folosită în mod normal, analizând modul de funcționare al tuturor funcțiilor. Toate căsuțele de text

au fost testate prin introducerea mai multor date corecte sau greșite sau neintroducerea datelor. De fiecare dată aplicația s-a comportat normal și s-au afișat mesajele corespunzătoare. Au fost testate toate butoanele aplicației și navigarea între pagini. Pentru testarea componentei de marcare a prezențelor cu ajutorul recunoașterii faciale, au fost înregistrați în aplicație mai mulți utilizatori cu imagini faciale luate din baza de date Labeled Faces in the Wild. S-a încercat marcarea prezențelor acestor utilizatori folosind alte imagini ale acelorași persoane din baza de date Labeled Faces in the Wild. De fiecare dată persoanele din imagini au fost identificate cu succes și s-au introdus în baza de date înregistrări corespunzătoare prezenței acestora.

Capitolul 5

Concluzii

Gestionarea prezențelor poate fi o sarcină dificilă, care necesită mult timp și care implică un risc de eroare, mai ales în cazul gestionării prezențelor unui volum mare de persoane. Există multe soluții pentru marcarea prezențelor, cele mai comune fiind marcarea manuală în tabele sau folosirea cardurilor RFID.

Una dintre cele mai bune soluții pentru gestionarea prezențelor este folosirea sistemelor automate care folosesc date biometrice precum imaginea facială. Folosirea recunoașterii faciale pentru gestionarea prezențelor elimină problemele metodelor obișnuite, reduce timpul necesar marcării prezențelor și face prelucrarea datelor mai ușoară.

Există mai multe metode pentru recunoașterea facială. Una dintre cele mai eficiente soluții atât din punct de vedere al acurateței cât și al timpului de execuție este folosirea algoritmilor bazați pe rețele neuronale convoluționale, de exemplu Dlib. Această metodă a fost implementată cu succes în sistemul IntAttend și a obținut rezultate deosebite.

Bibliografie

[BHK97] Peter N Belhumeur, Joao P Hespanha, and David J Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 19(7):711, 1997.

[Del17] Nicolas Delbiaggio. A comparison of facial recognition's algorithms. 2017. Accesat 9 mai 2022.

[DT05] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.

[FAC20] A brief history of facial recognition. `https://www.nec.co.nz/market-leadership/publications-media/a-brief-history-of-facial-recognition`, 2020. Accesat 6 Mai 2022.

[HL01] Erik Hjelmås and Boon Kee Low. Face detection: A survey. *Computer Vision and Image Understanding*, 83(3):236–274, 2001. Accesat 8 Mai 2022.

[KMT14] Hurieh Khalajzadeh, Mohammad Mansouri, and Mohammad Teshnehlab. Face recognition using convolutional neural network and simple logistic classifier. 2014.

[SAAW21] Sovantharith Seng, Mahdi Nasrullah Al-Ameen, and Matthew Wright. A first look into users' perceptions of facial recognition in the physical world. *Computers Security*, 105:102227, 2021. Accesat 5 Mai 2022.

[TP91] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86, 1991.

[VJ01] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. Ieee, 2001.