**APLICAȚIE DE RECUNOAȘTERE FACIALĂ, CORPORALĂ ȘI DE OBIECTE**

**Candidat: Casian-Ovidiu BORCEA**

**Coordonator științific: Conf.dr.ing. Loredana-Mihaela STANCIU**

Sesiunea: Iunie 2022

**REZUMAT**

Motivația acestei inițiative a fost dată de faptul că în prezent, inteligența artificială își face tot mai mult loc în viețile de zi cu zi, iar eu am vrut să vad cu ochii mei, potențialul acesteia. Am realizat o aplicație care să îmbine mai multe tehnologii, care se regasesc în aproape orice dispozitiv pe care îl folosim. Astfel, scopul aplicației este de a da lumii o idee despre cat de prezentă este inteligența artificială în viața noastră.

Aplicația este proiectată pentru a îndeplini trei funcționalități, toate fiind bazate pe conceptul de vedere computerizată. Prima și cea mai de bază funcționalitate este abilitatea de recunoaștere facială, rudimentară pentru nivelul lucrării, dar eficientă. Totul începe cu a popula folderul persoanelor „cunoscute”, acesta constă într-un folder în care se află imagini cu toate persoanele pe care vrem ca programul să le recunoască, numărul persoanelor este constrâns de timpul de procesare. Odata aflate în folder, cu ajutorul camerei web programul recunoaște cu succes orice persoană din folder. Folosind recunoașterea facială, o notificare alertă este trimisă prin windows la detectarea unei persoane necunoscute sau considerate neautorizată. A doua funcționalitate o reprezintă recunoașterea de obiecte bazată pe predicțiile unei rețele neuronale și pe un model antrenat, cu posibilitate de învațare infinită, aceasta funcționalitate este menită a explora posibilitățile nesfârsite ale inteligenței artificiale dar și le da un scop prin a alerta utilizatorul la detecția obiectelor periculoase. A treia funcționalitate, înclinând și mai mult balanța spre securitate, face capabilă captarea video, pentru a avea o noțiune mai clară despre cine s-a aflat în fața camerei, utilitatea sa constă în conceptul de „cameră de securitate”, aceasta este programată pentru a simula comportamentul unei camere de supraveghere, care poate fi instalată la intrarea unei casei, sau la un ATM.

Aplicația a fost realizată cu ajutorul limbajului Python, împreună cu OpenCV, o bibliotecă, care conține toate funcțiile necesare pentru a implementa soluția propusă.

Au avut loc foarte multe schimbări de la apariția inteligenței artificiale. Înaintea acestora, toate acțiunile presupuneau în totalitate implicarea unui om, acum sunt posibile la apăsarea unor butoane, fapt care duce la ușurarea muncii omului.

Dupa apariția inteligenței artificiale, lumea de rând dispune de avantaje care la ochiul liber, nu sunt toate vizibile: siguranța în transport, eficiența în producție, descoperiri în știintă și multe altele. Proiectul propus susține folosirea tehnologiei în avantajul omului.

**Cuprins**

[1 Introducere 6](#_Toc105944647)

[1.1 Tema proiectului 7](#_Toc105944648)

[1.2 Descriere domeniu 7](#_Toc105944649)

[1.3 Matematica din spatele inteligenței artificiale 8](#_Toc105944650)

[2 Studiu bibliografic 9](#_Toc105944651)

[2.1 Trueface 9](#_Toc105944652)

[2.1.1 C++ 9](#_Toc105944653)

[2.1.2 Structura aplicației 9](#_Toc105944654)

[2.2 OpenFaceTracker 10](#_Toc105944655)

[2.2.1 OpenCV 3.2 10](#_Toc105944656)

[2.2.2 Structura aplicației 10](#_Toc105944657)

[2.3 Face++ 10](#_Toc105944658)

[2.3.1 ‘Face’ 10](#_Toc105944659)

[2.3.2 Structura aplicației 11](#_Toc105944660)

[3 Baze teoretice 12](#_Toc105944661)

[3.1 Tehnologii folosite 12](#_Toc105944662)

[3.1.1 Limbajul Python 12](#_Toc105944663)

[3.1.2 OpenCV Python 13](#_Toc105944664)

[3.1.3 Biblioteca Dlib 13](#_Toc105944665)

[3.1.4 Vedere artificiala (Computer Vision) 14](#_Toc105944666)

[3.1.5 Teachable machine 15](#_Toc105944667)

[4 Soluția propusă 16](#_Toc105944668)

[4.1 Descriere aplicație 16](#_Toc105944669)

[4.2 Structura generală a aplicației 16](#_Toc105944670)

[5 Implementarea soluției propuse 18](#_Toc105944671)

[5.1 Descrierea în detaliu a componentelor aplicației 18](#_Toc105944672)

[5.1.1 Basic.py 18](#_Toc105944673)

[5.1.2 FacialRecognition.py 23](#_Toc105944674)

[5.1.3 ImagesDetection.py 33](#_Toc105944675)

[5.1.4 SecurityCamera.py 39](#_Toc105944676)

[5.1.5 ImagesDeposit folder 40](#_Toc105944677)

[5.1.6 Folderul aplicației 41](#_Toc105944678)

[5.1.7 Interfața cu utilizatorul 42](#_Toc105944679)

[6 Rezultate experimentale 43](#_Toc105944680)

[6.1 Testare comportament aplicatie 43](#_Toc105944681)

[6.1.1 Comportamentul recunoașterii faciale 43](#_Toc105944682)

[6.1.2 Comportamentul predicțiilor 45](#_Toc105944683)

[7 Concluzii 49](#_Toc105944684)

[7.1 Viitoare imbunatatiri 50](#_Toc105944685)

[8 Bibliografie 51](#_Toc105944686)

# Introducere

Evoluţia societăţii umane este marcată periodic de realizarea unor mari descoperiri ştiinţifice, care determină schimbări profunde în toate domeniile vieţii. Nu greşim dacă presupunem că apariţia inteligenței artificiale este o descoperire cel puţin egală ca impact, de altfel conducând la o desfașurare îmbunătațită a unor activități.

Ținând cont de evoluția tehnologiei în ultimii ani, putem spune fără dubii că societatea este una puternic dezvoltată. Întâlnim sisteme inteligente care facilitează sau îmbunătățesc lucrul în diversele sectoare de activitate, în viața de zi cu zi:

* Sistemul de sănătate
* Automobile
* Gaming
* Agricultură
* Robotică
* Educație
* Finanțe
* Supraveghere
* Social Media
* Energie



Figura 1: Utilități ale inteligenței artificiale în viața cotidiană

## Tema proiectului

Recunoașterea feței este una dintre cele mai relevante aplicații ale analizei imaginii. Este o adevărată provocare să construiești un sistem automatizat care echivalează cu capacitatea umană de a recunoaște fețele. Deși oamenii sunt destul de buni în identificarea fețelor cunoscute, nu sunt pricepuți când se confruntă cu o mare cantitate de chipuri necunoscute. Calculatoarele, cu o memorie aproape nelimitată și de calcul viteză, ar trebui să depășească limitele oamenilor.

Clasificarea obiectelor bazat pe predictii este a doua temă a proiectului, aceasta presupune folosiea retelelor neuronale și prelucrarea imaginii. Conceptul de rețele neuronale este în general unul necunoscut, această funcționalitate implementată este menită a pune în evidență o parte din potențialul acestora.

Sistemele de supraveghere și-au făcut apariția încă din anul 1942, de atunci tehnologia s-a dezvoltat exponențial, astăzi camerele de supraveghere și securitate fiind pe zi ce trece, din ce in ce mai performante. A treia temă este inspirată de evoluția menționată anterior, scopul este de a implementa comportamentul unei camere de securitate [1].

## Descriere domeniu

Putem începe cu definiția inteligenței artificiale. IA este capacitatea unei mașini de a imita funcții umane, cum ar fi raționamentul, învățarea, planificarea și creativitatea.

IA permite sistemelor tehnice să perceapă mediul în care funcționează, să prelucreze această percepție și să rezolve probleme, acționând pentru a atinge un anumit obiectiv. Calculatorul primește datele (deja pregătite sau colectate prin intermediul propriilor senzori, cum ar fi o cameră video), le prelucrează și reacționează. Sistemele IA sunt capabile să își adapteze, într-o anumită măsură, comportamentul, analizând efectele acțiunilor anterioare și funcționând autonom.

Pentru a putea descrie în totalitate domeniul, trebuie sa fim mai preciși, mai exact trebuie să răspundem la intrebarea „ce este un software de recunoaștere facială?” Recunoașterea facială este o formă de învațare automată unde un software procesează algoritmi complecsi în jurul trăsăturilor faciale ale unei persoane și le compară cu o identitate deja cunoscută din baza de date. Funcționează simliar cu software-ul de recunoaștere biometrică, cum ar fi recunoașterea amprentei sau recunoașterea irisului. Cu toate acestea, recunoașterea facială este mai avansată decât alte tipuri de software.

Software-ul este capabil să identifice diferite aspecte ale feței unei persoane, cum ar fi nasul, ochii, bărbia și obrajii și apoi să le cartografieze unul în raport cu celălalt. Aspectele unice ale caracteristicilor noastre care ne fac pe fiecare dintre noi sa arătăm diferit sunt luate în considerare de software, astfel încât să ne poată identifica. Precizia celui mai bun software de recunoaștere facială constă în puterea bazei de date. Cu cat baza de date este mai mare, cu atat este mai probabil ca software-ul să poată identifica o persoană.

Unul dintre marile avantaje ale software-ului de recunoaștere facială în afaceri îl reprezintă funcțiile de securitate avansate, ceea ce, am încercat sa reconstitui la scară mică.

Recent, criminalitatea cibernetică a fost in creștere, forțând multe companii să-și îmbunătățească măsurile de securitate. Una dintre cele mai comune modalitați de a proteja datele sensibile de un atac cibernetic este utilizarea parolelor complexe care pot împiedica un hacker să le descifreze și să obțină acces la un sistem. Dar chiar și cele mai puternice parole sunt încă vulnerabile la amenințări.

Aplicațiile de recunoaștere facială, pe de alta parte, sunt o alternativă mult mai sigură. Nu numai că sunt simplu de utilizat, deoarece un utilizator nu trebuie să își amintească parolele, dar sunt aproape imposibil de spart. Numai utilizatorul a cărui față a fost autorizată să se conecteze la sistem poate obține acces la acesta. Acest lucru face ca tehnologia de recunoaștere facială să fie extrem de importantă pentru companiile care fie se ocupă direct de volume mari de date sensibile, fie sunt paznicii datelor private.

## Matematica din spatele inteligenței artificiale

Pentru început, un sistem de recunoaștere nu trebuie să fie afectat atat de schimbările externe, cum ar fi lumina ambinentală, poziția și distanța persoanei față de cameră cât și de variațiile interne, precum expresia facială, îmbătrânirea și machiajul. Deoarece majoritatea aplicațiilor comerciale folosesc baze de date mari de fețe, sistemele de recunoaștere trebuie să fie eficiente din punct de vedere computațional. Aici intră în joc matematica.

Majoritatea algoritmilor de recunoaștere a feței se încadrează în unul din cele două mari grupuri de algoritmi: algoritmi bazați pe caracteristici și algoritmi bazați pe imagini. Modelele bazate pe caracteristici explorează un set de caracteristici geometrice, cum ar fi distanța dintre ochi sau mărimea ochilor și folosesc aceste măsuri pentru a reprezenta fața dată. Aceste caracteristici sunt calculate folosind filtre de corelație și sunt oarecum imune la modificările de lumină și ale poziției camerei. Cu toate acestea, sunt sensibili la îmbătrânire și la expresiile faciale. Pe de altă parte, sistemele bazate pe imagini, a doua cea mai mare abordare a recunoașterii faciale, se bazează pe ideea feței proprii, un set înrudit a caracteristicilor faciale pe care un computer le foloseste pentru a recunoaște fața unei persoane.

Fețele pot varia în funcție de cel puțin 100 de factori, computerul trebuie să înțeleagă care sunt acești factori. Fiecare imagine a feței este deconstruită într-un set separat de caracteristici faciale asociate iar un algoritm este creat astfel încât computerul să poată să înțeleagă imaginea și să o analizeze în comparație cu altele.

# Studiu bibliografic

În cadrul acestui capitol se vor prezenta trei aplicații studiate care au în comun utilizarea viziunii computerizate, precum și a unor funcționalități.

## 2.1 Trueface

Trueface este un SDK (software development kit) conceput pentru 4 funcționalități: recunoaștere facială, detectarea reperelor faciale, detectarea obiectelor și estimarea pozitiei capului.

2.1.1 C++

C++ este un limbaj de programare folosit pentru a dezvolta sisteme de operare, browsere, jocuri și așa mai departe. C++ oferă flexibilitate prin faptul ca suportă diferite feluri de programare precum: procedurală, orientată pe obiecte si funcțională [2].

2.1.2 Structura aplicației

SDK-ul are ca idee principală folosirea limbajului C++ pentru a reuni un grup de instrumente care permit programarea aplicațiilor mobile. O aplicație a Trueface este dezvoltarea unei aplicații de verificare a indentității într-o instituție bancară.

A person holding a phone

Description automatically generated with medium confidence

Figura 2: Aplicație de verificare a identității

## 2.2 OpenFaceTracker

OpenFaceTracker este un program de recunoaștere facială capabil să detecteze una sau mai multe fețe pe o imagine sau un videoclip și să le identifice printr-o bază de date.

2.2.1 OpenCV 3.2

Aplicația folosește versiunea 3.2 a OpenCV, superioară versunii folosită într-o versiune anterioară, 2.4, deschizând oportunitatea folosiri unor algoritmi si proprietăți noi.

2.2.2 Structura aplicației

Aceasta constă într-un UI, conținând o fereastră prin care este afișată înregistrarea video live de la una sau doua camere, un buton de reîncărcare, un buton de trecut la cadrul următor, un boton pentru statistici legate de eficacitatea recunoașterii si un buton pentru detectarea emoțiilor.

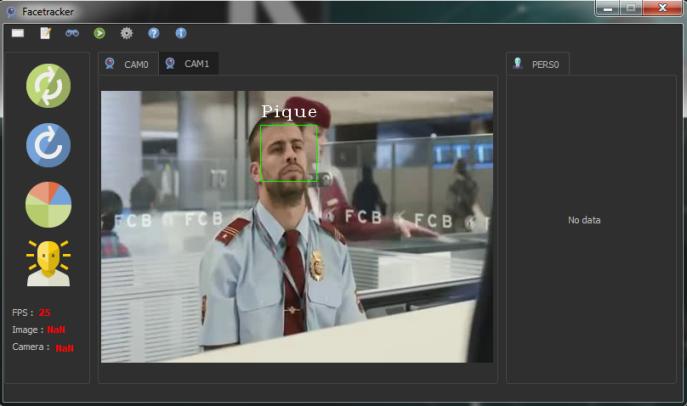


Figura 3: UI OpenFaceTracker

## 2.3 Face++

Face++ este o aplicație de recunoaștere a reperelor feței, această proprietate fiind folosită pentru a implementa un sistem sigur de tranzacții online. Doar printr-un set anume de mișcări ale feței, tranzacțiile se realizează cu succes.

2.3.1 ‘Face’

Face este numele algoritmului din spatele aplicației. La detectarea unei fețe dintr-o imagine, aceasta va fi înregistrată, împreună cu locațiile feței. Fața va fi marcată cu un ID unic numic face-token. Fiecare face-token este salvat într-o bază de date concepută astfel încât sa nu existe nici un duplicat [3].

2.3.2 Structura aplicației

Structura aplicației Face++ este reprezentată de modul prin care se asigură că operatorul din spatele tranzacției este un om. Trecând prin mai multe comenzi, orice tranzacție se aprobă doar la îndeplinirea tuturor cerințelor.

A picture containing text, monitor, iPod, screen

Description automatically generated

Figura 4: Respectarea comenzilor

# Baze teoretice

## Tehnologii folosite

Fără îndoială, un proiect de o asemenea amploare, chiar și la scară mică, presupune folosirea mai multor tehnologii, de la limbaje de programare, la hardware.

3.1.1 Limbajul Python

Python este un limbaj de programare de nivel înalt, interpretat, orientat pe obiecte, cu sintaxa simplă, ușor de învățat, accentuează lizibilitatea și prin urmare, reduce costul de întreținere a programului [4].

De ce am ales Python?

-dupa o perioada de cautare și informare, am observat ca Python este masiv folosit în proiecte care implică machine learning

-Python este un limbaj ușor de folosit, m-am obișnuit repede cu sintaxele iar erorile întâlnite pe parcurs au fost depanate relativ repede

Cu toate acestea, Python prezintă și dezavantaje, care au fost mai mult sau mai puțin întâlnite în proiectul realizat, așa cum apar în figura 5.

Figura 5: Avantaje/dezavantaje Python

3.1.2 OpenCV Python

OpenCV a fost dezvoltat inițial în 1999 de Intel, dar mai târziu a fost susținut de Willow Garage. OpenCV-Python este o bibliotecă imensă open-source pentru vedere computerizată, învățare automată și procesare de imagini. Joacă un rol major in operarea in timp real care este foarte importantă în sistemele actuale [5].

OpenCV este bazat pe doi algoritmi: **DNN** si **HAAR Cascades**

**Deep Neural Networks (DNN)** este un tip de învățare automată care imită modul în care creierul învață. Ideea generală a unui DNN este că învață printr-o acțiune repetitivă dintr-o colecție de mostre, cum ar fi 100 de imagini cu câini diferiți, spre deosebire de un set de reguli create de om, cum ar fi „un câine are nasul negru și coadă maro”. În acest fel, un DNN învață în același mod precum creierul uman – prin practică și greșeli [6].

**Haar Cascades** se bazează pe proprietăți similare ale fețelor umane pentru a forma potriviri din trăsăturile faciale: locația și dimensiunea ochiului, gurii, puntea nasului și gradienții orientați ale intensității pixelilor. Există 38 de straturi de clasificatoare în cascadă pentru a obține numărul total de 6061 de caracteristici din fiecare față frontală [7].

3.1.3 Biblioteca Dlib

Dlib este un set de instrumente C++ modern care conține algoritmi de învățare automată și instrumente pentru crearea de software complex în C++ pentru a rezolva problemele din lumea reală. Este folosit atât în industrie, cât și în mediul academic într-o gamă largă de domenii, inclusiv robotică, dispozitive încorporate, telefoane mobile și medii de calcul mari de înaltă performanță. Pe langa C++, acesta are multe aplicații în Python [8].

Dlib este bazat pe doi algoritmi: **CNN** si **HoG**

**O rețea neuronală convoluțională (CNN)** este un algoritm de învățare profundă care poate prelua o imagine de intrare, poate atribui importanță (greutăți și părtiniri învățate) diferitelor aspecte/obiecte din imagine și poate să le diferențieze unul de celălalt [9].

**Histogram of Oriented Gradients (HOG**) este un extractor de caracteristici pentru detectarea obiectelor. Caracteristicile extrase sunt distribuția direcțiilor gradienților imaginii.

Gradientele sunt margini și colțuri rotunjite mari și ne permit să detectăm acele regiuni. În loc să ia în considerare intensitățile pixelilor, algoritmul numără aparițiile vectorilor de gradient pentru a reprezenta direcția luminii, pentru a localiza segmentele de imagine. Metoda utilizează normalizarea contrastului local suprapus pentru a îmbunătăți acuratețea.

3.1.4 Vedere artificială (Computer Vision)

Pentru a înțelege conceptul de vedere computerizată, trebuie sa considerăm un scenariu:

-ne aflăm în vacanța mult dorită la mare, ne place să facem poze cu prietenii așadar vrem sa le postăm

-în momentul în care sunt postate, vrem să dăm tag oamenilor prezenți în poză iar aici facebook este cu un pas in față, dând automat tag la acele persoane

-cum functioneaza acest tag feature? În termeni simpli, prin vedere computerizată

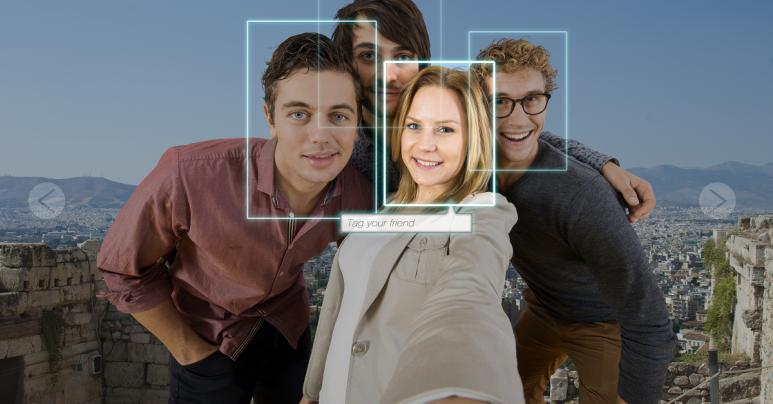


Figura 6: Facebook Auto-tag

Vederea computerizată este un domeniu care se ocupă de modul prin care calculatoarele obțin o ințelegere la nivel înalt asupra imaginilor și videourilor. Ideea este de a automatiza sarcinile pe care ochiul uman nu le poate realiza.

Vederea artificială permite computerelor să interpreteze și să analizeze lumea vizuală, simulând modul în care oamenii văd și înțeleg mediul lor. Acesta aplică modele de învățare automată pentru a identifica și clasifica obiectele în imagini și videoclipuri digitale, apoi permite computerelor să reacționeze la ceea ce văd. Diferite tipuri de viziune computerizată includ segmentarea imaginii, detectarea obiectelor, recunoașterea facială, detectarea marginilor, detectarea modelelor, clasificarea imaginilor și potrivirea caracteristicilor.



Figura 7: Utilitate vedere artificială

Vederea artificială permite o gamă largă de inovații tehnologice. Permite mașinilor cu conducere autonomă să vireze în siguranță pe străzi și autostrăzi, permite instrumentelor de recunoaștere facială să potrivească imaginile fețelor oamenilor cu identitățile lor și permite aplicațiilor de realitate augmentată să combine obiecte virtuale cu imagini din lumea reală [10].

Aplicațiile de vedere artificială sunt utilizate în toate industriile pentru a îmbunătăți experiența consumatorului, a reduce costurile și a întări securitatea. Producătorii îl folosesc pentru a identifica produsele defecte pe linia de asamblare și pentru a le împiedica să ajungă la clienți. Agenții de asigurări îl folosesc pentru a evalua daunele vehiculului și pentru a reduce frauda în procesul de reclamație. Medicii îl folosesc pentru a scana raze X, RMN și ultrasunete pentru a detecta probleme de sănătate. Băncile îl folosesc pentru a verifica identitățile clienților înainte de a efectua tranzacții mari.

3.1.5 Teachable machine

Teachable machine este un instrument Web care folosește rețele neuronale și care creează modele de învațare automată, instrument de care ne vom folosi pentru a atrena un model, pe care il vom folosi ulterior în cod. Teachable machine are la bază:

**MobileNet** – rețeaua neuronală, bazată pe depthwise separable convolutions

**Transfer learning** – tehnica folosită pentru modele; tehnica de învățare automată în care un model antrenat pentru o sarcină este reutilzat pentru o a doua sarcină asociată

# Soluția propusă

## 4.1 Descrierea aplicației

Aplicația este menită să îndeplinească trei funcționalități, fiecare conținând tehnologii și procese diferite. Prima idee, și ideea principală de la care a început aplicația a fost recunoașterea facială. Pornind de la acest concept, prima practicabilitate a aplicației o reprezintă viziunea computerizată, astfel, printr-o bibliotecă performantă care aparține de Python: face-recognition, prima funcționalitate este recunoașterea fețelor după un singur model. Recunoașterea facială este mai apoi folosită pentru a lansa o alarmă la detectarea unei persoane necunoscute, sau considerată neautorizată. A doua funcționalitate a aplicației clasifică imagini și obiecte prin ajutorul unei aplicații deja existente:’Teachable Machine’. Având la bază rețeaua neuronală MobileNet, aceasta are o capabilitate imensă de învățare, rezultând într-o precizie mare în detectarea obiectelor. Aceasta deschide oportunitatea de a scana și trimite alerte în cazul detectării unor obiecte periculoase. A treia și nu cea din urmă funcționaliate‚ camera de securitate, pune în lumină logica la nivel de cod din spatele unei camere de supraveghere și un algoritm renumit de detecție și recunoștere: haar cascades, algoritm folosit în multe aplicații din viața reală.

## 4.2 Structura generală a aplicației

Structura aplicației este susținuta în primul rând de interfața utilizatorului, întrucât expune fiecare funcționalitate într-o manieră simplă dar eficientă. Cu ajutorul bibliotecii tkinter, aplicația expune cele 3 funcționalitați în trei butoane.

Interfața utilizatorului

Rularea aplicației

Biblioteci necesare

Buton cameră de securitate

Depozit de imagini

Buton recunoaștere obiecte

Buton recunoaștere facială

Model antrenat

Executabil aplicație

Figura 8: Structura interfeței și a rulării aplicației

Recunoașterea facială este primul pilon al aplicației, la baza recunoașterii, se află locul de unde programul este învățat să își extragă și să găsească informațiile necesare: ImagesDeposit, foderul care conține toate imaginile, acesta jucând un rol important în structura aplicației. Aceasta este utilizată pentru a detecta persoane necunoscute, sau persoane considerate neautorizate.

Detecția obiectelor are ca fundație modelul keras: classes.h5. Acest model este rezultatul ‚antrenamentelor’, conține toate datele necesare pentru a putea implementa clasificările la nivel de cod. Clasificările sunt cunoscute și sub termenul de predicții. Detecția obiectelor are scopul nu doar de a detecta simple obiecte dintr-o cameră, ci și pentru a detecta obiecte periculoase precum cuțite sau arme.

Structura este completată de ‚camera de securitate’. Un concept transpus în cod și adus la viață prin folosirea clasificatoarelor în cascadă bazate pe caracteristici Haar, aceste caracteristici fiind esențiale în recunoașterea corporală. Camera de supraveghere completează cu încredere ideea de sistem de securitate bazat pe vedere computerizată.

# Implementarea soluției propuse

## Descrierea în detaliu a componentelor aplicației

Vom începe prin a lua pe rand fiecare fișier .py din folderul principal, documentând pas cu pas implementarile făcute și elaborând unde este nevoie, pentru o ințelegere cât mai fluidă asupra acestora și asupra structurii

5.1.1 Basic.py

Basic.py nu face parte din aplicația propriuzisă, acesta fiind doar prototipul pentru a asigura funcționalitatea principală adica recunoașterea facială, basic.py este motorul de la care pornește aplicația. Pentru a ajunge la funcționalitatea dorită, progresul va fi măsurat în pași.

**Pas 0**

Instalarea pachetelor necesare pentru a putea implementa proiectul dorit

**O imagine care conține text

Descriere generată automat**

Figura 9: Pachetele și bibliotecile esențiale

**Pas 1**

Avem o imagine cu Elon Musk și o imagine test cu acesta, de asemenea încă o imagine test pentru Bill Gates.

Învățăm programul să ne aducă codificările imaginii normale și vom folosi imaginea test să vedem dacă ne poate găsi imaginea normală a lui Elon sau nu.

Când un fișier imagine este citit prin funcția OpenCV: imread(), ordinea culorilor este BGR (blue, green, red). Pe de altă parte, în Pillow, ordinea culorilor este considerată RGB (red, green, blue). În imaginea de jos se află codul aferent conversiei.

Așadar, daca vrem să folosim atât funcția Pillow cât și OpenCV, trebuie să convertim imaginile din BGR in RGB, folosid funcția cvtColor().

O imagine care conține text, captură de ecran, interior, monitor

Descriere generată automat

Figura 10: Conversia imaginilor test în RGB

**Pas 2**

Găsirea fețelor și a codificărilor lor în imagine. Găsirea codificărilor este un pas important întrucât acestea sun responsabile pentru găsirea locațiilor unei fețe. Imaginea ce urmează reprezintă primul exemplu de față detectată.

O imagine care conține text, persoană, bărbat, interior

Descriere generată automat

Figura 11: Primul chip detectat

**Pas 3**

O imagine care conține text, interior, captură de ecran, monitor

Descriere generată automatComparăm fețele și găsim distanța dintre ele. Găsirea distanței este esențială pentru a putea determina cat de aproape sunt două imagini de potrivire. În imagine se dezvăluie codul pentru a calcula distanța.

Figura 12: Cod găsire distanță între fețe

Valoarea true = programul a gasit similare codificarile imaginilor = sunt aceeași persoană. Valoarea false = ne-a detectat fețele, dar codificările nu sunt aceleași.

La compararea între două fețe diferite, rezultatul a fost False, urmat de o distanță mare, așa cum se poate observa în imagine.

A screenshot of a person

Description automatically generated with low confidence

Figura 13: Rezultatul false la persoane diferite

**Pas 4**

În cazul în care avem mai multe imagini și există similarități, vrem sa gasim distanța, vrem să găsim cea mai buna potrivire. Cu cât mai mică distanța, cu atât mai bună potrivirea. Așadar, vom afișa pe ecran rezultatul distanței, prin codul evidențiat.

O imagine care conține text, captură de ecran, monitor, interior

Descriere generată automat

Figura 14: Cod găsirea valorii distanței

Dupa cum se poate vedea în imagine, dacă schimbăm înapoi la cele două imagini cu Elon, ne returnează valoarea 0.3, o distantă mult mai mică întrucât în imagini este aceeași persoană. Imaginea de mai jos este dovada unei comparații de succes, urmată de o distanță mică, susținând rezultatul.

O imagine care conține text, persoană, interior, postură

Descriere generată automat

Figura 15: Rezultatul True la aceeași persoană

**Pas 5**

Afișare pe ecran rezultatul comparației + distanța. Codul evidențiat din imagine arată sintaxa funției cv2.putText.

O imagine care conține text, interior, captură de ecran, monitor

Descriere generată automat

Figura 16: Cod afișare pe ecran comparație și distanța

Prin imaginile ce urmează consolidăm un comportament corect din codul scris până acum. Primul exemplu fiind Elon Musk, iar al doilea Bill Gates: un raspuns True, respectiv unul False.

O imagine care conține text, persoană, bărbat, interior

Descriere generată automatO imagine care conține text, persoană, bărbat, interior

Descriere generată automat

Figura 17: Afișare pe ecran rezultatul True și distanța

5.1.2 FacialRecognition.py

FacialRecognition.py este fisierul care este responsabil de recunoașterea facială propriuzisă. În imagine s-a facut transferul de cod, din main.py, in Facial.Recognition.py.

O imagine care conține text

Descriere generată automat

Figura 18: Începerea codului final

Ca înainte, primul pas este să ne importăm bibliotecile, al doilea este să ne importăm imaginile ca mai apoi să le convertim în RGB.

Daca ne uităm mai atent la acestă parte de cod, oricând aducem o imagine nouă, trebuie să o scriem manual, sa ii dăm un nume, să o stocăm și să ii găsim codificările. Este un proces lung și neeficient.

Ca să optimizăm procesul, creăm o listă care să aducă imaginile automat din folder-ul ‘ImagesDeposit’, să genereze codificările automat, ca într-un final să încerce să le găsească în webcam-ul nostru. Folderul se poate vedea în figura următoare.

Graphical user interface, text, application

Description automatically generated

Figura 19: Folderul cu imagini

🡪În continuare, vom cere programului să ne găsească acest folder, să numere câte imagini sunt în folder, să le importe și să găsească codificările lor. În figura următoare este evidențiată în cod, calea către depozitul de imagini.

Graphical user interface, text, application

Description automatically generated

Figura 20: Calea către folderul cu imagini

Importăm biblioteca OS, care ne oferă funcții pentru a interacționa cu sistemul de operare

images = lista imaginilor, classNames = preluarea numelor

Programul ne-a returnat cu succes numele tuturor celor trei imagini din folder

🡪Vom folosi aceste nume și vom importa imaginile una câte una, prin funcțiile cv.imread, cv2.os și append, după cum se vede în figură

O imagine care conține text, captură de ecran, monitor, interior

Descriere generată automat

Figura 21: Citirea imaginilor de către program

🡪 începem procesul de codificare, odată ce importăm o imagine trebuie să găsim codificările pentru fiecare în parte. În figura următoare este evidențiat procesul.

Creăm o funcție care să calculeze codificările

* În interiorul funcției, creăm o listă goală care să conțină toate codificarile la final : encodeList()
* Convertim imaginea
* Obținem codificarea

O imagine care conține text, captură de ecran, monitor, interior

Descriere generată automat

Figura 22: Cod găsirea codificărilor

Programul a găsit cu succes numărul imaginilor din folder, prin urmare trei codificări

🡪 găsirea potrivirilor între codificari

Nu avem o imagine cu care să se potrivească, acea imagine va veni din camera video

* Inițializăm camera video
* Folosim o structură repetitivă while pentru a găsi fiecare frame, unul câte unul
* Deoarece lucrăm în timp real, vrem să reducem mărimea imaginii (1/4 din imagine) făcând procesul mai rapid
* Convertim imaginea
* Găsirea codificării camerei web
* În camera web e posibil sa găsim multiple fețe, asadar vom găsi întâi locația fețelor, apoi trimitem acele locații funcției de encode, dupa cum se observă în imagineO imagine care conține text

  Descriere generată automat

Figura 23: Funcția encode preluând locațiile fețelor

🡪 trecem prin toate fețele pe care le-am găsit în Current Frame și comparăm toate fețele cu toate codificările pe care le-am găsit înainte

* Una câte una, structura lor ne va aduce o locație de fața din ‘facesCurFrame’ și o codificare din ‘encodesCurFrame’ folosind funcția zip
* Trimitem lisța fețelor cunoscute: ‘encodeListKnow’/ Comparăm o față cu una dintre codificări: ‘encodeFace’
* Găsim distanța trimițând aceeași parametri
* Deoarece trimitem o listă către funcția ‘face\_distance’, ne va returna tot o listă
* Ne va returna trei valori deoarece avem trei fețe cunoscute și ne va da distanța fiecăreia, cea mai mică distanța fiind cea mai bună potrivire. În figură este evidențiată structura repetitivă responsabilă pentru găsirea valorilor.O imagine care conține text

  Descriere generată automat

Figura 24: Cod returnare valori: distanța fețelor

Explicație valori din figură: ca un prim exemplu accesibil, mi-am apropiat fața de camera web iar valorile au fost cele așteptate, distanțe mari 0,8 0,9 întrucât fața mea nu seamănă deloc cu a celor trei oameni din imagini

O imagine care conține text, captură de ecran, monitor, negru

Descriere generată automat

Figura 25: Valori găsite corespunzator cu numărul de fețe din folder

Exemplu 2: am apropiat de camera web o poză cu Bill Gates descarcată pe telefon, el fiind primul din listă, valorile cosiderabil mai mici de 0.3 sunt corecte, față de 0.7 și 0.8

🡪 găsind cel mai mic element din lista ne rezultă cea mai bună potrivire. În figura de jos este evidențiat procesul pentru găsirea valorii mici.

O imagine care conține text, captură de ecran, monitor, computer

Descriere generată automat

Figura 26: Cod găsirea celei mai mici valori

Cand arăt la cameră imaginea cu Bill Gates, programul precizează că este vorba de el, urmat de un set de valori corecte. După cum se poate vedea în imagine.

Graphical user interface, application, Teams

Description automatically generated

Figura 27: Valori mici la imaginea cu Bill Gates

🡪 construim o încadrare pentru față, de asemenea, afișam numele, după cum este prezentat in figura de mai jos

A picture containing graphical user interface

Description automatically generated

Figura 28: Cod desenare dreptunghi peste chip

-Ruland codul păna aici ne dăm seama că avem o problemă, detectează persoana bine, dar locația este greșită

-Motivul din spate este că am modificat imaginea la ¼ din dimensiunea ei, apoi am efectuat toate calculele

-Pentru a reveni la imaginea full, soluția este să multiplicăm valorile (coordonatele) cu 4. Parte de cod evidențiată in figura ce urmează

Calendar

Description automatically generated with medium confidence

Figura 29: Corectarea locațiilor

Următoarele 6 figuri reprezintă un ‘album’ al persoanelor aflate în folderul Images deposit și recunoscute fără probleme.

Graphical user interface, application, Teams

Description automatically generatedGraphical user interface, application, Teams

Description automatically generated

Graphical user interface, application, Teams

Description automatically generatedO imagine care conține text, monitor, interior, captură de ecran

Descriere generată automat

Figura 30: Album recunoașteri cu succes

Ne intâlnim cu o problemă în ultima structură a codului. Figura de mai jos este bucata de cod problematică.

O imagine care conține text

Descriere generată automat

Figura 31: Dificultate în programarea Python

🡪expresia while true merge la infinit, întrucât avem nevoie ca programul să detecteze constant chipuri noi

🡪având o condiție while true ne asigurăm ca programul merge pâna il oprim noi

🡪aici ne întâlnim cu o problemă, spre deosebire de alte limbaje de programare, Python nu foloseste '{}' iar asta prezintă o dificultate în delimitarea bucăților de cod

🡪deoarece while true nu se oprește, iar bucata de cod nu poate fi delimitată cu acoalade, orice scriu în interiorul while-ului se va executa de atâtea ori cât merge el, iar in afara lui, ne lovim de eroarea 'unreachable code'

🡪prin urmare, e esențială găsirea unui mod de a ieși din while, soluția la prima vedere fiind un break

Pentru a ieși din program ne-am folosit de multitudinea de accesorii a bibliotecii cv2, am setat tasta ‘q’

ca o tastă de oprire, ieșirea din while se face la apăsarea tastei q, după cum se poate vedea în figură.

O imagine care conține text

Descriere generată automat

Figura 32: Apăsarea tastei q pentru a ieși din bucla infinită

Dupa cum am menționat mai sus, inteligența artificială are o gamă variată de aplicabilitate în domeniul securității. În acest sens, am implementat în cod trimiterea unei notificări la nivel de windows, cu atenționare sonoră, la detectarea unei persoane necunoscute sau considerată ‚neautorizată’. În următoarea imagine avem exemplu de cum arată această notificare.

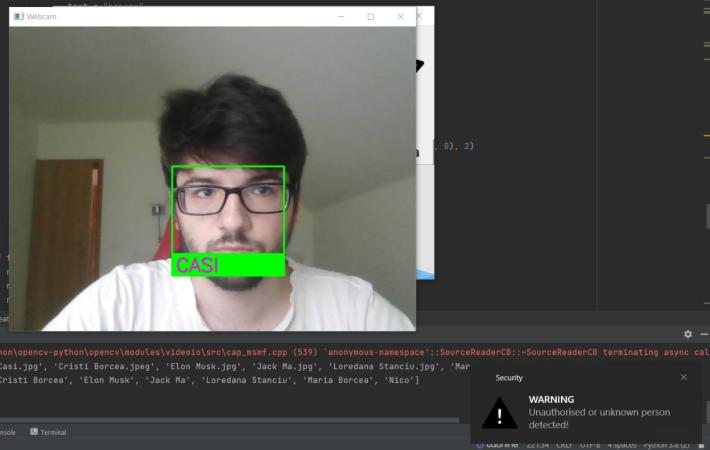


Figura 33: Notificare alertă recunoaștere facială

Notificarea este făcuta cu ajutorul bibliotecii notity, acesta permite lansarea și editarea notificărilor windows, având opțiunea de a avea titlu, mesaj, icon și efecte sonore.

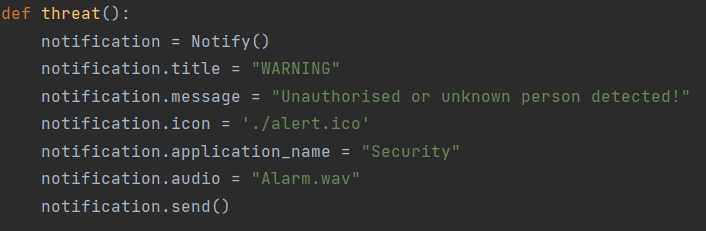


Figura 34: Cod notificare windows

Semnalul de alarmă este setat la nivelul buclei infinite, în care recunoaștere facială operează. Daca o persoană este considerată neautorizată sau necunoscută, se apelează funcția threat() menționată mai sus.

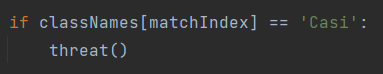


Figura 35: Cod persoană necunoscută sau neautorizată

5.1.3 ImagesDetection.py

ImagesDetection.py realizează recunoașterea de obiecte și ființe, prin intermediul unui model antrenat. În cadrul acestui fișier, este important să punctăm și tehnologiile folosite, pe lângă cod. Acestea sunt după cum urmează:

Teachable Machine

Graphical user interface, application

Description automatically generated

Figura 36: Utilizarea aplicației web Teachable Machine

Prin intermediul <https://teachablemachine.withgoogle.com/> am putut să îmi ‘antrenez’ un model

O imagine care conține text

Descriere generată automat

Figura 37: Exportarea modelului ca un model Keras

Dupa ce modelul este antrenat, putem descărca un model Keras, conținând predicțiile pentru fiecare clasă de obiecte.

Keras model

Keras este o interfață de programare a aplicațiilor (API) de rețea neuronală pentru Python, folosită pentru a construi modele de învațare automată. Modelele Keras oferă o modalitate simplă și ușor de utilizat de a defini o rețea neuronală, care va fi apoi creată prin TensorFlow

* în cod, modelul keras este găsit sub numele de ‘classes.h5’

În imaginea de jos avem o primă detecție și clasificare corectă, a unui telefon

O imagine care conține text, interior, monitor, computer

Descriere generată automat

Figura 38: Clasificarea obiectului detectat ca fiind telefon

Codul, explicat pas cu pas:

import tensorflow.keras  
import numpy as np  
import cv2  
  
# incarcam modelul extras  
model = tensorflow.keras.models.load\_model('classes.h5')  
  
# creez matricea de dimensiuni potrivite pentru a alimenta modelul keras  
data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)  
  
# calea catre camera mea web  
cam = cv2.VideoCapture(0)  
  
text = ""  
  
while True:  
  
 \_, img = cam.read()  
 img = cv2.resize(img, (224, 224))  
  
 # convertim imaginea intr-un numpy array prin np.asarray  
 image\_array = np.asarray(img)  
  
 # 'normalizam' imaginea = procesarea imaginii pentru a-i schimba nivelul intensivatii al pixelilor  
 # scopul este de a a aduce un contrast mai bun acolo unde avem stralucire proasta(glare)  
 normalized\_image\_array = (image\_array.astype(np.float32) / 127.0) - 1  
  
 # incarcam imaginea in array  
 data[0] = normalized\_image\_array  
  
 # rulam deducerea  
 prediction = model.predict(data)  
 for i in prediction:  
 if i[0] > 0.7: # pentru acuratete mai mare de 70%  
 text = "cat"  
 if i[1] > 0.7:  
 text = "mouse"  
 if i[2] > 0.7:  
 text = "bottle"  
 if i[3] > 0.7:  
 text = "scissors"  
 if i[4] > 0.7:  
 text = "tv remote"  
 if i[5] > 0.7:  
 text = "shoe"  
 if i[6] > 0.7:  
 text = "car keys"  
 if i[7] > 0.7:  
 text = "cup"  
 if i[8] > 0.7:  
 text = "apple"  
 if i[9] > 0.7:  
 text = "pen"  
 if i[10] > 0.7:  
 text = "person"  
 if i[11] > 0.7:  
 text = "phone"  
 if i[12] > 0.7:  
 text = "banana"  
 if i[13] > 0.7:  
 text = "marker"  
 if i[14] > 0.7:  
 text = "knife"  
 if i[15] > 0.7:  
 text = "firearm"  
 img = cv2.resize(img, (500, 500)) #redimensionam imaginea la 500 X 500  
 cv2.putText(img, text, (160, 40), cv2.FONT\_HERSHEY\_COMPLEX\_SMALL, 2, (0, 255, 0), 2) #afisarea prezicerii  
  
 if cv2.waitKey(1) == ord('q'): #apasare tasta 'q' pentru a iesi  
 break  
 cv2.imshow('img', img)

Pentru a susține ideea de securitate, pe care o putem regăsi în funcționalitățile unu și trei, modelul keras a fost reînnoit, conținând acum încă două clase menite a detecta arme periculoase. La fel ca la recunoașterea facială, programul semnalează o alarmă sonoră sub forma unei notificări windows, că un obiect vătămator a fost detectat.

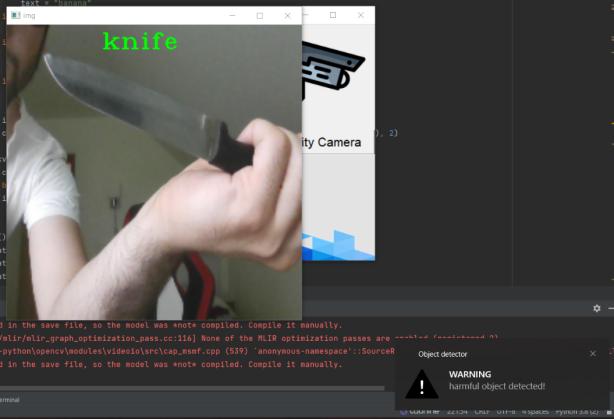


Figura 39: Notificare alertă detecție de obiecte

Similar cu recunoașterea facială, notificarea se gasește în cod ca o funcție (definiție în Python), alertând sonor user-ul și având opțiunea de a fi editată.

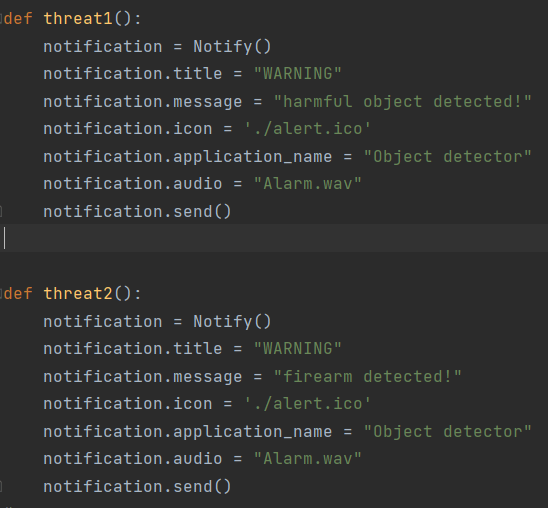


Figura 40: Cod notificare windows, detecție de obiecte periculoase

5.1.4 SecurityCamera.py

SecurityCamera.py asigură elementul de ‚securitate’, asigură funcționaliatea creării videoului/videourilor în cazul detectării unei fețe sau corp.

Implementarea unei ‘camere de securitate’ presupune următoarele: programul odată detectând o față sau o siluetă, să porneasca o înregistrare video, astfel captându-l sau captând-o pe durata în care apare pe ecran, înregistrarile mai apoi fiind salvate în folder. Este un mod eficient de a culege informații asupra persoanelor detectate, în eventualitatea nevoii de a revizui acele înregistrări. Ca și practicabilitate, acest sistem poate fi plantat într-un ATM, astfel orice acțiune fraudoasă va fi prinsă pe cameră iar avantajul e salvarea spațiului de stocare, înregistrarea pornind doar când detectează o față și oprindu-se când iese din cadru.

Codul, explicat pas cu pas:

import cv2  
import time # importurile necesare  
import datetime  
  
cap = cv2.VideoCapture(0) # creeam dispozitivul de capturare  
  
face\_cascade = cv2.CascadeClassifier(  
 cv2.data.haarcascades + "haarcascade\_frontalface\_default.xml")  
body\_cascade = cv2.CascadeClassifier( #initializam ‘haar cascades’  
 cv2.data.haarcascades + "haarcascade\_fullbody.xml")  
  
detection = False  
detection\_stopped\_time = None  
timer\_started = False #sectiunea de declaratii  
SECONDS\_TO\_RECORD\_AFTER\_DETECTION = 5  
  
frame\_size = (int(cap.get(3)), int(cap.get(4))) #captam dimensiunea cadrului si  
fourcc = cv2.VideoWriter\_fourcc(\*"mp4v") #setam formatul video prin ‘forcc’  
  
while True:  
 \_, frame = cap.read() #capturam cadrul  
   
 gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY) #convertim cadrul in gri  
 faces = face\_cascade.detectMultiScale(gray, 1.3, 5) #detectam chipurile  
 bodies = face\_cascade.detectMultiScale(gray, 1.3, 5) #si corpurile  
  
 if len(faces) + len(bodies) > 0: #am detectat vreun chip sau corp?  
 if detection:  
 timer\_started = False #daca inregistram, dam reset la timer  
 else:  
 detection = True  
 current\_time = datetime.datetime.now().strftime("%d-%m-%Y-%H-%M-%S")  
 out = cv2.VideoWriter(  
 f"{current\_time}.mp4", fourcc, 20, frame\_size)  
 print("Started Recording!")  
 elif detection: #daca nu se detecteaza nimic cat intregistrarea e  
 if timer\_started: #activa, verificam daca apare ceva in limita de timp  
 if time.time() - detection\_stopped\_time >= SECONDS\_TO\_RECORD\_AFTER\_DETECTION:  
 detection = False  
 timer\_started = False #daca timerul e pornit, verificam daca a  
 out.release() #expirat, daca da, oprim inregistrearea  
 print('Stop Recording!')  
 else:  
 timer\_started = True #daca timerul nu este pornit, il pornim si tinem  
 detection\_stopped\_time = time.time() # evidenta de cand a inceput  
  
 if detection:  
 out.write(frame)  
  
 cv2.imshow("Camera", frame)  
  
 if cv2.waitKey(1) == ord('q'):  
 break

5.1.5 ImagesDeposit folder

Folderul ImagesDeposit este resursa esențială a aplicației de recunoaștere facială. Deocamdată folderul este populat cu 201 imagini, însemnând 201 fețe umane.

În ceea ce privește respectarea condifențialitătii și siguranței datelor, patru persoane au oferit voluntar o poză, restul imaginilor fiind generate aleatoriu și descărcate, orice asemănare cu o persoana reală este pură coincidență.

Graphical user interface, application

Description automatically generated

Figura 41: Folderul cu imagini

5.1.6 Folderul aplicației

Aplicația finală este exportată sub forma unul executabil (MLX3.exe), pentru a avea toate functionalitățile operaționale, executabilul a fost generat împreună cu resursele necesare:

* ImagesDeposit – folderul cu imagini pentru recunoaștere facială
* lib- folderul care contine pachetele și bibliotecile Python
* cam.png, detect.png, face.png, geometry.png – imaginile folosite în interfață
* classes.h5 – modelul antrenat pentru a detecta obiecte
* fisierele dll – alte biblioteci necesare pentru proiect

Graphical user interface, application, table

Description automatically generated

Figura 42: Structura folderului aplicației

5.1.7 Interfața cu utilizatorul

Interfața utilizatorului este realizată cu biblioteca Tkinter din Python, aceasta are trei butoane, fiecare pornind câte o funcționalitate, ieșirea din fiecare fereastră deschisă se face cu tasta ‚q’.

Un buton realizat în Tkinter urmează următori pași: alegerea ferestrei, alegerea textului, alegerea unui font, alegerea lățimii, alegerea unei comenzi care apeleaza o funcție (definiție) și alegerea modului de împachetare și orientare în spațiu. Toate aceste proprietăți ale butonului se pot observa în figura următoare.

Text

Description automatically generated

Figura 43: cod buton în Tkinter

În imaginea de mai jos vedem rezultatul aranjării butoanelor și a icon-urilor, așadar și forma finală a interfeței utilizatorului.

A picture containing graphical user interface

Description automatically generated

Figura 44: Interfața cu utilizatorul

# Rezultate experimentale

## Testare comportament aplicație

6.1.1 Comportamentul recunoașterii faciale

Testarea acestei funcționalitati a aplicației, este cel mai profund determinată prin plasarea și raspunderea la mai multe întrebări de performanță.

Câte persoane poate aplicația să recunoască deodată?

Datorită limitărilor cadrului, în figura ce urmează avem dovada că numarul maxim de persoane recunoscute simultan este patru

A collage of a person wearing glasses and a white shirt

Description automatically generated with low confidenceGraphical user interface, application

Description automatically generated

Figura 45: Testare numar maxim de recunoaștere per cadru

Un om poate fi recunoscut dacă poartă mască?

Deși figura de mai jos arată o recunoaștere reușită, detecția în timp real are întreruperi

O imagine care conține text, monitor, interior, computer

Descriere generată automat

Figura 46: Recunoaștere cu succes persoană cu mască

Dar cagulă?

După cum putem vedea în figură, nu detectează, întrucât biblioteca ‚face-recognition’ din Python are un algoritm de recunoaștere de tipul feature-based, asta însemnând că reperele faciale precum nasul, gura, fruntea, părul sunt esențiale pentru o recunoaștere de succes.

Graphical user interface

Description automatically generated

Figura 47: Recunoaștere fără succes persoană cu cagulă

Întrucât aplicația poate lucra cu cantități de date, se impune realizarea unor teste de volum si stres. Testele sunt menite a determina cât de repede poate aplicația să îndeplinească o funcție, cât de bine poate să proceseze un volum mare de date și cum reacționează la suprasolicitari.

În tabelul ce urmează este reprezentat comportamentul aplicației, în coloana din stânga având numărul de imagini iar în coloana din dreapta timpul de procesare efectuat, la apăsarea butonului ‚face recognition’ din UI.

|  |  |
| --- | --- |
| Număr imagini | Timp de procesare |
| 200 | Necunoscut, aplicația se blochează |
| 50 | 34s |
| 30 | 24s |
| 15 | 18s |
| 10 | 15s |

Tabel 1: Timpul de procesare raportat la numărul de oameni

6.1.2 Comportamentul predicțiilor

Comportamentul predicțiilor poate fi afectat în funție de următoarele:

* Insuficiente date de antrenare
* La obținerea datelor de antrenare, obiectul nu este singur și clar în imagine
* Un număr inegal de date de antrenare per clase
* În figura 48, avem ca exemplu doua clasificări eșuate datorita unui număr mic de imagini de antrenare

A picture containing text, indoor, person

Description automatically generatedA picture containing text, person, indoor

Description automatically generated

Figura 48: Eroare în clasificarea obiectelor

Un alt exemplu al unei clase cu prea putine date de antrenare se găsește în figura 42

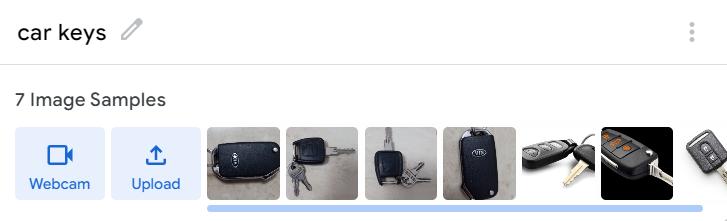


Figura 49: Exemplu clasa Teachable Machine

Eficacitatea predicțiilor este bazată pe cât de bine antrenat este modelul, în stadiile incipiente unele clase nu au avut mai mult de 10 modele/tip de obiect, fapt care rezultă în preziceri invalide. Numărul optim de date de antrenare este de la 500 în sus, așa cum se observă în imaginea de mai jos.

Graphical user interface

Description automatically generated

Figura 50: Adăugarea mai multor imagini de antrenare

În urma alocării timpului pentru a obține semnificativ mai multe mostre, eficacitatea clasificărilor a avut parte de o creștere.

Când ne referim la un model antrenat, care folosește o rețea neuronală, există un set de teste la care modelul poate fi supus, pentru a determina eficiența sa.

-înainte de a testa în sine modelul este important de menționat că:

* rețeaua neuronală pe care se bazează modelele este una convoluțională (CNN)
* metrica principală dupa care este testat modelul este acuratețea, modul principal prin care este calculată performanța modelului

Date de antrenare și date de test:

-date de antrenament: 85% din mostre, sunt folosite pentru a antrena modelul în a clasifica corect mostrele adăugate în clase

-date de test: 15% din mostre, niciodată folosite pentru a antrena modelul, folosite doar pentru a determina cât de bine se descurcă modelul cu o informație ‚nemaivazută’

* Acuratețe

Acuratețea este procentajul clasificărilor corecte făcute în timpul antrenării. Spre exemplu, daca 70 de mostre sunt clasificate corect din 100, atunci acuratetea este 70/100= 0.7. Daca predicțiile modelului sunt perfecte, acuratețea este 1. Din totalul de 16 clase, așadar 16 predicții, predicțiile clasei papuc nu au o acuratețe de 100%, așa cum se poate vedea în figura 51.

Table

Description automatically generated

Figura 51: Tabel acuratețe model

* Matricea de confuzie

Matricea de confuzie rezumă cât de precise sunt predicțiile modelului. Acestă matrice este folosită pentru a determina despre care clase este modelul ‚confuz’. Matricea de confuzie a modelului arată ca, acesta este cel mai confuz la detectarea obiectelor dăunatoare. Motivul este folosirea unor imagini de antrenare insuficient de clare.

Scatter chart

Description automatically generated

Figura 52: Tabel matrice de confuzie

* Pierderi

Pe langa acuratețe, pierderea este o masură pentru a afla cat de bine a învațat un model să prezică clasificările corecte, pe un număr dat de mostre. Daca predicțiile sunt zero, modelul este perfect, în caz contrar, pierderile sunt mai mari ca zero. În cazul modelului de față, pierderile nu sunt zero, indicând greșeli la învățarea clasificărilor corecte.

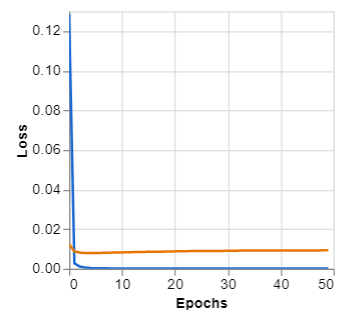


Figura 53: Grafic pierderi

# Concluzii

În concluzie, crearea unei aplicații care presupune viziunea computerizată este o călătorie care presupune o gândire perseverentă, care nu se dă in lături de la a invăța lucruri noi și a trece peste multe obstacole de implementare. Pomenind despre lucruri noi, inteligența artificială și învățarea automată, pun punctul pe i, întrucât posibilitățile sunt inimaginabile, cu cât te aventurezi cât mai tare în lumea inteligentei artificiale, cu atât mai mult iți dai seama de oportunitatea imensă de dezvoltare pe care acest domeniu o prezintă.

La baza aplicației se află limbajul de programare Python. Alegerea a fost făcută datorită stabilității, fexibilității și a uneltelor puse la dispoziție, oferă acces către multe biblioteci și arhitecturi pentru o gamă largă de proiecte. Python are o sintaxă simplă care ofera oportunitate programatorului de a se concentra pe a fi creativ, acest aspect este avantajul principal și motivul pentru care Python este alegerea corectă pentru a implementa aplicații care folosesc inteligența artificială.

Recunoașterea facială, corporală și recunoșterea obiectelor sunt niște descoperiri incredibile în domeniul tehnologiei. Procesul de implementare a acestora m-a înzestrat cu multe cunoștiințe dar ceea ce vreau sa subliniez este faptul ca ele au fost folosite ca o fundație. Asa cum s-a văzut in studiul bibliografic, fiecare aplicație folosea recunoașterile pentru a atinge un scop mai mare: verificarea identității, securizarea tranzacțiilor online, precum si detectarea emoțiilor umane. În acest spirit, aplicația realizată aduce un aport de utilitate prin toate elementele de securitate pe care le prezintă: diferențierea între o persoană cunoscută sau una necunoscută/neautorizată, respectiv între un obiect și un obiect periculos, emițând un semnal de alarmă în conformitate. În completarea ideii de securitate, aplicația dispune de o formă de supraveghere prin care toate detecțiile sunt înregistrate si salvate.

Nu in cele din urmă, vreau să închei prin a reaminti importanța domeniului atins. Aplicația implementată nu este decât o fractiune a puterii și posibilității nesfârsite pe care le deține inteligența artificială. În ciuda aplicabilității limitate a aplicației, experiența formării acesteia mi-a oferit, pe langă găsirea unei pasiuni noi, un bagaj semnificativ de cunoștiințe asupra cum pot fi folosite tehnologiile de învățare automată în avantajul nostru, cum funcționează și care sunt fundațiile din spatele acestora.

## 7.1 Viitoare îmbunătățiri

Aceasta este prima versiune a aplicației, precum concepul de ‚machine learning’ , aceasta trebuie mereu să tindă spre optimizarea continuă, spre funcționalități creative și utile, spre versiuni mai bune. Pentru ca aplicația să progreseze odată cu cererea mare de inovație și dezvoltare a domeniului, vom adăuga noi functionalități și vom optimiza procesele actuale. Acestea sunt dupa cum urmeaza:

* crearea unei baze de date care să stocheze imagini, pentru a înlocui folderul cu imagini ‚ImagesDeposit’ cu o sursă stabilă, mai ușoară de gestionat
* optimizarea procesului de prelucrare a imaginii recunoașterii faciale, în scopul înbunătățirii cadrelor pe secundă când o persoană este detectată
* folosirea unui algoritm nou de recunoaștere facială si corporală
* implementarea unei rețele neuronale proprii, pentru a avea acces la mai multe teste de performanță
* implementarea unei interfețe performate, cu timp de procesare scăzut
* depozitarea resurselor precum imagini și icon-uri într-un folder separat
* salvarea filmărilor capturate într-un folder securizat
* împachetarea aplicației Python într-un mod prin care doar executabilul sa fie accesibil de către utilizator

# Bibliografie

[1]. \*\*\*„A brief history of the security camera system”, Condortech, <https://www.condortech.com/security-camera-systems-washington-dc/a-brief-history-of-the-security-camera-system/> [Accesare 15.05.2022]

[2] \*\*\* „Cpp-programming”, Programiz, <https://www.programiz.com/cpp-programming> [Accesare 20.05.2022]

[3]. \*\*\* „Algoritm Face++”, Faceplusplus, <https://console.faceplusplus.com/documents/6329700> [Accesare 03.06.2022]

[4]. \*\*\* „Python advantages and disadvantages”, Techvidvan(<https://techvidvan.com/tutorials/Python-advantages-and-disadvantages/> [Accesare 5.06.2022]

[5]. \*\*\*„Python opencv tutorial”, Edureka, <https://www.edureka.co/blog/Python-opencv-tutorial/> [Accesare 01.06.2022]

[6]. \*\*\* „What is a deep neural network”, Oticon, <https://www.oticon.com/blog/what-is-a-deep-neural-network-dnn> [Accesare 08/06/2022]

[7]. \*\*\* „Face recognition explained state of the art methods and best tools”, Medium, <https://medium.com/sciforce/face-detection-explained-state-of-the-art-methods-and-best-tools-f730fca16294> [Accesare 20.05.2022]

[8]. Davis E. King. [Dlib-ml: A Machine Learning Toolkit](http://jmlr.csail.mit.edu/papers/volume10/king09a/king09a.pdf). Journal of Machine Learning Research 10, pp. 1755-1758, 2009 [Accesare 25.03.2022]

[9]. \*\*\* „A comprehensive guide to convolutional neural networks”, Towardsdatascience, <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53> [Accesare 01.05.2022]

[10]. \*\*\* „Computer vision”, Ibm, <https://www.ibm.com/topics/computer-vision> [Accesare 13.05.2022]

Anexa 1 – Lista figurilor

Figura 1: Utilități ale inteligenței artificiale în viața cotidiană.....................6

Figura 2: Aplicație de verificare a identității...............................................9

Figura 3: UI OpenFaceTracker..................................................................11

Figura 4: Respectarea comenzilor.............................................................11

Figura 5: Avantaje/dezavantaje Python.....................................................12

Figura 6: Facebook Auto-tag.....................................................................14

Figura 7: Utilitate vedere artificială............................................................15

Figura 8: Structura interfeței și a rulării aplicației......................................16

Figura 9: Pachetele și bibliotecile esențiale..............................................18

Figura 10: Conversia imaginilor test în RGB.............................................19

Figura 11: Primul chip detectat..................................................................19

Figura 12: Cod găsire distanța între fețe...................................................20

Figura 13: Rezultatul false la persoane diferite..........................................21

Figura 14: Cod găsirea valorii distanței......................................................21

Figura 15: Rezultatul True la aceeași persoană.........................................22

Figura 16: Cod afișare pe ecran comparație și distanță.............................19

Figura 17: Afișare pe ecran rezultatul True și distanța...............................22

Figura 18: Începerea codului final...............................................................22

Figura 19: Folderul cu imagini.....................................................................23

Figura 20: Calea către folderul cu imagini...................................................23

Figura 21: Citirea imaginilor de către program............................................24

Figura 22: Cod găsirea codificărilor.............................................................24

Figura 23: Funcția encode preluând locațiile fețelor....................................25

Figura 24: Cod returnare valori: distanța fețelor..........................................26

Figura 25: Valori găsite corespunzator cu numărul de fețe din folder.........26

Figura 26: Cod găsirea celei mai mici valori................................................27

Figura 27: Valori mici la imaginea cu Bill Gates..........................................27

Figura 28: Cod desenare dreptunghi peste chip.........................................27

Figura 29: Corectarea locațiilor...................................................................28

Figura 30: Album recunoașteri cu succes...................................................28

Figura 31: Dificultate în programarea Python.............................................29

Figura 32: Apăsarea tastei q pentru a ieși din bucla infinită.......................30

Figura 33: Notificare alertă recunoaștere facială........................................30

Figura 34: Cod notificare windows..............................................................31

Figura 35: Cod persoană necunoscută sau neautorizată...........................31

Figura 36: Utilizarea aplicației web Teachable Machine.............................31

Figura 37: Exportarea modelului ca un model Keras..................................32

Figura 38: Clasificarea obiectului detectat ca fiind telefon..........................32

Figura 39: Notificare alertă detecție de obiecte...........................................35

Figura 40: Cod notificare windows, detecție de obiecte periculoase...........35

Figura 41: Folderul cu imagini.....................................................................37

Figura 42: Structura folderului aplicației......................................................38

Figura 43: cod buton în Tkinter...................................................................39

Figura 44: Interfața cu utilizatorul................................................................39

Figura 45: Testare numar maxim de recunoaștere per cadru.....................40

Figura 46: Recunoaștere cu succes persoană cu masca...........................40

Figura 47: Recunoastere fara succes persoană cu cagulă.........................41

Figura 48: Eroare în clasificarea obiectelor.................................................42

Figura 49: Exemplu clasa Teachable Machine............................................42

Figura 50: Adăugarea mai multor imagini de antrenare...............................43

Figura 51: Tabel acuratețe model.................................................................44

Figura 52: Tabel matrice de confuzie...........................................................45

Figura 53: Grafic pierderi..............................................................................45

Anexa 2 – Lista tabelelor

Tabel 1: Timpul de procesare raportat la numărul de oameni........................41