Departamentul Automatică şi Informatică Industrială
Facultatea Automatică şi Calculatoare
Universitatea POLITEHNICA din Bucureşti

LUCRARE DE DIPLOMĂ

Reducerea fluxului de date în videoconferințe prin utilizarea IA

Coordonator
Conf. Dr. Ing. Maximilian-Eugen NICOLAE

Absolvent
Mihnea-Cristian MATEI

2023

CUPRINS

ABSTRACT

Scopul prezentei lucrări este crearea unei aplicații capabile să creeze un flux video conținând avatarele utilizatorilor actualizate în timp real în funcție de emoțiile acestora. Această aplicație își propune să crească intimitatea participanților la videoconferință, să crească interacțiunea între aceștia și să reducă fluxul de date necesar unei videoconferințe.

Aplicația este formată din două componente, client și server. La nivel de client este detectată figura utilizatorului, care este decupată și transmisă unui model de recunoaștere a sentimentelor. Recunoașterea sentimentelor se bazează pe expresia facială. În urma determinării sentimentului utilizatorului, componenta client transmite componentei server informația sub formă de șir de caractere. La nivelul serverului, fiecare client este manevrat independent, pentru a putea segrega fluxul de date primit. În funcție de datele primite, este selectat avatarul aferent emoției fiecărui client și este suprapus pe un fundal predefinit.

MULȚUMIRI

Doresc să adresez mulțumiri domnului profesor Conf. Dr. Ing. Maximilian Eugen Nicolae, care mi-a fost îndrumător pe parcursul prezentei lucrări pentru tot sprijinul, ideile și răbdarea pe care mi le-a oferit. Prezenta lucrare este prezentată în cea mai bună formă a sa datorită dânsului.

1. INTRODUCERE

1.1. Domeniul de studiu

Inteligența artificială (prescurtat IA sau AI, în engleză) este o ramură a științei calculatoarelor ce are ca scop crearea de mașini caracterizate prin "inteligență" ce pot executa sarcini care în mod normal ar necesita implicare umană. Datorită avansului tehnologic foarte rapid din ultimii 20 de ani și a volumului mare de date ce se află dispoziția oricui, inteligența artificială a devenit unul dintre domenile momentului. În anul 1943, apare prima lucrare de inteligență artificială, care descrie "neuroni artificiali" ce pot fi folosiți pentru a recunoaște sau decide alte seturi de reguli pentru manipularea datelor. După o perioadă de stagnare și scădere în popularitate, cercetarea inteligenței artificiale primește un nou impuls dat de apariția sistemelor expert, ce aveau ca scop găsirea de soluții specifice pentru probleme specifice.[1]

Inteligența artificială este clasificată în două categorii, în funcție de scopul și de complexitatea sarcinilor de lucru: inteligența artificială slabă sau îngustă și inteligența artificială puternică sau generală[2]. Inteligența artificială slabă este folosită îndeosebi în cadrul sistemelor care au de executat o sarcină clară, cum ar fi recunoașterea și clasificarea de imagini sau recunoașterea vocală. Aceste sisteme folosesc algoritmi de recunoaștere a modelelor de date și sunt antrenate pe seturi mari de informație pentru a executa sarcina cu o acuretețe și o viteză cât mai mari, acestea fiind principalele avantaje ale acestora. Majoritatea aplicațiilor de IA sunt aplicații de inteligență artificială slabă. Inteligență artificială puternică (generală) este folosită în cadrul sistemelor care au capacitatea de a execută un număr mai mare de sarcini, asemănător inteligenței umane, fiind capabile de rezolvarea unor sarcini complexe mai rapid și mai bine decât oamenii. Tehnologia de baza din spatele inteligenței artificiale este învățarea automată[3], care implică antrenarea algoritmilor utilizați pe seturi cât mai mari de date, astfel încât să poată face preziceri sau să ia decizii în baza 'antrenamentelor'. Tipurile de învățare automată variază, în funcție de tehnică folosită, aceasta putând fi asistată, ne-asistată și ranforsată.

Inteligența artificială are potențialul de a aduce schimbări radicale mai multor industrii. Spre exemplu, în industria medicală, inteligența artificială poate fi folosită pentru a analiza cantități mari de date ce conțin informații despre scheme de tratament și în ce context pot fi folosite acestea. În industria financiară, inteligența artificială poate fi antrenată să detecteze fraude și să propună decizii de investiție. În industria comerțului, poate fi folosită pentru a crea o experiență personalizată fiecărui cumpărător, în funcție de nevoile acestuia.[4] Însă în ciuda multelor avantaje pe care IA le oferă, sunt și suficiente semne de întrebare asupra impactului pe care utilizarea intensă a acesteia îl va avea asupra populației. Una dintre cele mai mari probleme ridicate este aceea a desființării anumitor locuri de muncă, dar și a eticii inteligenței artificiale, îndeosebi când se pune problema asupra luării deciziilor de către această.

În prezent, majoritatea aplicațiilor de inteligență artificială folosesc un model de învățare automată numit rețea neurală. Rețelele neurale au scopul de a imita creierul uman. Acestea sunt compuse din noduri conectate între ele, numite neuroni, ce primesc, procesează și transmit

informație. Într-o rețea neurală, informația este primită într-un nivel de intrare, este procesată printr-unul sau mai multe niveluri ascunse de neuroni, după care este produsă o ieșire la un nivel de ieșire. Rețelele neurale sunt capabile să învețe (asistat sau neasitat) o mulțime de tipare complexe și sunt folosite pentru o varietate de aplicații, precum recunoașterea vocală sau de imagini, viziune artificială sau procesarea limbajelor naturale.[5] Câteva dintre tipurile de rețele neurale des întâlnite sunt:

- Rețelele Adversariale Generative (GAN) ce sunt folosite preponderent pentru a genera imagini și videoclipuri foarte realiste, fiind foarte populare în ultima perioadă. Acestea sunt compuse din două rețele neurale: un generator, ce are rolul de a genera informație falsă (inexistentă în realitate), și un discriminator, ce are rolul de a face diferența între informația reală și cea falsă.[6]
- Rețelele neurale de tip Feedforward (FNN), numite și rețele neurale clasice, sunt cel mai simplu tip de rețea neurală și se folosesc în cadrul problemelor de regresie și clasificare. Acestea conțin un set de niveluri ce au rolul de a transforma intrarea in ieșirea dorită.
- Rețelele neurale Convoluționale (CNN) sunt folosite adesea în sarcini de procesare de imagine și video. Acestea constau într-un set de niveluri convoluționale ce extrag caracteristicile intrării cu scopul de a le procesa pentru a produce ieșirea finală.
- Rețelele neurale Recurente (RNN) sunt folosite pentru a procesa date secvențiale precum în cazul recunoașterii vocale sau a procesării de limbaj natural. Acestea folosesc o buclă de feedback care asigură persistența datelor în rețea.[7]

1.2. Context

Pandemia de COVID-19 din anul 2020 a adus schimbări majore în viața noastră cotidiană. În decurs de doar câteva săptămâni, normalitatea pe care o știam s-a schimbat dramatic. Din cauza lipsei de mobilitate la nivel global, comunicarea inter-umană a fost foarte impactată, fapt ce a condus la o creștere uluitoare a utilizării platformelor online de comunicare. Platforme precum Microsoft Teams, Google Meet sau Zoom au văzut o creștere neprevăzută a numărului de utilizatori peste noapte, interacțiunile online prin intermediul acestora ducând la apariția anumitor probleme printre utilizatori. Oboseala cauzată de numeroase videoconferințe la care utilizatorul a fost supus, lipsa intimității spațiului personal atunci când acesta era nevoit sa deschidă camera web și, nu în ultimul rând, problemele de conexiune cauzate de supraîncărcarea rețelelor de internet sunt trei dintre problemele apărute. Prime două probleme au solicitat moduri mai interactive și mai creative de comunicare. Câteva dintre soluțiile cu care dezvoltatorii platformelor destinate videoconferințelor au răspuns sunt integrarea algoritmilor destinați reducerii zgomotelor de fundal pentru utilizatori și introducerea de fundaluri interactive, pentru a crește gradul de intimitate pentru utilizatori. Acestea însă nu sunt de ajuns, întrucât utilizatorii au rămas reticenți la pornirea continuă a camerelor web, iar problema conexiunilor slabe la internet nu a fost abordată, fiind considerată o problemă de infrastructură. Astfel, am decis crearea unei aplicații ce permite utilizatorului să interacționeze cu alți utilizatori într-o manieră creativă și relativ realistă, dar care ii permite

păstrarea intimității în timpul interacțiunii. Aplicația va permite utilizatorului aflat fie într-o teleconferință, fie într-o sesiune de jocuri sau în cadrul unui curs online să păstreze un grad ridicat de intimitate, dar în același timp să afișeze o prezență cât se poate de apropiată de realitate prin avatarul său. În acest fel, utilizatorul va fi mai puțin predispus la oboseala cauzată de videoconferințe, iar spațiul său personal nu va fi expus celorlalți utilizatori.

Un beneficiu substanțial oferit de aplicația propusă este eficiența cu privire la datele transmise pe banda de internet. Această aplicație va scăpa de necesitatea unei conexiuni excelente la internet, întrucât datele transmise vor fi mult mai mici, deoarece aplicația nu va mai transmite un flux video de imagini, ci un simplu șir de caractere care să îl înlocuiască. Acest beneficiu este crucial pentru utilizatorii din zone izolate cu conexiuni slabe la internet și pentru cei a căror rețea este supraîncărcată des.

Într-o lume în care pandemia și-a lăsat adânc amprenta asupra modului de interacțiune cu cei din jurul nostru, această aplicație poate schimba radical modul în care percepem comunicarea la distanță, într-un mod unic și eficient.

1.3. Obiectivul aplicației

Obiectivul aplicației propuse este creșterea interacțiunii umane în cadrul comunicării directe, în timp real cu alți utilizatori, prin utilizarea unui avatar, permițându-i să-și păstreze intimitatea spațiului personal, în timp ce stresul asupra rețelei de internet este diminuat. Prin utilizarea inteligenței artificiale, se dorește crearea unei aplicații ce poate utiliza, pe baza trăsăturilor faciale ale utilizatorului, un avatar al acestuia, pe care îl va putea folosi în cadrul conferinței în care se află, fie că se află într-o teleconferință la locul de muncă, o sesiune de jocuri sau un curs online. Utilizarea acestui avatar va îmbunătăți interacțiunea între utilizatori prin posibilitatea de a controla emoțiile acestuia prin intermediul propriilor expresii faciale. Acest avatar va suplini plictisitoarea fotografie de profil a utilizatorilor pe care aceștia o pot vedea în timpul conferințelor cu un avatar creat pe baza propriilor lor trăsături faciale. Astfel, utilizatorii vor fi angrenați în teleconferință, fiindu-le oferită posibilitatea de a garanta celorlalți participant atenția lor în timp ce se pot bucura de protecția anonimatului.

Prin utilizarea acestor avatare se reduce cantitatea de date ce va fi transmisă către gazda conferinței (server), oferind un beneficiu important utilizatorilor din zone izolate cu o conexiune mai slabă la internet sau utilizatorilor a căror rețea de internet este suprasolicitată. Detecția sentimentelor se va face la nivel de client, iar selecța avatarului corespunzător se va face la nivel de server.

1.4. Descriere scurtă a aplicației

Scopul prezentei lucrări este crearea unei aplicații capabile să creeze un flux video conținând avatarele utilizatorilor actualizate în timp real în funcție de emoțiile acestora. Această

aplicație își propune să crească intimitatea spațiului personal al participanților la videoconferință, să crească interacțiunea între aceștia și totodată să reducă fluxul de date necesar al unei videoconferințe clasice.

Aplicația este formată din două componente principale, client și server. La nivel de client, prin utilizarea unui model de detecție facială bazat pe metoda cascadelor Haar, este detectată figura utilizatorului, care este decupată și transmisă unui model de recunoaștere a sentimentelor. Modelul de recunoaștere a sentimentelor se bazează pe determinarea expresiei faciale și a fost realizat pe baza unei arhitecturi VGGNet16, modificată și tunată pentru a obține performanțe mai bune. Modelul a fost antrenat folosind setul de date FER-2013 modificat pentru a corespunde mai bine sarcinii cerute. În urma determinării sentimentului utilizatorului, componenta client transmite componentei server informația sub formă de șir de caractere, reducând astfel fluxul de informație client-server. La nivelul serverului, fiecare client este manevrat independent, pentru a putea segrega fluxul de date primit. Informațiile sunt introduse într-o coadă. În funcție de datele primite, este selectat avatarul aferent emoției fiecărui client, identificat prin ID, și este suprapus pe un fundal predefinit. Avatarele utilizate sunt imagini predefinite ce reprezintă diferite emoții: fericire, supărare, plictiseală, neutralitate și surprindere. Pentru a crea un flux video cu aceste imagini este folosită biblioteca OpenCV. Serverul poate acționa drept gazdă în cadrul unei videoconferințe ce are loc pe orice platformă bine-cunoscută.

Folosind prezenta aplicație, utilizatorul își păstrează anonimatul și intimitatea, însă crește gradul de interacțiune cu ceilalți utilizatori, totodată reducând încărcarea serverelor destinate videoconferințelor cu date.

2. STUDIU DE PIAȚĂ

Pentru a ne asigura că aplicația propusă este dorită pe piață, ne-am propus efectuarea unui sondaj de opinie cu privire la dorința oamenilor de a vedea o astfel de aplicație sau o astfel de funcționalitate integrată în platformele deja existente. De asemenea, am vrut să înțelegem exact care ar fi avantajele unei astfel de aplicații, dar și temerile oamenilor. Principalele sentimente ce ar trebui detectate au fost determinate cu ajutorul acestui sondaj de opinie.

Conform sondajului, majoritatea persoanelor care au răspuns întrebărilor adresate fac parte din categoria de vârstă 15-30 de ani. Proporția este de 78%, pe locul doi fiind categoria de vârstă 30-45 de ani cu 17%. Așadar, deducem că sondajul de opinie este unul concludent din punct de vedere al persoanelor interogate, întrucât cele două categorii de vârstă prezentate anterior reprezintă majoritatea utilizatorilor platformelor de videoconferințe: elevi, studenți și persoane aflate în perioada definitorie a carierei profesionale. Întrebați care este frecvența utilizării acestor platforme, majoritatea respondenților (58.5%) le folosesc zilnic, 17% dintre aceștia le folosesc la 2-3 zile, iar alte 17 procente afirmă ca folosesc aceste platforme cel putin o dată pe săptămână. Concluzionăm, astfel, ca majoritatea persoanelor aflate în intervalul de vârstă 15-45 de ani sunt cumva dependente de aceste platforme de videoconferințe. Această dependență este, cel mai probabil, cauzată de intesificarea fenomenului "Work from Home", consecință a pandemiei de COVID-19, utilizarea acestor platforme luând locul clasicelor întâlniri de la birou.

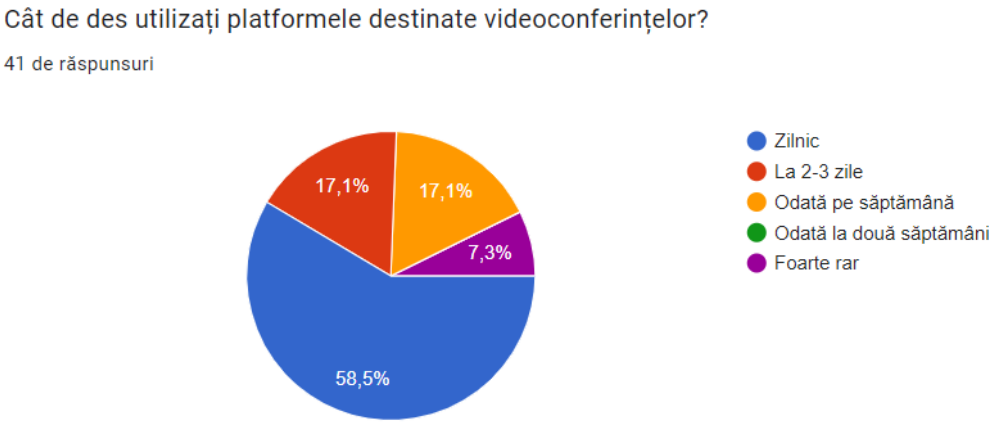

Fig. 2.1. Reprezentare grafică a frecvenței utilizării platformelor de videoconferințe în rândul respondenților

Conform studiului efectuat, cele mai populare platforme de videoconferințe în rândul respondenților sunt Microsoft Teams, într-o proporție de 73.2%, Discord – 58.5% și Google Meet – 53.7%. Toate aceste platforme au în comun flexibilitatea personalizării întâlnirilor. Utilizatorii ce participă la videoconferințe prin intermediul acestor platforme au posibilitatea de a utiliza diferite metode pentru a face utilizarea mai potrivită lor. Printre aceste metode se numără augmentarea mediului înconjurător lor, reducerea zgomotelor de fundal și zgomote de atenționare la diferite evenimente. Însă o majoritate covârșitoare a utilizatorilor, preferă să nu deschidă camera web sau microfonul în cadrul acestor întâlniri din motive de intimitate și conexiune slabă la internet. Peste 85% dintre participanții la sondajul de opinie au afirmat ca folosesc platformele de

videoconferințe la locul de muncă, iar peste 58% dintre aceștia și/sau la școală. Aceste două medii necesită un nivel crescut de interacțiune între participanți pentru a putea avea o perspectivă mai bună asupra impactului întâlnirii la care participă. În cazul situațiilor de videoconferință la locul de muncă, este necesar feedback-ul oferit de expresiile faciale ale participanților pentru a avea un control mai bun asupra impactului unei afirmații. În cadrul videoconferințelor cu scop educațional, feedback-ul elevilor sau studenților participanți la acestea este crucial pentru cadrul didactic al acestora. În funcție de emoțiile și de reacțiile transmise de aceștia profesorul va ști dacă metoda de abordare a subiectului este una corectă sau dacă subiectul este unul de interes.

În urma sondajului de opinie, am aflat că cele mai mari disconforturi pe care utilizatorii le întâlnesc în timpul videoconferințelor sunt cauzate fie de lipsa interacțiunii între utilizatori, fie de conexiuni instabile la internet sau de lipsa feedback-ului. Prin utilizarea prezentei aplicații, traficul de internet în cadrul videoconferințelor ar scădea semnificativ, reducând problema conexiunii la internet, iar prin afișarea, în format digital, a emoțiilor utilizatorilor, va crește substanțial gradul de interacțiune între aceștia.

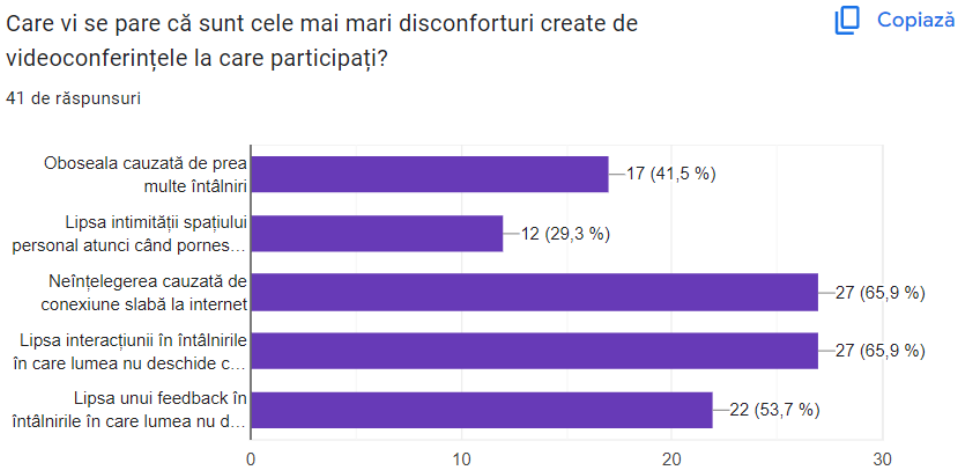

Fig. 2.2. Grafic reprezentând răspunsurile utilizatorilor cu privire la disconforturile avute în cadrul videoconferințelor

Întrebați dacă ar fi interesați de o astfel de aplicație sau de o astfel de funcționalitate, utilizatorii au răspuns în proporție de aproape 93% că ar fi interesați, iar peste 63% dintre aceștia ar utiliza-o des sau foarte des. Aproximativ 26% dintre participanții la sondaj ar utiliza o astfel de funcționalitate moderat, însă niciunul dintre participanți nu a afirmat ca nu ar utiliza o astfel de funcționalitate niciodată.

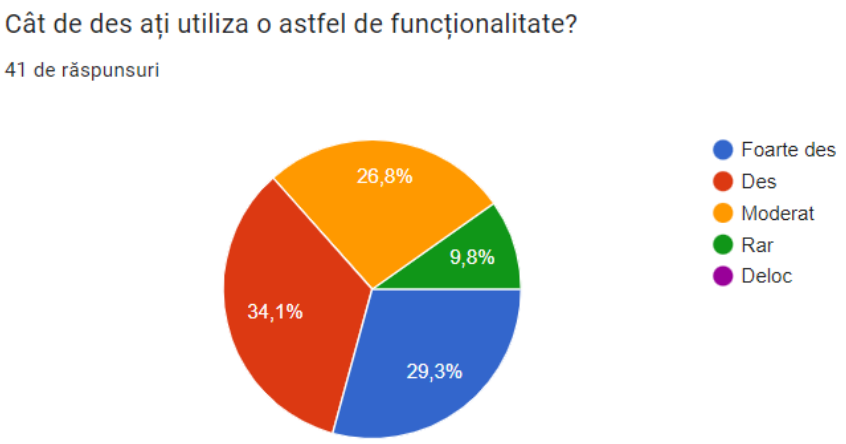

Fig. 2.3. Grafic reprezentând frecvența cu care utilizatorii ar folosi o astfel de funcționalitate/aplicație

Avantajele unei astfel de funcționalități, din perspectiva respondenților, sunt reprezentate în primul rând de creșterea gradului de interacțiune între utilizatori (75.6%) și facilitarea obținerii feedback-ului în cadrul discuțiilor (73.2%). Într-o mare măsură, potențialii utilizatori afirmă că o astfel de funcționalitate ar crește gradul de intimitate, dar și a accesibilității videoconferințelor pentru persoanele aflate în zone izolate, ce au o conexiune mai slabă la internet. Foarte important de menționat este faptul că niciun participant la sondajul de opinie nu consideră că o astfel de funcționalitate nu prezintă niciun avantaj. Conform acestora, principalele emoții pe care o astfel de aplicație ar trebui să fie capabilă să le diferențieze sunt fericirea (87.8%), mirarea (82.9%), plictiseala (80.5%), supărarea (78%) și neutralitatea (70.7%). Astfel, am decis restrângerea sentimentelor căutate la acestea 5, în cadrul implementării aplicației.

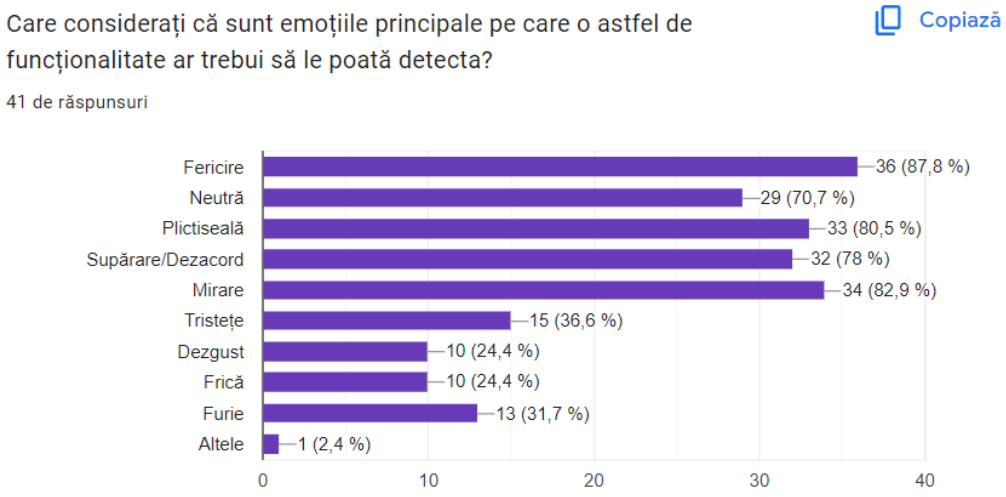

Fig. 2.4. Grafic reprezentând emoțiile relevante sugerate de participanții la sondajul de opinie

3. REALIZĂRI SIMILARE

3.1. Aplicații cu scop similar

Captarea mișcării și modificărilor feței reprezintă un subiect de interes crescând, devenind un punct foarte important în creșterea interacțiunii umane, dar și pentru diferite aplicații în industria auto, industria medicală sau chiar în finanțe, în era digitală. Cu ajutorul aplicațiilor ce se bazează pe informații faciale, utilizatorul se poate exprima mai ușor în mediul online și poate interacționa într-o maniera realistă cu ceilalți utilizatori, poate fi protejat mai bine în situații neprevăzute la volan sau poate avea o securitate crescută a conturilor sale bancare. În general sunt utilizate date provenind de la dispozitivele personale ale utilizatorului, cum ar fi telefonul mobil și computerul, datele fiind extrase cu acordul său.

3.1.1. Animoji & Memoji produse de Apple

Animoji reprezintă o funcționalitate a telefoanelor produse de Apple care permite utilizatorului sa transforme emoticoanele la care are acces în animații scurte, unice, folosind propria expresie facială și voce. Aceste animații se făcute prin scanarea și maparea feței utilizatorului pe emoticonul ales, astfel încât acesta din urmă să poată replica modificările faciale ale celui ce utilizează aplicația. Memoji permite utilizatorului să creeze un emoticon bazat pe trăsăturile lui faciale, în locul celor standard.[8]

Fig. 3.1. Animoji-ul unui editor LifeWire[8]

Pentru realizarea acestor trăsături, Apple folosește kit-ul propriu pentru realitate augmentată, ARKit. Prin intermediul acestui kit, hardware-ul dispozitivelor Apple urmăresc modificările feței utilizatorului în timp real prin detecție facială. Sistemul “TrueDepth” al camerei detectează elemente de adâncime a feței cu ajutorul camerei infraroșu și a iluminatorului încorporat. Software-ul Apple procesează aceste informații și aplică transformările necesare emoticoanelor.[9][10]

3.1.2. Affectiva

Affectiva este o companie americană specializată in detectarea emoțiilor și a percepțiilor umane prin intermediul inteligenței artificiale. Compania utilizează în soluțiile propuse o suită de arhitecturi de învățare adâncă pentru rețele neurale precum rețelele neurale convoluționale multi-disciplinare pentru problemele de regresie și clasificare și rețele neurale recurente, dar și combinații ale acestora. Affectiva oferă soluții pentru mai multe domenii, cum ar fi industria autoturismelor, unde produsul oferit este proiectat astfel încât să îmbunătățească siguranța în trafic prin analiza stărilor emoționale și cognitive ale șoferului, dar și a pasagerilor, numit "In-Cabin Sensing AI". Affectiva oferă produse și pentru analiza impactului reclamelor asupra publicului, folosindu-se tot de informațiile oferite de emoțiile publicului. [11][12]

3.1.3. Metaverse

Metaverse este un univers digital dezvoltat de compania Meta. Scopul produsului este crearea unui univers complet virtual care să mimeze lumea reală. Pentru a intra în acest univers, utilizatorul va avea nevoie de o cască specială, care include ochelari pentru vedere augmentată. În universul virtual, utilizatorul va fi reprezentat de un avatar care va trebui să fie capabil sa reproducă fidel inflexiunile expresiilor faciale ale acestuia, dar și a vocii și a gesticii. Pentru detecția modificărilor faciale și vocale ale persoanelor, vor fi folosite arhitecturi de rețele de învățare adâncă de tip convoluțional și recurent.[13]

3.1.4. Snapchat Lenses

Snapchat Lenses este o funcționalitate a aplicației Snapchat care folosește realitatea augmentată și recunoașterea facială pentru a aplica efecte 3D asupra feței utilizatorilor în timp real. Scopul acestei funcționalități este creșterea interacțiunii umane în mediul digital. Elementul de recunoaștere facială folosește tehnici de învățare automata pentru a crea o hartă a feței utilizatorului cu toate elementele ale acesteia. Odată ce Snapchat are aceste informații, poate aplica elemente de realitate augmentată care urmăresc, practic, fața utilizatorului.[14]

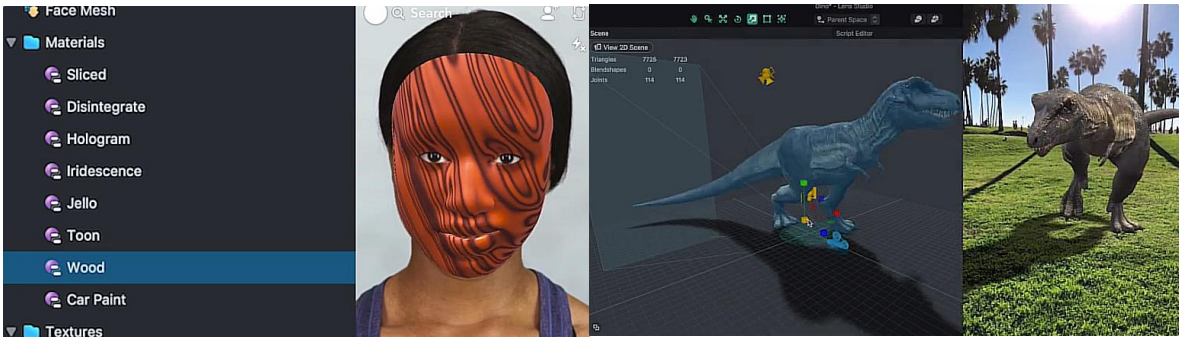

Fig. 3.2. Folosind Snapchat Lenses, în stânga, customizare facială și în dreapta, augmentarea mediului înconjurător[14]

3.2. Utilizarea arhitecturilor VGGNet pentru recunoașterea sentimentelor

Arhitecturile VGGNet, prescurtate VGG (en. Visual Geometry Group), sunt un tip de arhitectură de rețea neurală convoluțională dezvoltată de Universitatea Oxford. Comparativ cu alte arhitecturi similare, arhitecturile VGG se remarcă prin simplitate și prin adâncimea rețelei, cele mai comune rețele având între 16 și 19 straturi adâncime. Aceste arhitecturi folosesc filtre convoluționale de dimensiuni 3x3 și capturează o varietate impresionantă de detalii în imaginile folosite pentru antrenament.[15]

Sarcinile pentru care arhitecturile VGGNet au fost create sunt:

- Clasificarea de imagine, unde sarcina principală este clasificarea imaginilor pe categorii.
- Detecția de obiecte, sarcină la care modelul trebuie sa detecteze anumite obiecte în imagini și să le încadreze pentru a le localiza.
- Clasificarea de imagini cu granularitate ridicată este o sarcină la care rețeaua trebuie să fie capabilă să recunoască diferențele între imagini dintr-o clasă. Spre exemplu, rețeaua trebuie să fie capabilă să recunoască rasele de câini din imagini.[15]

Utilizarea analizei faciale pentru detectarea sentimentelor a devenit din ce în ce mai populară în ultimii, iar noi tehnici de învățare prin transfer au fost elaborate. Rețelele convoluționale adânci au prezentat rezultate foarte bune pentru recunoașterea sentimentelor pe baza expresiilor faciale, însă această sarcina rămâne una destul de complicată din cauza complexității feței umane. Prin intermediul acestor arhitecturi, sunt extrase caracteristicile imaginile prin procesarea acestora până la ultimul strat convoluțional al rețelei. Straturile complet conectate și stratul softmax sunt înghețate, iar în locul lor apar noi straturi complet conectate. La final există un strat softmax ce are ca număr de ieșiri numărul categoriilor de sentimente dorite.[16]

4. CONSIDERENTE TEORETICE ȘI TEHNOLOGII FOLOSITE

Scopul acestui capitol este explicarea noțiunilor teoretice prezente în această lucrare. Cu ajutorul acestor noțiuni, orice persoană care citește prezenta lucrare va înțelege modul în care aceasta funcționează. Vor fi prezentate, pe rând, tehnologiile folosite în cadrul dezvoltării aplicației.

4.1. Arhitectura aplicației

4.1.1. Mediul de dezvoltare și limbajul de programare

Python este un limbaj de programare interpretat de nivel înalt dezvoltat de programatorul olandez Guido van Rossum în anul 1991. Scopul principal al acestui limbaj a fost creșterea nivelului de înțelegere a codului de către oameni și să ofere posibilitatea de a scrie cod mai complex în mai puține linii. Python suportă mai multe paradigme de programare, cum ar fi programarea funcțională, programarea orientată pe obiect sau programarea procedurală.[17]

În ceea ce privește inteligența artificială, Python este cel mai comun limbaj de programare utilizat datorită simplității lui, dar și datorită numărului mare de biblioteci oferite pentru crearea de structuri specifice acestui domeniu. Biblioteci precum Tensorflow, Keras sau PyTorch sunt folosite pentru realizarea de structuri de învățare automată în doar câteva rânduri de cod. Pentru manipularea datelor, Python oferă biblioteci precum Pandas, care permit deschiderea de fișiere ce conțin date (de exemplu fișiere CSV) într-o manieră simplă.

Această flexibilitate vine însă cu un cost asupra performanțelor aplicațiilor scrise în Python, acesta fiind vizibil mai lent decât alte limbaje de programare precum Java sau C++, lucru care poate deveni o problemă atunci când există limitări stricte din punct de vedere hardware.

Alegerea limbajului de programare Python în realizarea prezentei aplicații a fost una simplă, datorită simplității și robusteții sale. Versiunea de Python utilizată în scrierea aplicației este 3.9.13, versiune lansată în anul 2020.

Mediul de dezvoltare principal folosit este Jupyter Notebook. Acest mediu de dezvoltare open-source este un mediu interactiv, bazat pe pagini web, care permite utilizatorului să creeze și să personalizeze documente ce conțin cod. Jupyter Notebook permite utilizatorului să scrie cod în celule bine definite, care pot fi executate individual. Acest mediu de dezvoltare lasă la latitudinea utilizatorului limbajul de programare folosit, însă cel mai adesea este utilizat împreună cu limbajul de programare Python pentru diferite aplicații, dar și pentru aplicații de învățare automată. Caracterul interactiv oferă posibilitatea vizualizării mai ușoare a codului, permițând utilizatorului să experimenteze mai mult. Instalarea Jupyter Notebook este locală și are nevoie de un server pe care să ruleze, acesta fiind, în general, tot local, însă poate fi și la distanță. A fost ales Jupyter Notebook pentru dezvoltarea prezentei aplicații datorită integrării facile a bibliotecilor de învățare automată și mai ales datorită caracterului interactiv pe care acesta îl oferă. [18][19]

În paralel cu mediul de dezvoltare Jupyter Notebook, a fost utilizat, pentru antrenarea rețelei neurale prezente în această lucrare, și mediul de dezvoltare Google Colab, care are la bază tot Jupyter Notebook. Google Colab este un mediu de dezvoltare bazat pe cloud, ce folosește file de tip Jupyter Notebook. Utilizarea Colab pentru antrenarea rețelei neurale a adus mai multe avantaje aplicației, cel mai important fiind posibilitatea de a folosi procesoare grafice în cadrul antrenamentelor. Procesoarele grafice excelează în calculcele pe care antrenarea unei rețele neurale le necesită, scurtând cu până la 90% timpul pe care un procesor clasic l-ar folosi. În cadrul procesului de antrenare a fost utilizat un procesor grafic Tesla T4.

4.1.2. Reprezentarea utilizatorului și detecția facială

Avatarele sunt forme de reprezentare a materialului, fiind adesea folosite pentru a "înlocui" apariția fizică a unor persoane. În prezent, cea mai comună formă a avatarelor este reprezentarea virtuală a anumitor personaje sau persoane în diferite contexte. Aceste avatare pot fi realiste sau nu, în funcție de rolul lor.

Utilizarea avatarelor realiste în cadrul videoconferințelor este un lucru benefic utilizatorilor, deoarece le oferă posibilitatea de a păstra un grad ridicat de anonimat și intimitate și totodată să crească gradul de interacțiune al utilizatorilor. Utilizarea unui avatar ar crește incluziunea membrilor în conferință și ar permite introducerea expresivității fără ca aceștia să fie nevoiți să iasă din zona de confort.[20] Acest mod de abordare ar crește eficiența utilizării datelor în cadrul acestor videoconferințe, prin scăderea cantității de informație necesară pentru o transmisiune bună. Astfel, utilizatorii care trăiesc în medii izolate, cu conexiune slabă la internet, sau utilizatorii din rețele supraîncărcate ar putea participa mai ușor în videoconferințe.

Avatarele utilizate în cadrul acestei aplicații au fost create folosind Bitmoji, o companie subsidiar al Snapchat, și au rolul de a reprezenta utilizatorul în cadrul unei videoconferințe. Această aplicație oferă un grad ridicat de personalizare, fapt ce lasă la latitudinea utilizatorului modul în care acesta va fi reprezentat. Acesta își poate adăuga diferite accesorii și poate selecta o gamă mai variată de expresii faciale. Au fost create mai multe avatare, pentru a reprezenta toate emoțiile necesare aplicației și fost decupate în zona facială.

Detecția facială este o tehnologie provenită din domeniul inteligenței artificiale și din domeniul vederii computerizate ce are scopul de a detecta fețe umane în imagini digitale. Tehnicile folosite pentru detecția facială provin din procesarea de imagine, învățarea automată și învățarea aprofundată sau adâncă. Această tehnologie are numeroase aplicații în viața reală, în domenii precum aplicarea legii, sisteme de securitate, sisteme biometrice și multe altele.

În general, metodele tradiționale de obținere a detecției faciale sunt reprezentate de două metode bine cunoscute, anume metoda Cascadelor Haar și Histograma Gradienților Orientați. Ambele metode sunt foarte sensibile la orientarea feței, având nevoie de o orientare frontală, dar și la variațiile de lumină. Metoda gradienților este sensibilă și la modificările de scară.[21] Recent,

au apărut noi metode pentru efectuarea detecției faciale folosind învățarea aprofundată, prin intermediul rețelelor neurale convoluționale, acestea dovedind o acuratețe net superioară vechiilor metode.

După cum am precizat anterior, provocările principale provin din cauza orientării persoanelor în imagine și din cauza luminii, dar și din cauze de acoperire parțială a feței. O pereche de ochelari de soare poate influența major modul în care modelele percep fața umană.[22]

Datorită vitezei de procesare, dar și a numărului relativ mic de resurse utilizate, în prezenta lucrare metoda de detecție facială folosită este metoda ce utilizează cascadele Haar. Sarcina implică prelucrarea unor imagini provenite de la camera web, în timp ce utilizatorul stă în fața acesteia, așadar limitările acestei metode sunt reduse.

4.1.3. Arhitectura Client-Server

Arhitectura client-server este un model ce reprezintă o aplicație de rețea ce are rolul de a găzdui, livra și organiza resursele sau serviciile solicitate de un client. În acest mod, resursele sunt centralizate într-un singur loc, fapt ce aduce o organizare ușoară a acestora, crește securitatea datelor și permite scalarea sistemului. În general, această arhitectură constă în conectarea mai multor clienți la o mașină numită server.[23]

Serverele sunt sisteme cu rolul de a oferi resurse, date, serivicii sau programe altor calculatoare, numite clienți. În contextul videoconferințelor, serverele au un rol foarte important în menținerea liniei de comunicație între diferite părți aflate în diverse colțuri ale lumii. Pentru ca aceste linii de comunicație să fie stabile, serverele trebuie să fie suficient de puternice și de robuste pentru a putea asigura transmiterea datelor audio-video în timp real. Sarcinile principale ale unui server destinat videoconferințelor sunt procesarea apelurilor, transmiterea datelor audio-video, decodificarea datelor primite și codificarea datelor transmise și securizarea liniei de comunicație.[24] În cazul prezentei aplicații serverul va avea rolul de a primi de la clienți date reprezentând sentimentele acestora și de a le procesa, astfel încât fiecărui utilizator să îi fie atribuit avatarul corespunzător, iar acesta să fie actualizat în timp real. Identificarea clienților la nivel de server se va face utilizând un ID. Transferul de date la server se va efectua folosind protocolul TCP. Serverul a fost creat utilizând biblioteca socket din Python. Componenta de server va putea fi utilizată drept host pentru a partaja avatarele utilizatorilor în cadrul conferințelor ce au loc pe diverse platforme.

În cadrul unei videoconferințe, clientul este software-ul sau hardware-ul care permit unui utilizator să participe la aceasta. Sarcinile clientului sunt de a captura și transmite către server datele audio-video, de a afișa datele audio-video primite de la server și de a oferi utilizatorului o interfață grafică care să îi permită participarea facilă la videoconferință. Securizarea datelor transmise și decodificarea datelor primite sunt efectuate tot de către client.[24] În cadrul prezentei lucrări, clientul va avea rolul de a transmite către server o informație sub forma de șir de caractere,

fiind necesară o prelucrare mai mare a datelor de către acesta. Clientul va fi responsabil de captura datelor, prelucrarea lor și obținerea inferenței de la rețeaua neurală ce va rula local. Acest rezultat îl va transmite serverului.

4.2. Framework și bibliotecile principale folosite

Scopul acestui capitol este descrierea cadrului de lucru pentru realizarea proiectului și descrierea principalelor biblioteci utilizate.

4.2.1. TensorFlow

TensorFlow este un framework ce nu necesită licență, utilizat în dezvoltarea de aplicații de învățare automată dezvoltat de Google Brain. TensorFlow oferă un set de unelte, biblioteci și resurse foarte flexibile care permit dezvoltatorilor și cercetătorilor să construiască și să lanseze aplicații de învățare automată.[25]

TensorFlow este proiectat pentru ușurarea calculelor necesare în cadrul aplicațiilor cu cantități mari de date, adică aplicații cu modele complexe. Printre beneficiile care au dus la alegerea acestui framework se numără:

- Oferirea de biblioteci ce permit un control precis asupra construcției modelului, precum Keras, care oferă simplitate și precizie. Cu ajutorul acestor biblioteci, utilizatorul poate personaliza straturile folosite, funcțiile de cost sau chiar modele pre-definite.[26]
- Posibilitatea de a personaliza buclele de antrenament ale rețelelor pentru un control mai bun asupra procesului de antrenare. Astfel, utilizatorului îi este mult mai ușor să experimenteze cu modelele sale.[27]
- Posibilitatea de a utiliza procesoare grafice sau procesoare tensoriale în procesul de antrenament și de inferență. Prin utilizarea acestor tipuri de procesoare, timpii de antrenament și de inferență scad notabil, eficientizând aceste procese.[28]

Datorită acestor avantaje, TensorFlow este un framework excelent pentru dezvoltarea aplicațiilor de învățare automată și se pretează aplicației propuse.

4.2.2. Biblioteci folosite

Keras este o bibliotecă fără licență de nivel înalt folosită în dezvoltarea rețelelor neurale. Keras este dezvoltată în limbajul de programare Python și poate fi folosită împreună cu framework-ul TensorFlow. Cu ajutorul Keras, utilizatorul poate proiecta și experimenta rețele neurale adânci mult mai simplu, oferind flexibilitate în construcția modelelor prin personalizarea unui număr ridicat de componente ale acestora. Utilizatorul poate opta pentru folosirea anumitor tipuri de straturi neuronale, precum cele convoluționale, cele de înghețare, dense sau de punere în comun. Odată stabilită structura, utilizatorul poate configura procesul de antrenare prin

personalizarea optimizatorului, a funcției de cost sau a metricilor de monitorizare, utilizatorul având posibilitatea să evalueze și să hiper-parametrizeze modelul în urma informațiilor obținute de la procesul de antrenament.[29]

Pandas este o bibliotecă fără licență utilizată pentru analiza datelor și manipularea acestora, oferind structuri de date și funcții pre-definite pentru aceste sarcini. Pentru operațiile matematice, Pandas utilizează biblioteca NumPy, iar pentru vizualizarea grafică a datelor, biblioteca Matplotlib este folosită. Pentru procesarea facilă a datelor, Pandas introduce structuri specifice bazelor de date in Python.[30] În contextul prezentei lucrări, biblioteca Pandas a fost utilizată pentru stocarea și încărcarea datelor, dar și pentru pre-procesarea datelor prin unirea seturilor de date și filtrarea valorilor anumitor câmpuri.

Deoarece sistemele de calcul lucrează mai ușor cu date numerice, datele au fost stocate sub forma de fișier CSV, utilizând biblioteca Pandas. Acestea au fost organizate în numărul stabilit de clase (inițial 8, final 5), fiecare clasă având posibilitatea sa fie de 2 tipuri. La antrenarea rețelei neurale prezentă în această lucrare, datele au fost extrase din fișierul CSV tot prin intermediul bibliotecii Pandas.

OpenCV este o bibliotecă fără licență construită pentru dezvoltarea de software destinat graficii computerizate și pentru software de învățare automată. Biblioteca vine cu numeroși algoritmi care ajută utilizatorul la procesare de imagine, detecție de obiecte, segmentare de imagine, rețele neurale și multe altele.[31]

În cadrul prezentei aplicații, biblioteca OpenCV a fost utilizată cu scopul de a procesa imagini provenite de la camera web și pentru a efectua detecție facială prin intermediul rețelei pre-antrenate bazată pe cascade Haar.

Biblioteca socket este o bibliotecă fără licență disponibilă în Python pentru realizarea conexiunilor la nivel de server și client. În prezenta lucrare, această bibliotecă a fost utilizată pentru crearea soclurilor necesare comunicației dintre client și server, dar și pentru inițializarea conexiunii între aceste componente. Tot prin intermediul soclurilor se efectuează și transferul propriu-zis de date.

4.3. Rețeaua neurală

4.3.1. Noțiuni fundamentale

Noțiunile descrise în rândurile ce urmează sunt noțiuni de bază în domeniul inteligenței artificiale. Aceste definiții sunt aplicabile în cadrul oricărui proiect care implică învățare automată, nefiind specifice unui anumit tip de învățare automată sau a unui anumit tip de rețea neurală.

Neuron – unitatea de bază a unei rețele neurale ce are rolul de a primi date, de a le procesa prin calcule matematice și de a produce un rezultat.

Filtru – mod de organizare al neuronilor în straturi, în cadrul unei rețele neurale convoluționale, ce se remarcă prin capabilitatea de a analiza o imagine întreagă la un singur moment de timp.

Tensor – obiect multidimensional ce conține date.

Strat – organizare specifică rețelelor neurale pentru neuroni, ce pot fi de mai multe tipuri: de intrare, ascuns sau de ieșire. Date sunt primite prin straturile de intrare, sunt procesate prin straturile ascunse, iar predicția specifică modelelor de inteligență artificială se face în stratul de ieșire.

Strat de punere în comun – strat comun în structurile rețelelor neurale convoluționale ce are rolul de a reduce progresiv dimensiunile datelor de intrare. În cazul imaginilor, aceste dimensiuni sunt înălțimea și lățimea. Astfel, se reduce complexitatea calculelor necesare.

Strat de cădere/îngheț – strat de regularizare a rețelelor neurale ce are rolul de a preveni înclinațiile pe care le-ar putea avea rețeaua.

Strat de compresie – strat care are rolul de a converti tensorii multi-dimensionali la o singură dimensiune.

Strat de normalizare în loturi – strat ce are rolul de a stabiliza și îmbunătăți rata de învățare a modelului prin reducerea problemei deplasării covariate interne, ce schimbă distribuția intrărilor din cauza modificării parametrilor straturilor anterioare.

Pondere – parametru ajustabil determinat de rețeaua neurală. Ponderile sunt parametrii deduși pe baza tiparelor datelor de intrare.

Înclinație – parametru care reprezintă tendința modelului spre un anumit rezultat.

Funcție de activare – funcție matematica care, pe baza mediei ponderate a intrărilor și a înclinațiilor, produce o ieșire.

Propagare directă – proces de mutare al datelor în rețeaua neurală. La propagarea directă, datele se mișcă “înainte”, adică de la intrare spre ieșire.

Propagare inversă – proces de învățare al rețelei neurale. În urma propagării directe, rețeaua calculează eroarea dintre ieșire și intrare, reprezentată de diferența dintre predicție și realitate.

Rată de învățare – hiper-parametru care ajută la ajustarea ponderilor și a înclinațiilor.

Rată de descompunere – hiper-parametru ce are rolul de a micșora rata de învățare.

Supraadaptare – rețeaua învață “pe de rost” datele de antrenare și are performanțe scăzute pe date noi.

Subadaptare – rețeaua nu a învățat suficient și are performanțe scăzute atât pe datele de antrenament, cât și pe date noi.

Regularizare – tehnică folosită pentru a preveni supraadaptarea.

Lot – subset al datelor de antrenament.

Epocă – atunci când, la antrenare, rețeaua a realizat o parcurgere completă a setului de date.

Optimizator – algoritm sau metodă pentru ajustarea parametrilor modelului. Optimizatorul este folosit pentru reducerea erorii sau a funcției de cost. Cei mai comuni algoritmi de optimizare sunt SGD (Stochastic Gradient Descent) și Adam (Adaptive Moment Estimation).

Funcție de cost – metodă matematica de a calcula diferența dintre ieșirile prezise de model și ieșirile reale. Funcția de cost măsoară performanțele modelului în raport cu datele de antrenare. În cadrul prezentei lucrări a fost utilizată funcția de cost "Categorical Cross-Entropy", deoarece este cea mai bună variantă pentru problemele de clasificare cu mai mult de două clase.

Fig. 4.1. Clasificare binară - în stânga, subadaptarea rețelei la setul de date, iar în dreapta supraadaptarea [32]

4.3.2. Rețele neurale convoluționale

Rețelele neurale convoluționale sunt rețele folosite în domeniul inteligenței artificiale, în special în cadrul sarcinilor ce constau în procesarea de imagini, recunoașterea anumitor tiparuri sau în cadrul sarcinilor de vedere artificială. Aceste rețele prezintă o eficiență sportiă atunci când vine vorba despre procesarea datelor multi-dimensionale, așa cum sunt imaginile. Straturile ce compun rețelele convoluționale sunt fundația care stă la baza acestora, acestea făcând, de fapt, treaba grea în cadrul rețelei. În sine, rețelele convoluționale sunt asemănătoare cu rețelele neurale clasice: sunt construite din foarte multe entități numite neuroni, organizați în structuri numite filtre, care prezintă ponderi (en. weights) și înclinații (en. biases) antrenabile. Pe baza intrărilor primite, neuronii realizează o operație matematică numită produs scalar asupra unor parametrii și apoi încearcă să introducă neliniaritate asupra rezultatului. Necesitatea introducerii neliniarității provine din faptul că datele din lumea reală sunt arareori liniare, iar tiparele sunt complexe, făcându-le imposibil de prezis de către un sistem liniar. În absența neliniarității, adâncimea rețelei neurale ar deveni irelevantă, deoarece prin compoziția unor funcții liniare, rezultatul va fi tot o funcție liniară.[33]

Practic, în cadrul rețelelor neurale clasice, acestea primesc o intrare, care de obicei este reprezentată de un vector, și printr-o serie de transformări succesive care se produc în interiorul straturilor ascunse de neuroni este generată o ieșire. Uzual, aceste transformări sunt reprezentate de realizarea produsul scalar între parametrii pentru a găsi caracteristicile specifice ale datelor. Toți neuronii sunt interconectați între ei la o rețea neurală clasică, insă nu și la rețelele neurale convoluționale, unde neuronii sunt complet conectați între ei doar la nivelul unui filtru. Un filtru este un set de neuroni interconectați ce au rolul de a mapa caracteristicile unei imagini la un moment dat – adică toată imaginea este încărcată la același moment de timp în filtru. Acest tip de structură funcționează foarte bine când vine vorba despre imagini. Să luăm drept exemplu setul de date CIFAR-10, care conține un set de imagini de dimensiunile 32x32x3. Făcând un calcul simplu, un singur neuron al unei rețele neurale clasice va avea un număr de 3072 de ponderi, număr care reprezintă produsul dimensiunilor imaginii. Chiar dacă acest număr nu pare că ar trebui să dea bătăi de cap rețelei, fiind un număr relativ mic, modul de calcul al acestuia o face. De ce? Pentru că și o simplă creștere a dimensiunii imaginii, la doar 100 de pixeli lățime și 100 de pixeli înălțime, va crește numărul de ponderi la 30000. Trebuie să avem în vedere faptul că o rețea nu este formată dintr-un singur neuron, deci acești parametrii se vor aduna foarte mult.[33] Însă ne ridicăm problema următoare: dacă o rețea clasică ar avea atât de mulți parametrii, atunci o rețea convoluțională, care folosește filtre, va avea și mai mulți. Într-adevăr, însă modul de organizare al neuronilor în filtre care încarcă intreaga imagine la un moment de timp permit eficientizarea modului în care numărul parametrilor variază în funcție de dimensiunea intrării. După cum am precizat anterior, la o rețea neurală clasică, numărul acestora ar crește dramatic, însă datorită organizării în filtre cu dimensiune constantă, numărul parametrilor într-o rețea convoluțională nu va crește, ci va rămâne constant. Astfel, rețelele neurale convoluționale folosesc dimensiunile imaginii ca un avantaj în organizarea neuronilor în cadrul rețelei – înălțime, lățime și numărul de canale, parametru cunoscut drept adâncime. Astfel, se formează straturi convoluționale in trei dimensiuni, față de straturile utilizate de o rețea clasică, care sunt în doar două dimensiuni.

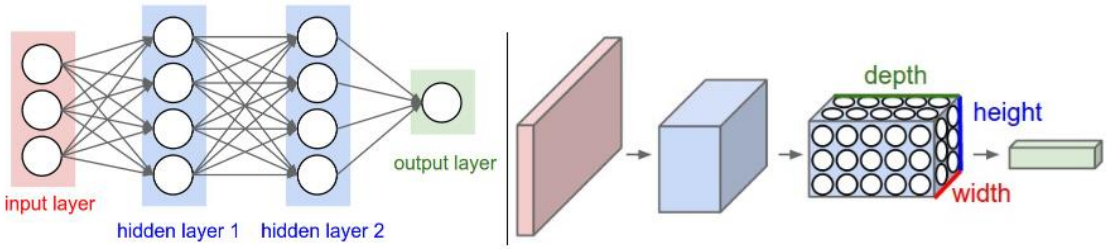

Fig. 4.2. În dreapta avem modul de organizare al unei rețele convoluționale, iar în stânga avem modul de aranjare al unei rețele clasice [33]

5. CONSIDERENTE DE IMPLEMENTARE

În acest capitol vor fi explicate detaliile de implementare ale aplicației. Astfel, vor fi descrise particularitățile lucrării și modul în care acestea au fost elaborate.

5.1. Implementarea rețelei neurale

5.1.1. Structura rețelei neurale

Structura rețelei neurale joacă un rol esențial în dezvoltarea prezentei aplicații. Pentru atingerea unor performanțe cât mai ridicate, au fost testate mai multe structuri de rețele și au fost comparate rezultatele acestora.

Primele structuri testate sunt modelele ResNet. Aceste arhitecturi se diferențiază de alte rețele prin modul în care straturile de ponderi învață informații despre date. Practic, acestea se folosesc de funcții reziduale pentru a face conexiuni între date. Acest model de rețele neurale au crescut dramatic performanțele inteligenței artificiale în materie de sarcini legate de procesarea de imagini. Problema principală pe care această arhitectură a atacat-o este saturarea acurateții în cazul rețelelor foarte adânci, ceea ce duce la degradarea acesteia. Inovația care a fost adusă rețelei pentru a contracara problema degradării este blocul rezidual. Acest bloc rezidual are rolul de a acționa drept scurtătură, în cazul în care performanța rețelei ar scădea ca urmare a trecerii datelor prin următorul strat. Astfel, în cazul acestei scăderi a performanței, modelul sare peste anumite straturi, iar în acest fel se asigură faptul că straturile superioare au performanțe cel puțin la fel de bune ca cele inferioare și în niciun caz mai slabe.[34]

Structurile ResNet testate în cadrul prezentei aplicații au fost ResNet18 – o rețea reziduală de 18 straturi, fiind cea mai mica rețea ResNet, ResNet34 – similară rețelei de 18 straturi, ResNet34 vine cu aproape dublul numărului de straturi și ResNet50 – cea mai frecvent întâlnită rețea ResNet, care vine cu 50 de straturi și introduce un strat de reducere al complexității calculelor (en. bottleneck).

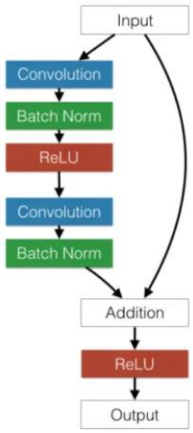

Fig. 5.1. Bloc simplu al unei rețele ResNet [35]

Un alt model de structură testat, reprezentând modelul de bază de la care s-a început proiectarea rețelei finale, este VGGNet – Grup Geometric Vizual (en. Visual Geometry Group). Acest tip de model este mai simplu decât modelele ResNet, însă aceste rețele sunt foarte adânci, adâncimea rețelei fiind caracteristica principală a acestora. Modelele VGG au de obicei între 16 si 19 straturi complet conectate între ele, având o arhitectură omogenă. Deși performanțele rețelei cresc notabil pentru anumite sarcini, rețelele VGGNet folosesc mult mai multe resurse și parametri pentru calcule, acestea fiind mult mai complexe.[15] Uzual, rețelele VGGNet folosesc un număr de 6 blocuri de straturi. Primele 5 blocuri sunt similare între VGG16 și VGG19, diferența principală dintre blocuri constând în creșterea numărului de neuroni prezenți în fiecare strat. În cazul rețelei VGG16, primul bloc conține două straturi convoluționale de dimensiune 64 și un strat de tip "MaxPooling2D", care are rolul de a reduce dimensiunile spațiale ale intrărilor și, deci, complexitatea calculelor efectuate asupra datelor. Numărul neuronilor straturilor următoare se dublează, blocul 2 având straturi convoluționale de 128 de filtre, blocul 3 are straturi de 256 de filtre, blocul 4 de 512 filtre, iar blocul 5 este identic cu blocul 4. Blocul 6 este blocul cel mai dens, avand 4096 de filtre pe fiecare dintre cele două straturi convoluționale. Acest bloc este numit și blocul de clasificare, deoarece la nivelul acestuia se efectuează clasificarea datelor după numărul de clase. Cele două straturi au atribuite o probabilitate de evitare de 0.5. Acest lucru previne supraadaptarea (en. overfitting) modelului la setul de date de antrenament.

Structura folosită în cadrul prezentei aplicații este o variație a modelului VGG16, care folosește 20 de straturi convoluționale. Aceste straturi au fost combinate cu mai multe straturi de tip "MaxPooling2D", dar și cu straturi pentru normalizarea loturilor de date și straturi de cădere.

Inițial, structura folosită a fost antrenată pe 8 clase, reprezentând 8 emoții: fericire, neutralitate, furie, tristețe, dezgust, plictiseală, surprindere și frică. În urma testelor, am constatat ca indiferent de adâncimea rețelei, anumite clase nu vor putea fi surprinse suficient de bine cu setul de date folosit, întrucât acesta prezintă imagini de mici dimensiuni în format scară de gri, fapt ce face detecția anumitor schimbări de nuanță la nivelul trăsăturilor faciale aproape imposibil de detectat. Astfel, au fost identificate clasele care nu erau balansate din punct de vedere al setului de date, dar și care produceau ambiguități si s-a renunțat la acestea, clasele fiind tristețea, dezgustul și frica, antrenarea producând rezultate mult mai bune cu 5 clase de emoție.

Primul bloc este responsabil cu învățarea caracteristicilor de bază ale imaginilor, precum margini sau gradienți. Acesta este compus din două straturi convoluționale cu 64 de filtre cărora li se aplică o normalizare în loturi, pentru îmbunătățirea ratei de învățare. Stratului final i se adaugă și un strat de punere în comun pentru a reduce dimensiunile datelor și pentru a reduce complexitatea calculelor, dar și un strat de cădere cu o rată mică, de 0.2, pentru a reduce posibilitatea apariției unor înclinații nedorite. Pe măsură ce datele avansează, blocurile în care ajung învață caracteristici din ce in ce mai complexe. Spre exemplu, în blocul 2 modelul ar putea învăța să diferențieze texturi, iar în blocul 3 ar putea începe să diferențieze părți ale feței. De aceea și numărul de filtre va crește odată cu avansarea în blocuri, pentru ca modelul să aibă posibilitatea să învețe tipare mai complexe.

Blocul numărul 2 prezintă 3 straturi convoluționale, însă de 128 de filtre fiecare, pentru a putea determina tipare mai complexe, așa cum am precizat anterior. Fiecărui strat i se aplică o normalizare în loturi, iar ultimului strat i se aplica o regularizare prin introducerea unui strat de punere în comun, urmat de un strat de cădere cu o rată de 0.2.

Blocul 3 conține 4 straturi convoluționale de 256 de filtre, fiecare strat fiind urmat de un strat de normalizare în loturi. Se aplică o regularizare prin introducrerea unui strat de punere în comun, urmat de aplicarea unui strat de cădere cu rată de 0.3. Creșterea ratei de cădere este datorată faptului că înclinațiile apar, de obicei, la straturile mai adânci, când modelul începe să învețe tipare mai complexe, ce sunt mult mai greu de identificat.

```
# blocul 3, la nivelul căruia rețeaua începe să învețe tipare mai complexe
conv3_1 = Conv2D(256, kernel_size=3, activation='relu', padding='same', name = 'conv3_1')(drop2_1)
conv3_1 = BatchNormalization()(conv3_1)
# declararea stratului convoluțional conv3_2, cu 256 de filtre și activare ReLU
conv3_2 = Conv2D(256, kernel_size=3, activation='relu', padding='same', name = 'conv3_2')(conv3_1)
conv3_2 = BatchNormalization()(conv3_2)
conv3_3 = Conv2D(256, kernel_size=3, activation='relu', padding='same', name = 'conv3_3')(conv3_2)
conv3_3 = BatchNormalization()(conv3_3)
conv3_4 = Conv2D(256, kernel_size=3, activation='relu', padding='same', name = 'conv3_4')(conv3_3)
# introducerea unui strat de normalizare în loturi asupra stratului convoluțional conv_3_4
conv3_4 = BatchNormalization()(conv3_4)
# aplicarea regularizării prin introducerea unui strat de punere în comun
pool3_1 = MaxPooling2D(pool_size=(2,2), name = 'pool3_1')(conv3_4)
# aplicarea regularizării prin introducerea stratului de cădere
drop3_1 = Dropout(0.3, name = 'drop3_1')(pool3_1)
```

Fig. 5.2. Blocul 3 al rețelei neurale finale, responsabil cu învățarea tiparelor mai complexe – secvență de cod

Blocul 4 al rețelei constă în 4 straturi convoluționale, fiecare având un număr de 256 de filtre. Fiecărui strat al blocului 4 i se aplică un strat de normalizare în loturi, iar ultimului strat din acest bloc i se aplică, pentru regularizare, un strat de punere în comun și un strat de cădere cu o rată de 0.3, identică cu cea a blocului 3.

Blocul 5 conține 2 straturi convoluționale de 512 filtre, care sunt urmate de câte un strat de normalizare în loturi și 2 straturi convolutuinale de 1024 de filtre, urmate, de asemenea, de câte un strat de normalizare în loturi. Ultimului strat i se aplica și o regularizare prin introducerea unui strat de punere în comun, dar și un strat de cădere cu o rată de 0.5, deoarece la nivelul acestui bloc sunt cele mai mari șanse să apară înclinații serioase în model.

La final, se aplică ultimului strat de cădere un strat de compresie, pentru a reduce dimensiunea datelor la o singură dimensiune și este aplicat stratul final, de ieșire, care este de tipul dens, adică complet conectat, cu o dimensiune de 8 ieșiri, respectiv 5 în urma reducerii numărului de clase.

În cadrul tuturor straturilor, cu excepția stratului de ieșire, funcția de activare folosită este ReLU – Rectified Linear Unit. În cadrul ultimului strat a fost folosită funcția de activare SoftMax.

Rectified Linear Unit – ReLU – este un tip de funcție de activare folosită, în general, în rețelele neurale adânci, precum cele convoluționale. Această funcție returnează înapoi orice valoare pozitivă primită, iar în cazul primirii unei valori negative, va returna 0. Fiind o funcție

simplă, nu necesită foarte multă putere de calcul. Deși arată ca o funcție liniară, schimbarea direcției pe care ReLU o suferă în origine (x = 0) o face non-liniară. Deși prezintă numeroase avantaje, poate suferi de o problemă, anume neuronii se pot bloca, producând doar valoarea 0 dacă rata de transfer este prea mare.[36]

$$F(x) = \max(0, x), \text{ unde}$$

max(a, b) este funcția care caculează maximul dintre a și b.

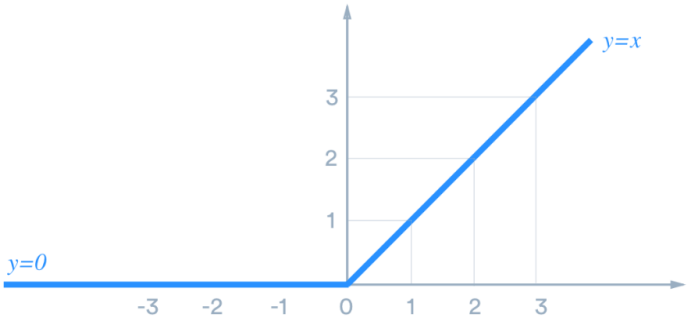

Fig. 5.3. Reprezentarea grafică a funcției de activare ReLU[36]

Softmax este o funcție de activare folosită la straturile de ieșire în cadrul rețelelor neurale. Rolul acestei funcții este acela de a prelua un vector de n dimensiuni ce conține numere reale și de a-l transforma într-un vector ce conține n numere reale în intervalul (0, 1). Ieșirea returnată de funcție este un vector reprezentând distribuțiile de probabilitate ale unei liste de rezultate. Avantajul principal al funcției softmax este faptul că scoate în evidență valorile mari din vector, fiind o funcție ideală în cazul problemelor de clasificare.[37]

$$\sigma(z)_i = \frac{e^{z_i}}{\Sigma_{j=1}^{K} e^{z_j}}, \text{ unde}$$

- z este vectorul intrărilor în funcția softmax
- z_i este valoarea elementului i din vectorul z
- e^{z_i} este funcția exponențială aplicată fiecărui element din vectorul de intrare în funcția de activare
- $\Sigma_{j=1}^{K} e^{z_j}$ este termenul de normalizare, ce se asigură ca toate valorile din vectorul de ieșire vor însuma valoarea 1.
- K este numărul de clase

5.1.2. Setul de date

Setul de date utilizat la antrenarea rețelei neurale folosite are la bază setul FER-2013. FER-2013 este un set de date care constă în 7 clase de imagini ce definesc principalele stări emoționale umane: neutralitate, supărare, furie, dezgust, frică, fericire și surprindere. Setul de date FER-2013

însumează un număr de aproximativ 36000 de imagini în format scară de gri de dimensiune redusă, 48x48, în care fețele sunt plasate aproximativ central și ocupă un spațiu asemănător în fiecare imagine. [38]

Setul inițial a fost modificat pentru a determina cele mai bune rezultate. Inițial, setul de date a fost modificat să conțină aproximativ 42000 de imagini împărțite în 8 clase, cea de-a opta clasă reprezentând starea emoțională dată de plictiseală.

Rezultatele modelelor folosind setul de date cu 8 clase nu au fost însă foarte bune, din cauza lipsei de echilibru din acesta. Prin calcularea acurateții modelului de a prezice fiecare clasă, s-a constatat faptul că modelul avea performanțe foarte slabe pe clasele dezgust, frică și tristețe. Motivul este lipsa mai multor mostre pentru clasele dezgust și frică, iar în cazul clasei tristețe, majoritatea datelor puteau fi prea ușor confundate cu o expresie neutră, ne-existând de multe ori o diferență clară nici pentru un observator uman. Astfel, datele destinate emoției tristețe au fost împărțite către clasa reprezentând expresia facială neutră, iar setul de date a fost redus la 5 clase: fericire, nemulțumire/supărare, neutralitate, plictiseală și surprindere. Setul final de date conține aproximativ 26500 de imagini de antrenament și aproximativ 6500 de imagini de test.

Pentru crearea unor noi imagini a fost utilizat un script, care să faciliteze prelucrarea imaginilor inițiale, de la dimensiuni mari, de 1920x1080x3, imagini care conțineau multe elemente de fundal și zgomot, la dimensiunile imaginilor din setul de date FER-2013, anume 48x48x1.

Fig. 5.4. Exemplu de imagine neprelucrată utilizată la antrenarea detecției stării de fericire

Pentru detecția facială a fost utilizat un fișier de tip .caffemodel, împreună cu un fișier de tip .prototxt. Acest tip de fișiere sunt asociate arhitecturilor de tip Caffe. Modelele bazate pe aceste arhitecturi sunt niște modele ce prezintă performanțe foarte bune în materie de viteză. [39]

Imaginile neprelucrate sunt adăugate într-un folder, în cadrul aceluiași director cu scriptul Python, după care sunt parcurse pe rând. Mai întâi este extrasă o secțiune centrală din imagine,

pentru a reduce posibilitatea ca performanța să fie afectată de elemente de fundal, după care este transmisă modelului Caffe. Dacă modelul reușește să detecteze o față umană în cadrul imaginii, atunci va fi plasată o încadrare în jurul feței, iar imaginea va fi decupată, astfel încât doar fața va rămâne. [40]

```
count = 0
#este deschis directorul in care se afla imaginile neprelucrate
for file in os.listdir(base_dir + '/images/'):
    file_name, file_extension = os.path.splitext(file)
    if (file_extension in ['.jpg','.jpeg']): #este precizata doar extensia .jpg, deoarece cu aceasta se lucreaza.
        image = cv2.imread(base_dir + '/images/' + file)
        (h, w) = image.shape[:2]
        #imaginea este sectionata pentru a reduce elementele de fundal care pot scadea performanta detectiei
        blob = cv2.dnn.blobFromImage(cv2.resize(image, (300, 300)), 1.0, (300, 300), (104.0, 177.0, 123.0))
        #sectiunea este transmisa modelului pentru detecția facială
        model.setInput(blob)
        detections = model.forward()

        # Fața sau fețele sunt identificate si este trasată încadrarea
        for i in range(0, detections.shape[2]):
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])

            (startX, startY, endX, endY) = box.astype("int")
```

Fig. 5.5. Secțiune de cod în care sunt prezentate transformările suferite de o imagine neprelucrată

În urma acestor transformări, noua imagine, care conține secțiunea imaginii în care se află fața persoanei (sau fețele, dacă există mai multe), va fi transformată într-o imagine de dimensiunile 48x48x1, adică o imagine de tip scară de gri. Imaginile procesate vor fi utilizate la antrenarea rețelei convoluționale.

```
input_dir = base_dir + '/faces/'
output_dir = base_dir + '/processed/'

count = 0
# Itereaza fisierul in care se afla imaginile
for file in os.listdir(input_dir):
    #imaginile sunt citite pe rand
    img = cv2.imread(os.path.join(input_dir, file), cv2.IMREAD_UNCHANGED)
    #imaginea este transformata in format grayscale
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    #dimensiunea imaginii este redusa la 48x48
    img2 = cv2.resize(gray, (48, 48))
    #noua imagine este salvata
    print(output_dir + str(count))
    cv2.imwrite(output_dir + str(count) + '.jpg', img2)
    count+=1
```

Fig. 5.6. Secțiune de cod în care este prezentată procesarea finală a imaginii pentru a fi utilizată la antrenarea rețelei convoluționale

După cum știm, însă, calculatorul nu se descurcă foarte bine cu obiecte care nu sunt numere. Așadar, pentru a facilita citirea datelor, am decis transformarea fiecărei imagini într-un obiect uni-dimensional, adică într-un vector. Deși această metoda poate prezenta anumite dezavantaje precum lipsa flexibilității când vine vorba despre augmentarea datelor sau utilizarea excesivă a memoriei (stocarea imaginilor în format compresat de imagine ocupă mai puțină memorie decât stocarea ca vector într-un fișier). După transformarea imaginilor în vectori numerici, a fost stabilit rolul lor în cadrul procesului de antrenare, anume daca imaginea va deservi drept imagine de antrenament sau imagine de test și a fost stabilită și clasa imaginii: neutralitate,

supărare, fericire, surprindere sau plictiseală. Aceste informații au fost adăugate într-un fișier CSV, sub formula clasă, imagine, utilizare.[39][38] Avantajele se rezumă la creșterea semnificativă a vitezei de procesare a datelor, întrucât este evitată deschiderea și închiderea succesivă a unui număr ridicat de fișiere, iar datele sunt direct pre-procesate. De asemenea, setul de date este mai ușor de transportat sub forma unui singur fișier CSV. La final, toate datele introduse sunt amestecate în interiorul fișierului CSV pentru a preveni introducerea accidentală a unor înclinații asupra anumitor clase și pentru a avea o generalizare mai buna a datelor.[40]

```
# toate imaginile din folderul unei clase sunt parcurse
for filename in os.listdir(bored_test_folder):
    # este precizata clasa din care fac parte imaginile
    class_label = '7'

    im = Image.open(os.path.join(bored_test_folder,filename), 'r')
    #sunt extrasi pixelii imaginilor
    pixels = list(im.getdata())
    p_val = ''
    p_val = " ".join([str(item) for item in pixels])
    # se seteaza rolul imaginii in procesul de antrenament
    usage = 'PublicTest'
    # noile date sunt adăugate in listă
    new_data.append([class_label, p_val, usage])

# este citit fișierul CSV
existing_data = pd.read_csv(csv_path)
#se creeaza un dataframe din noile date si se adaugă datelor existente
new_data_df = pd.DataFrame(new_data, columns=['emotion', 'pixels', 'Usage'])
new_data_df.to_csv(csv_path, mode='a', header=False, index=False)
new_data.clear()
```

Fig. 5.7. Secțiune de cod în care este realizată transformarea imaginilor dintr-o clasă în vectori numerici ce vor fi adăugați fișierului CSV

5.1.3. Antrenarea rețelei neurale

Pentru antrenarea rețelei neurale am decis, așa cum am precizat anterior, utilizarea setului de date sub formă de fișier CSV pentru a ușura puțin treaba calculatorului, întrucât va exista un singur fișier de deschis, în locul a aproximativ 33000. Astfel, pentru antrenarea rețelei, prima dată este deschis fișierul CSV conținând setul de date prin intermediul bibliotecii Pandas. Cunoaștem modul de aranjare al datelor din fișier sub forma “emoție, pixeli, utilizare” și dimensiunea imaginilor care au fost stocate acolo, așadar vom defini dimensiunile în două variabile și vom defini și o variabilă care conține numărul de clase utilizate. Se creează o listă cu numele claselor utilizate, dar și un vector numeric al numelor acestora pentru ca mașina să îl înțeleagă, după care sunt citite datele din colona de pixeli de pe fiecare rând în formă numerică și le transforma într-o matrice.

Pentru a organiza datele, vor fi create 4 liste: o listă va conține imaginile de antrenament, o listă va conține rezultatele imaginilor de antrenament, o listă pentru imaginile de test și o listă pentru rezultatele imaginilor de test. Datele corespunzătoare sunt introduse în liste și convertite în format uint8, iar dimensiunea lor este schimbată în 48x48x1. Listele ce conțin rezultatele, atât pentru imaginile de antrenament, cât și pentru cele de test sunt transformate în vectori de clase, adică fiecare etichetă reprezentând un rezultat va fi convertită la un vector binar cu 5 elemente în

care clasa corectă are valoarea 1, iar restul claselor au valoarea 0. Acest tip de transformare se numește codificare “one-hot”.

Pentru a augmenta datele, sunt create două structuri de tip generator de imagine ce au rolul de a aplica diferite transformări datelor. Pentru datele de antrenament am folosit normalizarea valorilor pixelilor între 0 și 1, rotirea imaginilor cu până la 10 grade, întoarcerea imaginilor după axa orizontală, permutarea verticală și orizontală la întâmplare cu până la 10 procente din dimensiuni și umplerea spațiilor goale cu valorile pixelilor vecini. Aceste transformări au rolul de a crește artificial diversitatea setului de date. Astfel, se reduce probabilitatea apariției supraadaptării, iar modelul se va descurca mai bine în cazul în care setul de date nu este echilibrat. Imaginile de test vor avea doar valorile pixelilor normalizate. Datele au fost organizate în loturi de câte 32 de mostre. Modificarea dimensiunii loturilor nu ar aduce beneficii substanțiale, deoarece setul de date este suficient de mare pentru a nu exista probleme de subadaptare cauzate de o dimensiune prea mare a lotului, dar nici probleme de supraadaptare pe care o dimensiune prea mică a lotului ar putea-o cauza.

```
# augmentarea datelor pentru antrenare
datagen = ImageDataGenerator(
    # normalizarea pixelilor imaginilor
    rescale=1./255,
    # rotirea imaginilor cu până la 10 grade
    rotation_range = 10,
    # întoarcerea imaginilor după axa orizontală, la întâmplare
    horizontal_flip = True,
    # permutarea imaginilor cu până la 10% din dimensiuni pe orizontala
    width_shift_range=0.1,
    # permutarea imaginilor cu până la 10% din dimensiuni pe verticală
    height_shift_range=0.1,
    # umplerea golurilor cu valorile cele mai apropiate ale pixelilor
    fill_mode = 'nearest')

# augmentarea datelor pentru validare
testgen = ImageDataGenerator(
    rescale=1./255
    )
```

Fig. 5.8. Secțiune de cod care efectuează augmentarea datelor pentru antrenare și validare

Optimizatorul folosit este Adam, deoarece prezintă performanțe bune pentru acest tip de sarcini, setat la o rată de învățare de 0.0001 și o rată de descompunere de 10^{-6}, aceste valori fiind obținute prin experimentare. Modelul va fi compilat utilizând optimizatorul Adam, precizat anterior, iar metricile folosite vor fi cele specifice bibliotecii Keras împreună cu funcția de cost entropie încrucișată categorică, care este funcția de cost utilizată în cadrul modelelor a căror sarcină este clasificarea multi-clasă. Pentru a reduce rata de învățare periodic a fost setat un organizator din biblioteca Keras, numit “ReduceLROnPlateau”, ce are rolul de a reduce rata de învățare dacă timp de 10 epoci consecutive valoarea acurateții pe datele de validare a modelului nu se îmbunătățește cu o valoare minim setată.

Pentru a putea salva cea mai bună variantă a modelului, acesta este verificat după fiecare epocă, iar dacă ultima valoare a acurateții pe datele de validare nu o depășește pe cea precedentă, modelul nu va fi salvat.

5.2. Implementarea serverului

Pentru implementarea serverului, a fost folosită biblioteca „socket”, pentru a putea crea soclul acestuia. Însă componenta server a aplicației are și rolul de a genera imaginea cu avatare folosită drept flux de date înlocuitor camerelor utilizatorilor. A fost utilizată biblioteca “PyVirtualCam” drept unealtă pentru a putea folosi fluxul respectiv drept flux de cameră web. Pentru generarea imaginii a fost utilizată biblioteca OpenCV. Pentru a putea utiliza avatarele predefinite, a fost declarată calea către acestea la începutul scriptului.

5.2.1. Pornirea serverului și manevrarea clienților

Pentru a porni serverul, mai întâi este creat un soclu de tip flux, pentru ca acesta să poată utiliza protocolul TCP, folosind familia de adrese internet. Serverului îi este atribuită adresa IP a mașinii de pe care rulează, la un port liber. În acest fel, serverul va asculta toate conexiunile efectuate la această adresă. Adresa poate fi modificată cu adresa ‘0.0.0.0’, pentru ca serverul să accepte toate conexiunile efectuate din aria locală, indiferent de interfața de rețea utilizată. Pentru o toleranță mai mare la multiple conexiuni simultane, serverului i-a fost atribuită o coadă de conexiuni de dimensiune 5. Pentru ca serverul să asculte continuu pentru conexiuni noi, este utilizată o buclă fără condiție de ieșire. Când este inițializată o conexiune nouă, este deschis un nou fir de execuție. Fiecare client are firul de execuție propriu, fiind gestionat independent de ceilalți. Gestiunea clienților se face utilizând funcția “handle_client”, care folosește soclul, adresa clientului și ID-ul acestuia. Pentru a începe generarea cadrelor ce conțin avatarele, trebuie ca serverul să detecteze o conexiune deschisă și să observe că firul de execuție destinat generării și afișării acestor cadre nu este pornit. Acest fir de execuție este închis atunci când nu mai există clienți pe server.

Funcția “handle_client” este funcția ce are rolul de a gestiona independent clienții conectați la server. Această funcție primește ca parametrii soclul clienților, adresa IP și ID-ul acestora. Pentru a stoca informațiile despre clienți, este folosită o variabilă globală, iar pentru stocarea sentimentelor transmise în fluxul de date, este folosită o coadă, pentru ca acestea să poată fi introduse în cadrul aferent. La inițializarea conexiunii, este afișat un mesaj care să ateste acest lucru, după care, cât timp clientul transmite serverului informații (conexiunea este activă), mesajul este decodificat, deoarece este transmis în codificare utf-8, și este extrasă informația legată de sentiment, care este mai apoi așezată în coadă. Dacă serverul nu mai primește informații de la client, se consideră că acesta a întrerupt conexiunea, șterge informațiile despre clientul respectiv și închide firul de execuție destinat acestuia.

```python
# variabile globale pentru a stoca date despre client și coada de emoții
clients_info = OrderedDict()
emotion_queue = queue.Queue()

# funcția destinată manevrării clienților
def handle_client(client_socket, address, client_id):
    global clients_info
    print(f"{client_id}: Conectarea de la {address} a avut succes!")
    client_socket.send(bytes("Welcome to the server!", "utf-8"))
    # buclă activă pe parcursul conexiunii
    while True:
        msg = client_socket.recv(64)
        if not msg:
            break
        # informațiile despre sentimentul detectat sunt decodificate
        emotion = msg.decode('utf-8')
        # în informațiile despre client sunt adăugate emoțiile
        clients_info[client_id] = emotion
        # informația despre emoție este adăugată în coadă
        emotion_queue.put((client_id, emotion))
    # la încetarea conexiunii, firul este închis și datele despre client șterse
    del clients_info[client_id]  # Remove the client when it disconnects
    client_socket.close()
```

Fig. 5.9. Secțiune de cod reprezentând funcția “handle_client”, destinată manevrării clienților

5.2.2. Crearea și afișarea fluxului de date pentru camera virtuală

Pentru ca serverul să poată fi utilizat drept gazdă în cadrul acestei aplicații, este necesar ca acesta să poată crea un flux de date care poate fi utilizat drept cameră virtuală. Inițial, este stabilită imaginea folosită ca fundal pentru avatare, dar și dimensiunea acesteia, împreună cu rata de cadre pe secundă care va fi livrată de camera virtuală. Este deschisă o instanță de cameră virtuală, iar cât timp există utilizatori conectați la server care transmit informație despre sentimentele detectate la nivelul client, este inițializat fundalul și o listă de emoții primite. Cât timp există informații despre sentimente în coada destinată acestora, acestea sunt extrase, împreună cu ID-ul clientului care le-a transmis. În funcție de ID-ul clientului se atribuie avatarul aferent unei variabile. Avatarul este redimensionat astfel încât să nu ocupe o dimensiune prea mare și este poziționat pe fundal în funcție de numărul de clienți conectați la server. Momentan, componenta server este realizată astfel încât să suporte patru conexiuni simultane. În urma aplicării avatarului pe imaginea de fundal, cadrul este salvat în format RGB și transmis camerei virtuale.

```python
with pyvirtualcam.Camera(width=width, height=height, fps=fps) as cam:
    # cât timp există conexiuni la server
    while not stop_streaming.is_set():
        background = og_background.copy()
        emotions = {}
        # dacă există informații în coadă, acestea sunt extrase
        if not emotion_queue.empty():
            emotion_data = emotion_queue.get()
            client_id, emotion = emotion_data
            emotions[client_id] = emotion
        # avatarele și fundalul sunt actualizate în timp real, în funcție de informația primită
```

Fig. 5.10. Secțiune de cod în care sunt extrase informațiile din coadă

În figura 5.11. poate fi observat un cadru ce va fi folosit în fluxul generat la nivelul serverului.

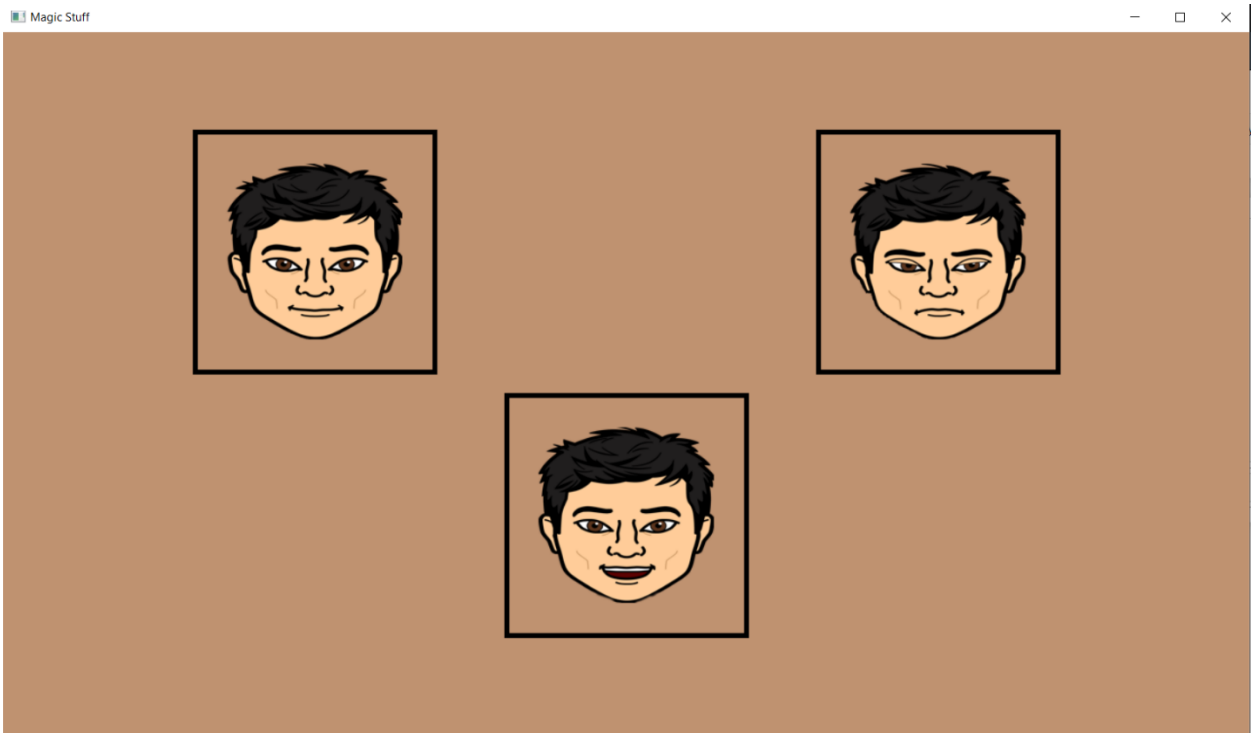

Fig. 5.11. Exemplu de cadru generat la nivelul serverului, conținând trei avatare

5.3. Crearea scriptului pentru client

Pentru realizarea scriptului client, a fost necesară utilizarea mai multor componente, deoarece atât detecția facială, cât și detectarea emoțiilor se face la nivel de client.

5.3.1. Interfața grafică și conectarea la server

Pentru a facilita cât mai mult utilizarea aplicației, la nivel de client a fost introdusă o interfață grafică rudimentară, utilizând biblioteca “Tkinter” din Python. Interfața grafică conține doar 3 butoane, unul pentru conectarea la server, unul pentru deconectarea de la server și un buton pentru a ieși din aplicație. Astfel, utilizatorului îi va fi mai simplă utilizarea aplicației. În urma apăsării butonului ce corespunde conectării la server, este creat un soclu de tip flux, folosind familia de adrese “Internet”. Soclul a fost creat astfel încât să poată fi utilizat pentru un protocol de tip TCP – Transmission Control Protocol, fiind protocolul standard pentru comunicații. Protocolul TCP este un protocol orientat asupra conexiunii, fiind necesară o conexiune între client și server înainte ca un flux de date să poată fi transmis. Pentru ca o conexiune să poată fi stabilită, serverul trebuie să asculte pasiv, adică să fie deschis. Pentru a putea crea o conexiune cu serverul, clientului îi va fi atribuită la nivel de soclu adresa IP și un port de pe care va comunica cu serverul, după care va fi stabilită o conexiune cu serverul prin conectarea soclului client la adresa IP și portul

serverului. Pentru a putea fi identificat la nivel de server mai ușor, clientului îi este atribuit și un ID, care este transmis serverului pentru identificare.

```
# Este declarat soclul client, de tip flux din familia de adrese internet
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# ne asiguram ca soclului i-a fost atribuită adresa ip si portul
s.bind((socket.gethostname(), 1900))
# este initializata conexiunea clientului la server
s.connect(('192.168.56.1', 1234))
# este atribuit id-ul clientului si transmis serverului
client_id = 'client1'
s.send(bytes(client_id, 'utf-8'))
```

Fig. 5.12. Secțiune de cod în care soclul client este declarat și conexiunea cu serverul este inițializată

La momentul părăsirii serverului, conexiunea cu acesta este închisă.

5.3.2. Detecția facială

Pentru a putea oferi modelului de detecție a emoțiilor o imagine cât mai bună, este necesară detecția feței în imagine pentru a putea fi decupată. Pentru performanțe cât mai mari, va fi decupată doar fața din imagine pentru o predicție cât mai bună.

Metoda de detecție facială utilizată este cea a cascadelor Haar, anume modelul de detecție facială frontală. Pentru a determina zonele de interes într-o imagine, au fost introduse mostre de imagini reprezentând caracteristici Haar. Acestea au rolul de a ușura găsirea zonelor în care imaginile prezintă trăsături notabile precum margini sau schimbarea bruscă a valorii pixelilor. Imaginile Haar sunt practic imagini ce sunt formate dintr-un tipar alb-negru. Spre exemplu, o astfel de imagine este un dreptunghi împărțit după axa verticală ce într-o jumătate este alb, iar în cealaltă jumătate este negru. Valorile pixelilor se află între 0 și 1, 0 reprezentând alb, iar 1 negru. Această imagine este suprapusă peste întreaga imagine în care este căutată fața. Prin această suprapunere sunt calculate sumele valorilor pixelilor din regiunea întunecată a imaginii Haar și a pixelilor din regiunea luminată. Aceste sume sunt împărțite la numărul de pixeli ai fiecărei porțiuni pentru a obține media acelei valori. Dacă din valoarea medie a pixelilor din zona întunecată substragem pe cea a pixelilor din regiunea luminată, obținem o valoare între 0 și 1. Dacă această valoare este apropiată de 1, înseamnă ca zona respectivă din imagine conține o trăsătură de interes.[41]

Fig. 5.13. Aplicarea unei trăsături Haar peste o imagine[41]

Pentru a descopei trăsăturile imaginilor, există mai multe tipuri de caracteristici Haar. Toate acestea parcurg imaginea pixel cu pixel pentru un rezultat cât mai corect. Însă toate aceste parcurgeri implică multe calcule, motiv pentru care a fost introdus conceput de "imagine integrală". Această imagine integrală este calculată din imaginea originala, astfel încât fiecare pixel este suma pixelilor din stânga și de deasupra sa. Pixelul din colțul din dreapta jos al imaginii este practic suma tuturor pixelilor din imagine, astfel reducând numărul calculelor necesare la doar 4, deoarece pot fi determinate zone ale imaginii prin poziția pixelului căutat.[41]

Această metodă este una foarte rapidă și eficientă pentru a detecta fețe sau trăsături faciale în imagini, fiind ușor de utilizat împreună cu biblioteca OpenCV. Prin apelarea metodei Haar, este găsită zona în care există o față în imaginea capturată de camera web, iar zona respectivă este "îngrădită" de un dreptunghi pentru a putea fi decupată și procesată pentru modelul responsabil de detecția sentimentelor.

5.3.3. Procesarea imaginii și predicția

În urma detecției faciale efectuate prin intermediul metodei cascadelor Haar care folosește un model pre-antrenat, imaginea este transformată în scară de gri. În cazul în care modelul nu a detectat o figură umană în imagine, emoția umană transmisă serverului este de plictiseală, deoarece înseamnă că, cel mai probabil, utilizatorul nu este în fața calculatorului, deci nu este atent. În cazul în care detecția are rezultat pozitiv, după transformarea imaginii în scară de gri, aceasta este decupată în jurul feței detectate și redimensionată la dimensiunile 48x48. Pentru ca imaginea oferită modelului de detecție sentimentală să fie cât mai bună, valorile pixelilor imaginii decupate sunt convertite la tipul real și normalizate între 0 și 1. Imaginii îi este adăugată încă o dimensiune, aceea de lot, pentru a putea fi procesată de rețeaua neurală, după care este extrasă predicția rezultată în urma inferenței rețelei. Deoarece predicția este sub forma de vector de probabilități, este extrasă din vector valoarea cu probabilitatea cea mai mare și transmisă serverului sub formă de șir.

```
# imaginea este transformată în scară de gri
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
# sunt detectate fețele din imagine
faces = face_classifier.detectMultiScale(gray)
# decizie în funcție de existența feței
if isinstance(faces, tuple):
    s.sendall(b'Bored')
else:
    for (x, y, w, h) in faces:
        # este extrasă regiunea de interes, adică fața
        roi_gray = gray[y:y+h, x:x+w]
        # imaginea conținând fața este redimensionată
        roi_gray = cv2.resize(roi_gray, (48, 48), interpolation=cv2.INTER_AREA)

        if np.sum([roi_gray]) != 0:
            # valorile pixelilor sunt normalizate
            roi = roi_gray.astype('float') / 255.0
            roi = np.expand_dims(roi, axis=0)
            # este executat procesul de inferență
            prediction = classifier.predict(roi)[0]
            # este extrasă probabilitatea maximă din vectorul de predicții
            label = emotion_labels[prediction.argmax()]
```

Fig. 5.14. Secțiune de cod conținând procesul de detecție facială, de pre-procesare și de apelare al inferenței

În figura 5.14. poate fi observată o imagine procesată ce poate fi transmisă rețelei neurale destinate detecției sentimentelor pe baza trăsăturilor faciale.

Fig. 5.15. Imagine procesată asupra căreia poate fi efectuată predicția de către modelul de detecție sentimentală

6. ANALIZA REZULTATELOR

În cadrul acestui capitol vor fi prezentate rezultatele structurii finale, dar și comparativ cu a structurilor ce au condus la aceasta. Întrucât au fost testate mai multe tipuri de rețele, având diferite arhitecturi, vor fi comparate rezultatele acestora. De asemenea, vor fi expuse și rezultatele ratei de transfer comparativ cu transferul de informație video, în cazul unei videoconferințe clasice.

6.1. Rezultatele structurii rețelei finale

Structura rețelei finale este o structură ce trebuie să ofere performanțe bune pentru a putea permite ca modelul să fie folosit într-o aplicație ce rulează în timp real. Astfel, în realizarea acesteia au fost luate în calcul performanțele dorite atât legat de acuratețea rețelei, cât și de timpul de inferență. Având în vedere faptul că aplicația va rula folosind imagini capturate de camera web, trebuie să luăm în calcul rata de cadre cu care aceasta funcționează, anume 30 de cadre pe secundă. Acest număr de cadre duce la un timp de inferență ideal mai mic de 40 de milisecunde pe pas, pentru ca aplicația să poată procesa aproape toate cadrele în timp real.

În urma testelor efectuate, am constatat că timpul de inferență real este de aproximativ 36 de milisecunde, însă dată fiind natura sarcinii, nu este necesară procesarea absolut fiecărui cadru în timp real, iar timpul de inferență obținut este perfect acceptabil.

În timpul procesului de antrenare, a fost urmărită variabila corespunzătoare acurateții, în principal, deoarece în cazul problemelor de clasificare multiplă, cum este aceasta, variabilele de cost nu sunt neapărat relevante, ci mai degrabă este de interes acuratețea modelului. Astfel, variabila în funcție de care a fost salvat modelul cel mai bun este variabila responsabilă de acuratețea pe datele de validare. Curbele obținute în timpul primelor 20 de epoci de antrenament demonstrează abilitatea modelului de a se antrena repede, atât în cazul antrenamentului pe 8 clase distincte, cât și în cazul antrenamentului pe 5 clase distincte.

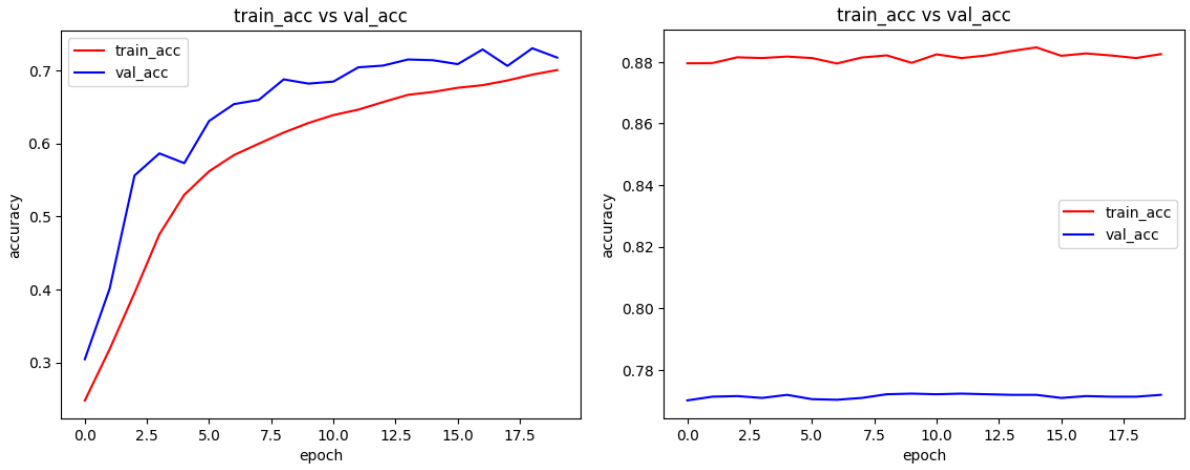

Fig. 6.1. În stânga se poate observa antrenamentul rețelei finale pe setul de date cu 8 clase în primele 20 de epoci, iar în dreapta se poate observa antrenamentul aceleiași rețele în ultimele 20 de epoci

Valoarea maximă a acurateții de validare atinsă în cadrul acestei sesiuni de antrenament a fost de 77.5%, fiind varianta modelului utilizată în testele reale. Însă în urma calculului acurateții medii reale a rețelei, s-a constatat o performanță slabă cauzată de lipsa echilibrului în setul de date. Astfel, s-a constatat o valoare a acurateții medii reale de doar 43.2%, calculată folosind formula:

$$\sigma = \frac{\Sigma_1^k a_k * n_k}{\Sigma_1^k n_k}, \text{ unde}$$

- σ este valoarea acurateții medii reale.
- a_k este valoarea acurateții clasei k
- n_k este numărul de mostre al clasei k
- k este numărul de clase

În urma schimbării setului de date prin renunțarea la clasele ce erau responsabile de această discrepanță, s-a obținut o valoare maximă a acurateții pe setul de date de validare de 84.6%. Modelul respectiv a fost salvat pentru a fi folosit în testarea aplicației. Folosind aceeași formulă descrisă mai sus a fost obținută o acuratețe reală medie de 86.5%, un rezultat net superior celui anterior. Astfel, modelul final este cel capabil să detecteze 5 emoții distincte.

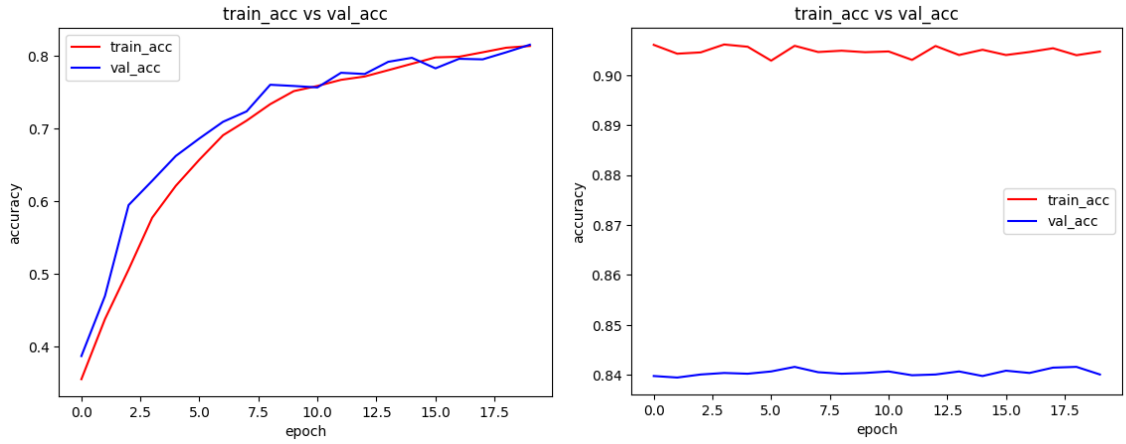

Fig. 6.2. În stânga se poate observa antrenamentul rețelei finale pe setul de date cu 5 clase în primele 20 de epoci, iar în dreapta se poate observa antrenamentul aceleiași rețele în ultimele 20 de epoci

După cum se poate observa în fig. 6.2., în stânga, curbele reprezentante pentru acuratețea de antrenare (roșu) și acuratețea de validare sau test (albastru) sunt curbe aproximativ netede (este normal în cazul validării să apară mici oscilații) cu ascensiune constantă. Acest lucru demonstrează faptul ca nu există e diferență majoră între setul de date de antrenament și cel de validare, imaginile fiind asemănătoare, ceea ce denotă absența supraadaptării modelului la setul de date de antrenament, afirmație întărită de creșterea constantă a acurateții pe parcursul antrenamentului, până la stagnare. Se poate observa în ambele figuri faptul ca la finalul perioadei de antrenare, atât acuratețea de antrenare, cât și cea de validare ating valori aproximativ constante, suficient de mari pentru a exclude o subadaptare a modelului. Lipsa oscilațiilor majore la nivelul curbelor acurateții de validare arată ca dimensiunea setului de date de validare este corespunzătoare.

6.2. Compararea rezultatelor cu alte structuri de rețea testate

Având în vedere schimbarea setului de date de la 8 la 5 clase, compararea rezultatelor va fi efectuată separat între modelele antrenate pe detecția a 8 clase și modelele antrenate pe detecția a 5 clase. Este de menționat faptul că în toate cazurile analizate, în cadrul fiecărui set de date, augmentarea datelor a fost aceeași, iar setul de date nu fost alterat.

6.2.1. Compararea modelelor antrenate pe 8 clase

Primele rețele testate pe modelul de 8 clase au fost cele având arhitecturi ResNet, începând de la cea mai mică rețea de acest fel, până la varianta de 50 de straturi. Curbele de învățare date de valoarea acurateții pe datele de test și de validare nu au fost întocmai netede în cazul ResNet18 și ResNet34. La ResNet50, însă, această curbă a fost netedă și cu creștere aproximativ constantă. Valorile maxime de acuratețe atinse pentru ResNet34 și ResNet50 au fost 69.8%, respectiv 71.1%, niște performanțe mai slabe decât anticipasem, chiar și pentru o sarcină dificilă precum aceasta.

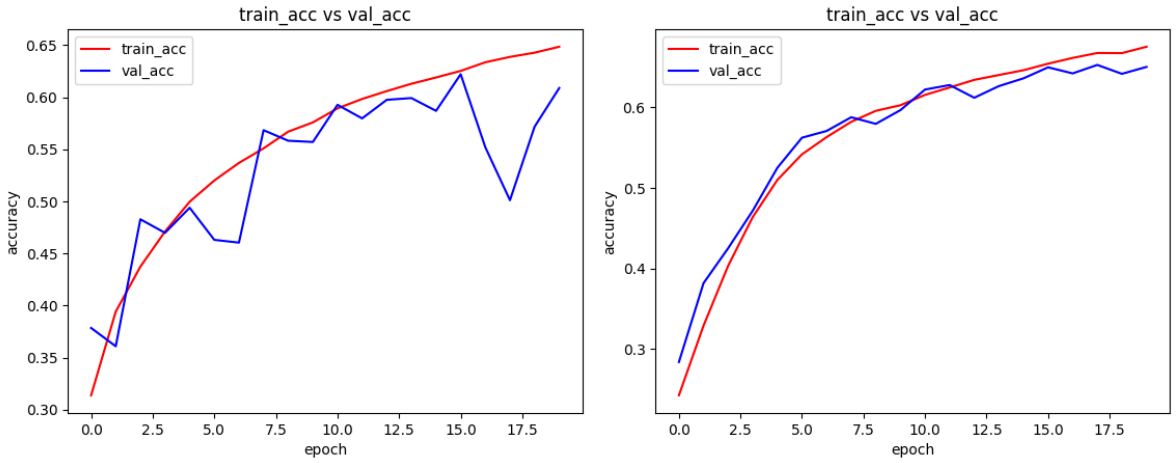

Fig. 6.3. În stânga, curba dată de valorii acurateții din primele 20 de epoci pe datele de antrenament (roșu) și pe datele de validare (albastru) a rețelei ResNet34, iar în dreapta aceleași valori ale rețelei ResNet50

După cum se poate observa în figura 24, în cazul rețelei ResNet34 procesul de antrenare nu este unul întocmai performant, existând fluctuații majore. Astfel, s-a decis schimbarea arhitecturii rețelei cu o arhitectură mai performantă, teoretic, pe astfel de sarcini, anume arhitectura VGGNet.

Inițial, au fost testate ambele structuri cunoscute de VGGNet, atât cea de 16 straturi convoluționale, cât și cea de 19 straturi, însă nu au existat diferențe notabile de performanță la nivelul acurateții, rețeaua cu 16 straturi fiind doar puțin mai rapidă și mai puțin costisitoare. Din punct de vedere al acurateții pe setul de date de validare, arhitecturile VGGNet au atins o valoare maximă de 74%, fiind un rezultat destul de bun. Astfel, s-a decis modificarea ușoară a acestor

arhitecturi pentru a crește performanțele acestora pe sarcina propusă, modelul modificat atingând valoarea maximă a acurateții de 77.5%, fiind o creștere notabilă.

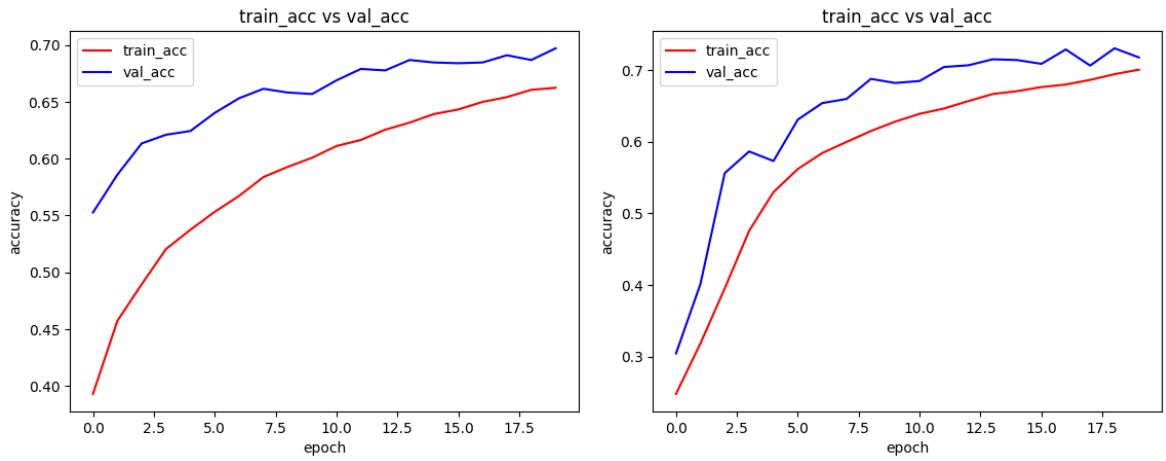

Fig. 6.4. În stânga, variația acurateții de antrenament (roșu) și de validare (albastru) a structurii nemodificate VGG16, iar în dreapta, variația acurateții de antrenament (roșu) și validare (albastru) a structurii modificate

În figura 25, putem observa cum acuratețea rețelelor crește neted (mici oscilații sunt permise) în primele 20 de epoci de antrenament. Valorile înregistrate sunt mai mare decât în cazul rețelelor bazate pe arhitecturi ResNet, încă din primele 20 de epoci. Se poate observa că acuratețea modelului pe setul de date de validare are valori mai mari decât acuratețea pe setul de date de antrenament. Acest lucru se datorează tehnicilor de regularizare folosite. Straturile de cădere folosite au tendința să reducă acuratețea pe setul de date de antrenament din cauza opririi anumitor straturi pentru a reduce supraadaptarea. În cazul validării, aceste straturi sunt complet funcționale.

6.2.2. Compararea modelelor antrenate pe 5 clase

Deoarece performanțele rețelelor antrenate pe 8 clase distincte nu au fost cele dorite, s-a decis verificarea amănunțită a setului de date pentru a identifica posibilele probleme ce ar putea proveni de acolo. Deși acuratețea obținută pe setul de date de validare era una destul de bună pentru sarcina stabilită, performanțele în timp real erau net inferioare. În urma analizei setului de date a fost descoperită o lipsă de echilibru la nivelul anumitor clase, fapt ce a determinat eliminarea acestora. Au fost testate prin antrenarea pe noul set de date structura VGG19 și structura VGG16 modificată, care a avut cele mai bune performanțe pe setul de date anterior. În urma modificării setului de date, numărul mostrelor destinate validării a fost dublat pentru a obține o acuratețe mai apropiată de realitate.

Structura VGG19 a prezentat o acuratețe maxima de 82.8%, stabilizându-se în jurul acelei valori, având un proces de antrenare reprezentat de o curbă netedă cu oscilații minime. Deși procesul de antrenare a durat 100 de epoci, modelul a încetat să se îmbunătățească după primele 50 de epoci, rămânând constant în jurul valorii maxime în ceea ce privește acuratețea pe setul de date de validare.

Modelul final, având la bază structura VGG16, dar modificată, a atins o acuratețe maximă de 84.6%, așa cum a fost precizat anterior, dar și performanțe suficient de bune pentru utilizarea sa în timp real.

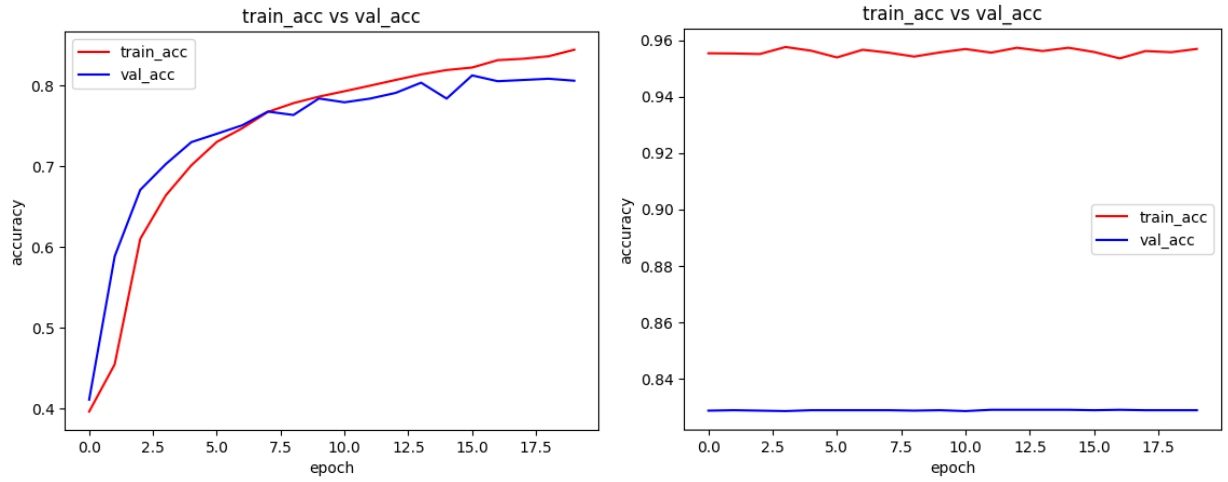

Fig. 6.5. Variația acurateții modelului VGG19 asupra datelor de antrenament (roșu) si a datelor de validare (albastru), în stânga în primele 20 de epoci, iar în dreapta în ultimele 20 de epoci

6.3. Rezultatele ratei de transfer și resursele utilizate

Conform unui studiu publicat în anul 2021 de către cercetători de la Universitatea din Chicago, aplicațiile de videoconferințe au devenit critice în ultimii ani, în special în timpul pandemiei de COVID-19. Această creștere bruscă a ridicat noi întrebări legate de utilizarea acestora în diferite condiții, precum viteză de internet variabilă, calitate ridicată a datelor transmise și cantitatea datelor transmise. Acest studiu a demonstrat că utilizarea aplicațiilor de videoconferință precum Zoom, Google Meet sau Microsoft Teams consumă un volum de date ce variază între 0.8Mbps și 2Mbps și pot consuma până la 75% din lățimea de bandă disponibilă.[42]

Pornind de la acest studiu, a fost realizată o comparație privind transferul de date atunci când există o componentă video în cadrul transmisiunii și atunci când există doar componenta audio. La nivelul aplicațiilor de videoconferință, cele două componente nu doar sunt captate separat, ci sunt și transmise separat. Pentru a realiza acest test, am pornit de la una dintre platformele cele mai utilizate conform sondajului de opinie realizat la începutul acestei lucrări, anume Discord. Pentru a măsura variația ratei de transfer către server, a fost utilizată unealta “Task Manager”, inclusă în setul de unelte Windows, dar și unealta “Resource Monitor”, de asemenea inclusă în setul de unelte Windows. Mai întâi a fost testată conexiunea cu video, ca în cazul unei videoconferințe clasice, iar rezultatul a fost transmiterea unui flux de date de aproximativ 3Mbps.

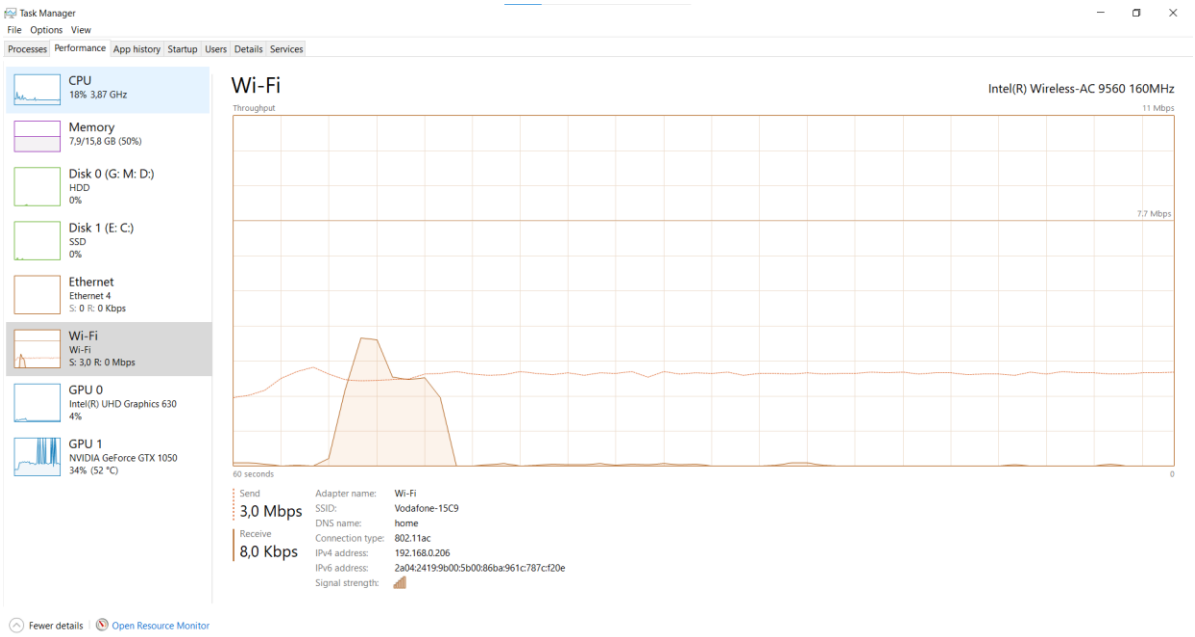

Fig. 6.6. Rezultatul testului în cazul transferului video și audio prin intermediul Task Manager

Pentru a realiza măsurătoarea, a fost deschisă aplicația Discord și inițializat un apel video, după care a fost deschisă aplicația Task Manager, a fost selectată secțiunea “Performanțe”, de unde a fost selectat modulul Wi-Fi pentru afișare. După cum putem observa în figura 6.6, rata de transfer este constantă în jurul valorii de 3Mbps, fără oscilații mari. Această rată de transfer rezultă într-un consum mediu de aproximativ 10Gb pe oră, pe utilizator participant la un apel video.

În cazul apelului în care a fost transmis doar semnal audio, rezultatul este unul surprinzător: rata de transfer a fost în jurul valorii de doar 100Kbps, acest lucru înseamnă de aproximativ 30 de ori mai puține date decât în cazul apelului mixt. Prin prezenta aplicație, clientul ar transmite aceeași valoare audio, iar informația legată de sentimentele detectate, împreună cu ID-ul clientului însumează doar 32Bps, așadar o creștere cu aproximativ 0.032% față de un apel audio și o scădere de aproximativ 30 de ori a cantității de date transmise. Această diferență ar permite tuturor utilizatorilor o implicare egală în cadrul videoconferinței, crescând interacțiunea în cadrul acesteia. Astfel, problemele legate de conexiune slabă la internet semnalate de participanții la sondajul de opinie prezentat la începutul acestei lucrări s-ar reduce substanțial. La o astfel de rată de transfer, consumul de date pe parcursul unei ore ar scădea la o valoare de doar 0.35 Gb pe utilizator, adică o scădere cu 2850% a cantității de date.

Din punct de vedere al resurselor computaționale, în urma deschiderii componentei client a aplicației s-a observat o creștere cu aproximativ 1.3 GHz a frecvenței procesorului și alocarea cu 30% (de la 50% la 80%) mai multe resurse acestuia față de o videoconferință clasică în care fluxul de imagini este direct cel provenit de la camera web. Memoria principală nu este foarte afectată de pornirea acestui script, fiind alocați doar 250Mb în plus, față de o videoconferință normală.

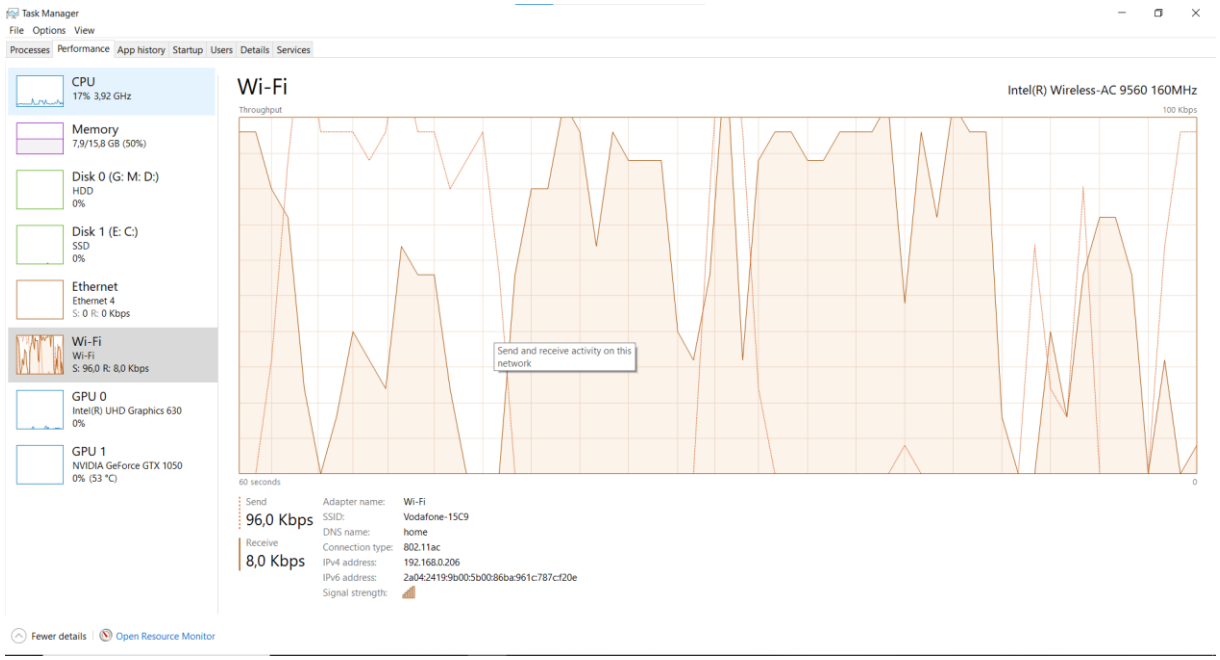

Fig. 6.7. Rezultatul testului în cazul transferului exclusiv audio prin intermediul Task Manager

7. CONCLUZII

Prezenta aplicație prezintă numeroase limitări, dată fiind complexitatea acesteia. Este de precizat faptul că orice limitare descoperită reprezintă un punct de plecare pentru cu posibilă dezvoltare ulterioară.

Orientarea facială a utilizatorului este una dintre limitările principale ale prezentei aplicații. Pentru ca rețelele neurale utilizate în cadrul componentei client să funcționeze eficient, utilizatorul trebuie să fie orientat aproximativ frontal față de camera care captează imagini. În cazul unei orientări diferite, rețelele neurale vor funcționa cu performanțe scăzute. Pentru a îmbunătăți performanțele în aceste cazuri, un pas efectiv ar fi modificarea setului de date. Prin diversificarea setului de date, atât din punct de vedere al tipurilor de emoții detectate, pentru a crește diversitatea emoțiilor ce pot fi afișate în cadrul videoconferinței, cât și prin modificarea diferitor unghiuri ale fețelor în cadrul imaginilor, rețeaua neurală destinată acestei sarcini ar putea produce rezultate mai bune. Similar, diferențele de lumină ar putea impacta negativ performanțele rețelei, deoarece trăsăturile faciale vor fi detectate cu dificultate. Prin creșterea rezoluției imaginilor din setul de date destinat antrenamentului, modelul ar deprinde mai multe caracteristici dintr-o imagine, deoarece în cadrul imaginilor folosite actualmente, de dimensiuni 48x48, numărul trăsăturilor faciale pe care rețeaua le poate învăța este destul de mic, comparativ cu nuanțele pe care o expresie facială le poate avea. Din cauza complexității expresiilor faciale, utilizarea avatarelor pre-definite ar putea fi înlocuită cu o metodă de generare automată a unui avatar cât mai realist, fie prin intermediul unor rețele neurale, fie prin intermediul unei unelte care să aplice transformări automate imaginilor pe care utilizatorul le-ar selecta ca fiind reprezentative lui.

Pentru a putea rula aplicația, utilizatorul are nevoie de multe biblioteci instalate și de limbajul Python, precum și de un mediu de dezvoltare pentru acesta. De asemenea, utilizatorul trebuie să se afle în rețeaua locală din care face parte serverul pentru a se putea conecta la acesta. Acest lucru limitează aplicația la o singură rețea, nefiind accesabilă din afara acesteia. Decizia de menținere a serverului în rețeaua locală a fost luată din considerente de securitate cauzate de deschiderea unui port fără protecție. Pentru a îmbunătăți securitatea aplicației, o soluție ar fi implementarea autentificării prin protocolul SSH (Secure Shell). Acest protocol este utilizat pentru securizarea canalului de comunicație în cadrul rețelelor nesecurizate și folosește o metodă de autentificare prin chei criptate pentru a stabilit autenticitatea utilizatorului și a serverului. Astfel, doar utilizatorii autorizați se pot conecta la server, totodată păstrând confidențialitatea și integritatea datelor transmise. Introducerea unui astfel de protocol nu este, însă, suficientă, fiind necesară și o configurare de firewall pentru a limita conexiunile pe portul respectiv. Deschiderea unei rețele virtuale private, prin intermediul căreia utilizatorii să se poată conecta la server, constituie o metodă eficientă de a limita accesul la server doar persoanelor autorizate.

Pentru performanțe mai bune ale modelului final cuantizarea acestuia ar putea îmbunătăți acuratețea pe datele de validare, deoarece modelul ar generaliza mai bine. Totodată, dimensiunile acestuia ar scădea, iar resursele folosite în cadrul inferențelor ar fi mai puține.

Prezenta aplicație permite utilizatorului ce participă la o videoconferință să își păstreze intimitatea și totodată să mențină un nivel crescut de interacțiune cu ceilalți utilizatori. Acesta va oferi feedback indirect celorlalți participanți la videoconferință prin intermediul expresiilor sale faciale care definesc diferite emoții. Prin utilizarea acestei aplicații, cantitatea de date necesară unei videoconferințe este redusă substanțial, oferind o deschidere mai bună utilizatorilor cu viteză de internet redusă.

Prin dezvoltarea prezentei aplicații au fost dobândite cunoștințe atât în domeniul inteligenței artificiale, cât și în domeniul transmisiilor de date. În cadrul dezvoltării rețelei neurale destinate recunoașterii sentimentelor faciale, au fost deprinse cunoștințe legate de modul de abordare al diferitelor probleme de inteligență artificială, în funcție de sarcina propusă. Pentru ca rețeaua să răspundă cât mai bine unei astfel de sarcini complexe, a fost studiată structura rețelelor neurale convoluționale pentru a găsi cel mai bun model care să învețe caracteristicile imaginilor ce conțin fețe umane în diverse ipostaze. Au fost testate mai multe arhitecturi de rețele convoluționale adânci pentru a determina care dintre acestea au cele mai bune rezultate pentru sarcina propusă, descoperind că nu toate arhitecturile de rețele convoluționale adânci răspund la fel de bine la aceeași sarcină. O astfel de clasificare cu mai multe clase a necesitat adâncirea unor structuri bine-cunoscute, precum rețelele de tip VGGNet, dar și aplicarea unor tehnici de regularizare a datelor din interiorul rețelei pentru a evita probleme de supraadaptare la setul de date și pentru a ne asigura că rețeaua învață cât mai eficient.

Prin crearea scripturilor de client și de server, au fost dobândite cunoștințe valoroase în cadrul domeniului de transmisii de date. Au fost înțelese noțiuni cu privire la modul de funcționare al videoconferințelor și cum fiecare utilizator transmite datele cu privire la voce și video, separat, către un server pentru a putea fi centralizate și sincronizate. Această transmisie se realizează, de regulă, prin intermediul protocolului TCP, care a fost utilizat și în prezenta aplicație în scop similar. Pentru conectarea componentelor client și server, a fost necesară înțelegerea noțiunii de soclu (en. socket) și rolul lui în cadrul acestei arhitecturi. A fost necesară înțelegerea modului de funcționare a acestui tip de arhitectură pentru a dezvolta o aplicație cât mai eficientă.

A fost îmbunătățită experiența de lucru cu limbajul de programare Python și cu librăriile specifice domeniilor din care face parte această aplicație.

8. BIBLIOGRAFIE

[1] Russell Stuart, Norvig Peter (2010), "Artificial Intelligence: A Modern Approach (3rd ed.)". Disponibil: https://people.engr.tamu.edu/guni/csce421/files/AI_Russell_Norvig.pdf Accesat: 2023

[2] Analytics Vidhya (2023), "Weak AI vs Strong AI: Exploring Key Differences and Future Potential of AI". Disponibil: https://www.analyticsvidhya.com/blog/2023/04/weak-ai-vs-strong-ai/ Accesat: 2023

[3] Sara Brown (2021), "Machine learning, explained". Disponibil: https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained Accesat: 2023

[4] Forbes (2023), "Applications of Artificial Intelligence Across Various Industries". Disponibil: https://www.forbes.com/sites/qai/2023/01/06/applications-of-artificial-intelligence/?sh=7b6732fb3be4 Accesat: 2023

[5] IBM (2023), "What are Neural Networks?". Disponibil: https://www.ibm.com/topics/neural-networks Accesat: 2023

[6] Jason Brownlee (2019), "A Gentle Introduction to Generative Adversarial Networks (GANs)". Disponibil: https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/ Accesat: 2023

[7] Ian Goodfellow, Yoshua Bengio, Aaron Courville (2016), "Deep Learning". Disponibil: https://www.deeplearningbook.org Accesat: 2023

[8] Sam Costello (2022), "What are Animoji and Memoji?". Disponibil: https://www.lifewire.com/animoji-4153078 Accesat: 2023

[9] Elyse Betters Picaro (2022), "Apple ARKit explained: Everything you need to know about Apple's augmented reality platform". Disponibil: https://www.pocket-lint.com/ar-vr/news/apple/141615-apple-ar-kit-explained/ Accesat: 2023

[10] Hannah Davies (2022), "What is a TrueDepth camera? All about the iPhone camera feature". Disponibil: https://www.trustedreviews.com/explainer/what-is-a-truedepth-camera-4263845 Accesat: 2023

[11] MZ (2020), "Affectiva: building AI that reads human emotions". Disponibil: https://d3.harvard.edu/platform-digit/submission/affectiva-building-ai-that-reads-human-emotions/ Accesat: 2023

[12] Ashley McManus (2021), "The Evolution of the Automotive In-Cabin Sensing Market". Disponibil: https://blog.affectiva.com/the-evolution-of-the-automotive-in-cabin-sensing-market Accesat: 2023

[13] metamandrill (2023), “Metaverse Avatar Guide; Emobdy Yourself in the Metaverse”. Disponibil: https://metamandrill.com/metaverse-avatar/ Accesat: 2023

[14] Snapchat (2023), “AR Lens Studio”. Disponibil: https://ar.snap.com/lens-studio Accesat: 2023

[15] Karen Simonyan, Andrew Zisserman (2015), “Very Deep Convolutional Networks for Large-Scale Image Recognition”. Disponibil: https://arxiv.org/abs/1409.1556 Accesat: 2023

[16] Gaurav Meena, Krishna Kumar Mohbey, Ajay Indian, Sunil Kumar (2022), “Sentiment Analysis from Images using VGG19 based Transfer Learning Approach”. Disponibil: https://www.sciencedirect.com/science/article/pii/S1877050922007888 Accesat: 2023

[17] Python (2023), “What is Python? Executive Summary”. Disponibil: https://www.python.org/doc/essays/blurb/ Accesat: 2023

[18] Jupyter Notebook (2015), “Project Jupyter Documentation”. Disponibil: https://docs.jupyter.org/en/latest/ Accesat: 2023

[19] Thomas Kluyver et al. (2016), “Jupyter Notebooks”. Disponibil: https://eprints.soton.ac.uk/403913/1/STAL9781614996491-0087.pdf Accesat: 2023

[20] Adam Rowe (2023), “Virtual Avatars are rolling out on Microsoft Teams”. Disponibil: https://tech.co/news/virtual-avatars-rolling-out-microsoft-teams Accesat: 2023

[21] Paul A. Viola, Michael J. Jones (2004), “Robust Real-Time Face Detection”. Disponibil: https://www.researchgate.net/publication/220660094_Robust_Real-Time_Face_Detection Accesat: 2023

[22] K. Zhang, Z. Zhang, Z. Li, Y. Qiao (2016), “Join Face Detection and Alignment using Multi-Task Cascaded Convolutional Networks”. Disponibil: https://arxiv.org/abs/1604.02878 Accesat: 2023

[23] John Terra (2023), “What is Client-Server Architecture? Everything you should know”. Disponibil: https://www.simplilearn.com/what-is-client-server-architecture-article Accesat: 2023

[24] Shanika Wickramasinghe (2022), “How does video conferencing work?”. Disponibil: https://cloudinfrastructureservices.co.uk/how-does-video-conferencing-work/ Accesat: 2023

[25] TensorFlow (2022), “Create production-grade machine learning models with TensorFlow”. Disponibil: https://www.tensorflow.org Accesat: 2023

[26] TensorFlow (2023), “Making new layers and models via subclassing”. Disponibil: https://www.tensorflow.org/guide/keras/making_new_layers_and_models_via_subclassing Accesat: 2023

[27] TensorFlow (2023), “Writing a training loop from scratch”. Disponibil: https://www.tensorflow.org/guide/keras/writing_a_training_loop_from_scratch Accesat: 2023

[28] TensorFlow (2023), “Distributed training with TensorFlow”. Disponibil: https://www.tensorflow.org/guide/distributed_training Accesat: 2023

[29] Keras (2022), “About Keras”. Disponibil: https://keras.io/about/ Accesat: 2023

[30] pandas (2023), “Intro to data structures”. Disponibil: https://pandas.pydata.org/pandas-docs/stable/user_guide/dsintro.html Accesat: 2023

[31] OpenCV (2023), “Introduction”. Disponibil: https://docs.opencv.org/master/d1/dfb/intro.html Accesat: 2023

[32] Tigran P. (2022), “Overfitting and underfitting in Machine Learning”. Disponibil: https://www.superannotate.com/blog/overfitting-and-underfitting-in-machine-learning Accesat: 2023

[33] Stanford University (2023), “CS231n: Convolutional Neural Networks for visual Recognition”. Disponibil: https://cs231n.github.io/convolutional-networks/#conv Accesat: 2023

[34] K. He, X. Zhang, S. Ren, J. Sun (2015), “Deep Residual Learning for Image Recognition”. Disponibil: https://arxiv.org/abs/1512.03385 Accesat: 2023

[35] Satyam Kumar Singh (2020), “ResNet Architecture: Deep Learning with PyTorch”. Disponibil: https://pub.towardsai.net/resnet-architecture-deep-learning-with-pytorch-19ecb7ca359e Accesat: 2023

[36] Vivek Praharsha (2023), “ReLU (Rectified Linear Unit) Activation Function”. Disponibil: https://iq.opengenus.org/relu-activation/ Accesat: 2023

[37] Thomas Wood (2023), “Softmax function”. Disponibil: https://deepai.org/machine-learning-glossary-and-terms/softmax-layer Accesat: 2023

[38] Manas Sambare (2020), “FER-2013”. Disponibil: https://www.kaggle.com/datasets/msambare/fer2013 Accesat: 2023

[39] Yangqing Jia, Evan Shelhamer et al. (2014), “Caffe: Convolutional Architecture for Fast Feature Embedding”. Disponibil: https://arxiv.org/abs/1408.5093 Accesat: 2023

[40] Karan Bhanot (2019), “Extracting faces using OpenCV Face Detection Neural Network”. Disponibil: https://towardsdatascience.com/extracting-faces-using-opencv-face-detection-neural-network-475c5cd0c260 Accesat: 2023

[41] Mirko Stojiljkovic (2019), “pandas: How to Read and Write Files”. Disponibil: https://realpython.com/pandas-read-write-files/ Accesat: 2023

[40] Jason Brownlee (2019), “Embrace Randomness in Machine Learning”. Disponibil: https://machinelearningmastery.com/randomness-in-machine-learning/ Accesat: 2023

[41] Girija Shankar Behera (2020), “Face Detection with Haar Cascade”. Disponibil: https://towardsdatascience.com/face-detection-with-haar-cascade-727f68dafd08 Accesat: 2023

[42] K. MacMillan, T. Mangla, J. Saxon, N. Feamster (2021), “Measuring the Performance and Network Utilization of Popular Video Conferencing Applications”. Disponibil: https://www.researchgate.net/publication/351990830_Measuring_the_Performance_and_Network_Utilization_of_Popular_Video_Conferencing_Applications Accesat: 2023