

UNIVERSITATEA "ALEXANDRU-IOAN CUZA" DIN IASI

FACULTATEA DE INFORMATICA



LUCRARE DE LICENTA

MoodLands - platformer interactiv în care influențezi inamicii prin expresiile faciale

propusă de

Adriana Vlad

Sesiunea: iulie, 2024

Coordonator științific

Lect. Dr. Pătruț Bogdan

UNIVERSITATEA "ALEXANDRU-IOAN CUZA" DIN IASI

FACULTATEA DE INFORMATICA

**MoodLands - platformer interactiv în
care influențezi inamicii prin expresiile
faciale**

Adriana Vlad

Sesiunea: iulie, 2024

Coordonator științific

Lect. Dr. Pătruț Bogdan

Avizat,

Îndrumător lucrare de licență,

Lect. Dr. Pătruț Bogdan.

Data:

Semnătura:

Declarație privind originalitatea conținutului lucrării de licență

Subsemnatul **Vlad Adriana** domiciliat în România, jud. Iași, mun. Iași, alea Rozelor, nr. 12, bloc B8, scara A, ap. 3, născut la data de 30 mai 2002, identificat prin CNP 6020530226728, absolvent al Facultății de informatică, **Facultatea de informatică** specializarea **informatică**, promoția 2024, declar pe propria răspundere cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art. 143 al. 4 și 5 referitoare la plagiat, că lucrarea de licență cu titlul **MoodLands - platformer interactiv în care influențezi inamicii prin expresiile faciale** elaborată sub îndrumarea domnului **Lect. Dr. Pătruț Bogdan**, pe care urmează să o susțin în fața comisiei este originală, îmi aparține și îmi asum conținutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată prin orice modalitate legală pentru confirmarea originalității, consimțind inclusiv la introducerea conținutului ei într-o bază de date în acest scop.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări științifice în vederea facilitării falsificării de către cumpărător a calității de autor al unei lucrări de licență, de diplomă sau de disertație și în acest sens, declar pe proprie răspundere că lucrarea de față nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Data:

Semnătura:

Declarație de consimțământ

Prin prezenta declar că sunt de acord ca lucrarea de licență cu titlul **MoodLands - platformer interactiv în care influențezi inamicii prin expresiile faciale**, codul sursă al programelor și celealte conținuturi (grafice, multimedia, date de test, etc.) care însășesc această lucrare să fie utilizate în cadrul Facultății de informatică.

De asemenea, sunt de acord ca Facultatea de informatică de la Universitatea "Alexandru-Ioan Cuza" din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Absolvent Adriana Vlad

Data:

Semnătura:

Cuprins

| | |
|-------------------------------|-----------|
| Introducere | 2 |
| Contribuții | 4 |
| 1 Descrierea problemei | 5 |
| 2 Abordări anterioare | 9 |
| 3 Descrierea soluției | 12 |
| 3.1 Modelele | 12 |
| 3.2 Legătura | 42 |
| 3.3 Jocul | 45 |
| Concluzii | 62 |
| Bibliografie | 63 |

Introducere

În ultima vreme se poate observa creșterea popularității utilizării tehnologiilor IA și ML în tot mai multe domenii tehnologice și artistice. Odată cu această creștere apar și din ce în ce mai mulți utilizatori sceptici, care din diverse motive percep aceste alternative ca pe o amenințare, ca pe ceva neplăcut sau inutil. Cu toate acestea, aplicațiile de tip IA generative sau de clasificare pot fi folosite pentru a realiza lucruri anterior imposibile și pentru ușurarea unor sarcini repetitive. În practică, există numeroase astfel de aplicații folositoare, dar nu toate sunt user friendly sau doar nu sunt create în scopul de a fi distractive și accesibile tuturor.

Din aceste constatări reiese urmatoarea problemă: mulți oameni au o percepție mai negativă decât ar trebui asupra tehnologiilor mentionate. Putem menționa și o problemă auxiliară, anume cea a lipsei de aplicații de tip entertainment, care ar putea să familiarizeze utilizatorii, cât și dezvoltatorii, cu vastele modalități de a folosi IA uriașă.

Luând acestea în considerare, doresc să aduc o influență pozitivă asupra acestui domeniu prin utilizarea lui într-un mod mai rar întâlnit. În această lucrare am urmărit, deci, dezvoltarea unei aplicații ce poate fi plăcută publicului larg, nu doar folositoare în business sau în scopuri academice, pentru a ameliora problema menționată. Aceasta este motivația din spatele proiectului care, pe scurt, constă în realizarea unui joc interactiv care permite utilizatorului să interacționeze cu inamicii din joc prin diverse vrăji alese pe baza expresiei faciale / emoțiilor arătate la cameră, oferind și alte adaptări pe baza aspectului precum ajustarea aparenței protagonistului pentru a semăna cât mai mult cu jucătorul, cât și ajustarea dificultății.

Antrenarea modelului de recunoaștere facială s-a realizat cu ajutorul bibliotecii tensorflow. Au fost încercate mai multe variante de transfer learning și arhitecturi noi pe bază de Conv2D, în final pentru fiecare categorie alegându-se modelul cu acuratețe maximă la validare. Seturile de date au fost preluate din surse externe și nu realizate manual. Sistemul care realizează predicțiile este scris în Python și realizează comuni-

carea cu jocul prin intermediul unui fișier text. Jocul este la bază de tip platformer și a fost dezvoltat folosind Unity și C#.

Desigur, ideea de a folosi IA în dezvoltarea jocurilor nu este nouă: tactici similare sunt folosite pentru automatizarea unor părți repetitive din procesul de dezvoltare a jocurilor: generarea de teren, a animațiilor faciale pe baza dialogurilor înregistrate, de NPC-uri(Non Player Character) etc.; există, însă, foarte puține jocuri ce utilizează aceste idei în gameplay, adică pentru a interacționa direct cu jucătorul.

Contribuții

Am studiat modalități de antrenare a RN pentru categorizarea imaginilor.

Am căutat numeroase seturi de date pentru gen, vârstă și emoții.

Am realizat mai multe arhitecturi pe bază de Conv2D și pe bază de transfer learning. Le-am testat și comparat pentru a determina cea mai eficientă soluție.

Am făcut un script în python care rulează intermitent predicțiile pe capturi de pe webcamul dispozitivului și le pune într-un fișier pentru a fi transmise mai departe. Scriptul abordează și diversele situații în care o predicție nu poate fi făcută.

Am căutat assets pentru jucător, muzică, efecte sonore, mediu, inamici și fundal.

Am realizat un joc de tip platformer interactiv în Unity cu 3 zone și 5 nivele fiecare.

Am realizat un script C# care interpretează predicțiile din fișier, realizând și un istoric al ultimelor 10 predicții de gen și varsta pentru a evita excepții eronate.

Am îmbunătățit scriptul de controlare a jucătorului cu care a venit assetul: funcția de sărit, abilitatea de resetare a nivelului, mișcarea camerei, alergare și mers. Am adăugat limite pentru numărul de dăți cât poate fi lovit și poate folosi abilitatea.

Am implementat abilitatea specială: oprește temporar toate aspectele jocului în afara de partea de interpretare a camerei și permite selectarea prin click a unui inamic. Pentru dispozitivele fără cameră funcțională am implementat o soluție alternativă.

Am stabilit efectele abilității bazate pe emotia prezisă de model și tipul de inamic.

Am realizat toate scripturile legate de comportamentul inamicilor: schimbarea animațiilor, schimbarea direcției la lovirea peretilor și abilități caracteristice tipului.

Am implementat funcționalitatea de pauză automată a jocului când nu este nici un jucător în fața camerei pentru mai mult de o secundă.

Am testat extensiv manual toate nivelele pentru debugging și level design.

Am implementat sonoritate spațială, paralaxa fundalului, mecanici unice fiecarei zone și fiecărui tip de inamic și un cronometru în ultimul nivel al fiecarei zone.

Am creat un launcher care rulează și jocul și predicțiile, cu toate dependențele.

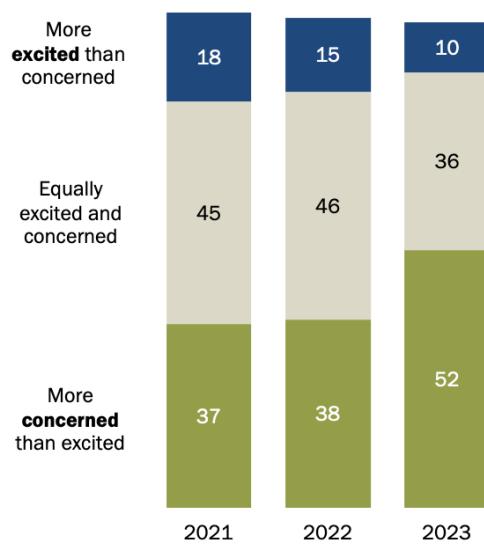
Capitolul 1

Descrierea problemei

După cum am enunțat în introducere, toate lucrurile legate de IA, inclusiv termenul însuși, sunt privite negativ de o parte semnificativă a populației.¹

Concern about artificial intelligence in daily life far outweighs excitement

% of U.S. adults who say the increased use of artificial intelligence in daily life makes them feel ...

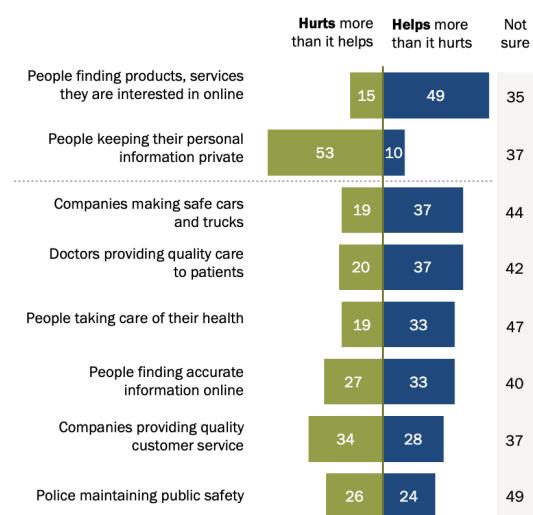


Note: Respondents who did not give an answer are not shown.
Source: Survey conducted July 31-Aug. 6, 2023.

PEW RESEARCH CENTER

Americans have a negative view of AI's impact on privacy, more positive toward impact in other areas

% of U.S. adults who say for each of the following, artificial intelligence ...



Note: Respondents who did not give an answer are not shown.
Source: Survey conducted July 31-Aug. 6, 2023.

PEW RESEARCH CENTER

(a) Păreri pe viitor

(b) Păreri pe domenii

Figura 1.1: Păreri pozitive vs negative

Studiul a fost realizat doar în SUA, nu la scală globală, dar este suficient de expresiv pentru a ne ajuta să înțelegem situația. Putem estima că există mai mult pesimism

¹American views of artificial intelligence

decât optimism când vine vorba de IA, atunci când luam în considerare părerile oamenilor obișnuiți, nu doar ale investitorilor și ale oamenilor care lucrează în domenii adiacente informaticii.

Aceste păreri negative sunt o problemă deoarece cauzează în mod nenecesar disconfort oamenilor. Tehnologiile IA vor continua cu siguranță să fie folosite în viitor, iar sentimentele neplăcute asociate IA îi vor putea determina pe acești oameni să refuse să le folosească și înțeleagă, ducând la o viață mult mai grea decât ar putea de altfel avea.

În urma cercetării acestui subiect, am observat următoarele impresii predominante:

Mulți spun că este din ce în ce mai greu să te întreți financiar în ziua de astăzi, parțial din cauza dificultății crescute de a găsi locuri de muncă. Problema pare să fie înrăutățită de IA, atunci când vedem câte poziții sunt înlocuite de astfel de programe, rezultând în mult mai puțini angajați. Conform unui raport realizat de McKinsey Global Institute, până în 2030, aproximativ 400-800 de milioane de locuri de muncă ar putea fi automatizate la nivel global. Această situație pare similară cu unele din trecut, în care oamenii simțeau că sursa lor de venit era amenințată de automatizarea din diverse industrii. De exemplu, în perioada revoluției industriale, un procent semnificativ din forța de muncă din sectorul agricol a fost nevoie să se reprofileze pe locuri de muncă în industrie și servicii, fenomen care a generat atât anxietate cât și noi oportunități.

Ca și în acel caz, însă, consider că această schimbare poate să aducă noi oportunități financiare și cariere. De exemplu, un raport al World Economic Forum estimează că până în 2025 vor apărea 97 de milioane de noi locuri de muncă în domenii emergente legate de IA, care vor compensa pierderile din sectoarele automatizate.

Cu toate acestea nu putem fi siguri exact ce efecte vor apărea pe termen lung. Tot ce putem face este să creăm noi însine schimbări pozitive în acest sens și să încurajăm alte persoane care încearcă să facă astfel de schimbări. Consider că dezvoltarea unor noi modalități de a interacționa cu calculatorul, asistate de IA de clasificare și grupare, sunt o astfel de schimbare.

Un alt argument este că, în general, crearea de lucruri artistice este ceva plăcut pentru majoritatea. Prin urmare, crearea unei noi tehnologii care să ne înlocuiască în această privință nu aduce nici un beneficiu. Desigur, aici se face referire la un anumit subtip de IA generativ, cum ar fi modelele GPT-3 și DALL-E, dezvoltate de OpenAI, care pot crea texte și imagini care imită stilurile umane. Acestea stârnesc îngrijorări în

rândul scriitorilor și artiștilor. De exemplu, un raport al Authors Guild arată că mulți scriitori sunt preocupați de faptul că IA ar putea reduce cererea pentru munca lor creativă. Organizația a evidențiat necesitatea unor reglementări clare și a unei protecții mai mari pentru drepturile de autor, argumentând că munca creativă a scriitorilor nu ar trebui utilizată pentru antrenarea modelelor IA generative fără consimțământul lor. În 2023, conform unui studiu realizat de National Endowment for the Arts, aproximativ 12% din scriitori și artiști vizuali au raportat că și-au pierdut locurile de muncă sau au sesizat o reducere semnificativă a veniturilor din cauza automatizării și utilizării pe scară largă a acestor IA uri.

Acest argument subliniază necesitatea de creare a IA care să asiste oamenii în crearea lucrurilor pe care le doresc, sau în crearea de lucruri noi, în favoarea celor care pot face în locul oamenilor ceea ce ei ar face cu plăcere.

Un alt aspect atins mai sus este faptul că oamenii nu vor ca tot ce postează pe internet să poată fi folosit pentru antrenarea de IA. Mulți doresc legi de protecție împotriva lor, în mare parte din motive legate de confidențialitate.

Astfel s-a ajuns la dezvoltarea unor legi noi. În Uniunea Europeană, de exemplu, a fost adoptat Regulamentul privind Inteligența Artificială. Aceasta are ca scop crearea unui cadru juridic armonizat pentru funcționarea și utilizarea IA, spre exemplu prin impunerea unui nivel de transparentă în dezvoltarea IA.

Popularea rețelelor sociale cu informații învățate doar din IA sau imagini generate poate duce la răspândirea mult mai intensă ca până acum de informații false. De exemplu, în domeniul sănătății, expunerea la conținut generat incorect de IA poate avea drept efect informarea eronată a persoanelor care poate au nevoie urgentă de sfaturi medicale.

Ca reacție la aceste posibilități, însă, alte sisteme asistate de IA sunt dezvoltate pentru a detecta și modera dezinformarea online, la o scară mult mai mare decât ar fi fezabil manual.

Scopul proiectului nu este de a aborda această latură a problemei, dar cert este că unele frici pe care le au oamenii legate de IA au origini justificate, dar există și multe măsuri împotriva lor în curs de dezvoltare.

Devine aparentă o tendință de generalizare, justificată adesea de faptul că exemplele negative devin mult mai populare pe rețelele sociale decât cele pozitive. Este evident că în era digitală, perceptia publică asupra IA este influențată în mare măsură de exemplele care circulă pe rețelele sociale. Acestea pot crea o imagine distorsionată

a IA, în care beneficiile sale sunt adesea trecute cu vederea. Devine deosebit de importantă existența și a suficient de multe exemple pozitive pentru a combată astfel de influențe.

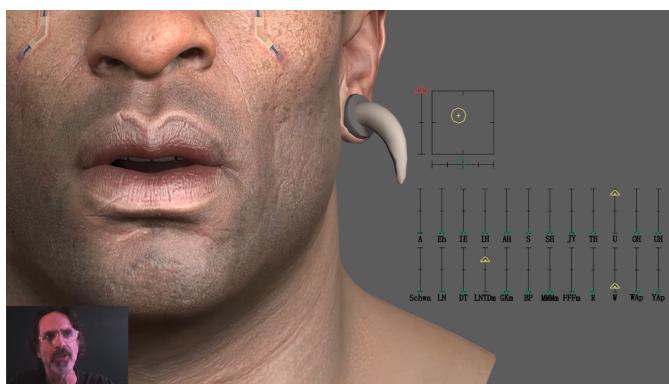
Într-o notă similară, este important să ne amintim că percepția publică asupra IA nu a fost întotdeauna negativă. În trecut, oamenii priveau cu optimism IA, considerând această idee ca fiind un instrument puternic pentru rezolvarea problemelor complexe și îmbunătățirea vietii, sau poate doar o nouă tehnologie interesantă și distractivă. Cu toate acestea, în ultimii ani, preocupările legate de controlul asupra IA, problemele etice și impactul negativ asupra muncii au crescut.

Luând toate aceste date în considerare, nu ar fi greșit să considerăm că o posibilă soluție ar fi popularizarea și normalizarea folosirii diverselor tipuri de IA pentru a asista procesul creativ uman. Cu alte cuvinte, consider că IA urile ar trebui mai des folosite în realizarea de creații inovative, anterior imposibile, și că aceste abordări ar trebui mai public anunțate, răspândite și popularizate.

Capitolul 2

Abordări anterioare

În primul rând, după cum am menționat anterior, utilizarea IA urilor pentru ușurarea procesului de dezvoltare a jocurilor nu este nimic nou în industrie. Printre cele mai faimoase exemple se numără jocuri precum Cyberpunk 2077¹ care au folosit aceste tehnologii pentru generarea automată de mișcări ale gurii pe baza liniilor de dialog înregistrate în mai multe limbi, lăsând mai mult timp angajaților să adauge detalii. Jocurile nu sunt, totuși, singurul tip de entertainment care poate fi asistat de IA în producție. Pentru filmul "Spider-Man: Across the Spider-Verse"², echipa a folosit IA generativ, antrenat pe munca anterioară a angajaților, pentru a-i ajuta să adauge mai rapid detalii modelelor 3D folosite în animație, cum ar fi liniile negre stilistice adăugate tuturor personajelor.



(a) Cyberpunk 2077 Lipsync



(b) Spider-Man: Across the Spider-Verse Automation

¹JALI Driven Expressive Facial Animation & Multilingual Speech in CYBERPUNK 2077 with CDPR

²Making Of SPIDER-MAN: ACROSS THE SPIDER-VERSE - Best Of Behind The Scenes & Creating

The Animations

Din punctul de vedere al prezentei lucrări, cele mai bune exemple de analizat sunt cele care folosesc diverse tipuri de IA ca parte din gameplay, nu doar în crearea de obiecte. Un astfel de joc este Suck Up!³. În acest dialogul tuturor personajelor este generat de IA, pentru a putea reacționa la cuvintele jucătorului. Mai exact, aspectul central al acestui joc este abilitatea jucătorului de a vorbi direct cu personajele, fapt asistat de un IA de recunoaștere a vorbirii. Jocul adaptează expresiile faciale și unele dintre acțiunile personajelor în funcție de ce spune jucătorul. În ceea ce privește premissa, jucătorul este pus în poziția unui vampir, care trebuie să își convingă vecinii să îl lase la ei în casă. Alte jocuri, precum Quick, Draw⁴, se bazează pe încercările unui IA de a categoriza și înțelege acțiunile jucătorului. Jucătorul are sarcina de a desena diverse obiecte cât mai repede, într-un fel în care IA-ul poate ghici cu succes ce încercă să deseneze.



Figura 2.2: Suck Up!

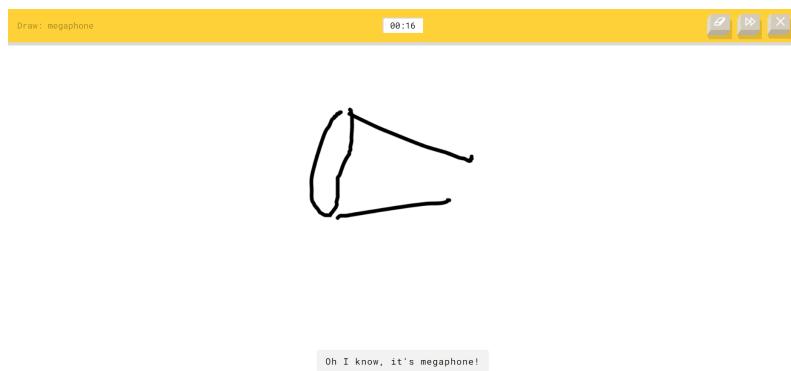


Figura 2.3: Quick, Draw

Există multe jocuri de tip text adventure, unde un IA generativ joacă rolul unui

³Suck Up! - main website

⁴Quick, Draw - main website

game master dintr-un joc de rol și reacționează la solicitările jucătorului; un exemplu notabil pentru astfel de jocuri este AI Dungeon⁵.

De asemenea, merită menționate jocurile mici publicate de Google în încercarea de a testa limitele IA urilor dezvoltate, cum ar fi Odd One Out — Google Arts & Culture⁶. În acesta jucătorul trebuie să ghicească ce imagine este generată de IA și care e reală. Google are multe astfel de proiecte, multe dintre ele găsindu-se pe platforma Google Research⁷.

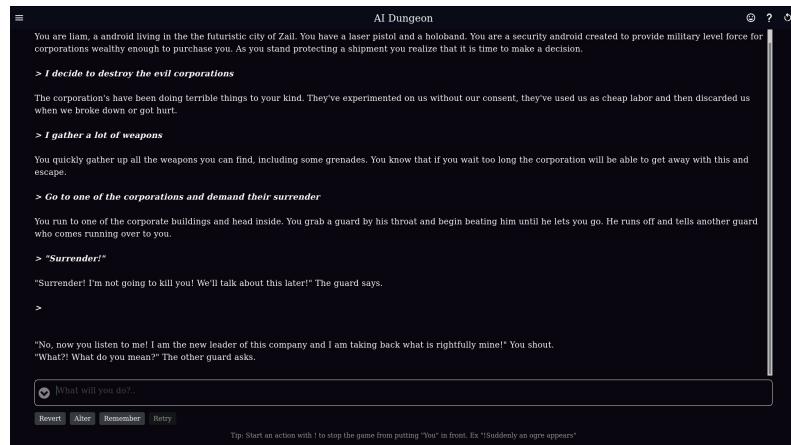


Figura 2.4: AI Dungeon



Figura 2.5: Odd One Out

După cum putem observa din exemplele enunțate, există multe companii care încearcă să extindă limitele acestui domeniu, să creeze prototipuri și să încurajeze utilizarea IA în moduri distractive și creative. Totuși domeniul este relativ nou și, deci, mai are mult până își poate atinge potențialul adevarat. Cert este că IA poate fi distractiv și pentru utilizator, fiind capabil de mult mai multe decât folosirea doar în producție.

⁵AI Dungeon - main website

⁶Odd One Out - main website

⁷Google Research - main website

Capitolul 3

Descrierea soluției

Totul a început din dorința de a învăța despre rețele neuronale pentru a le utiliza în alt proiect. Aspectul care mi s-a părut cel mai interesant a fost analiza imaginilor pentru detectarea de trăsături și categorizare, aşa că am realizat modele pentru gen și varsta pe care doream să le folosesc pentru o aplicație web. Abia după ce am realizat deja aceste modele am început să mă gândesc ce alte aplicații le-ar putea folosi și care dintre ele mi-ar face cea mai mare placere să o dezvolt. Fiind pasionată de jocuri și având deja experiență cu Unity, am ales să construiesc un joc care folosește modelele pe care le aveam deja. Între timp am realizat că a fost o alegere cu adevărat inspirată, tocmai din cauza problemei anterior discutate, despre care am devenit destul de pasionată.

Ca aspecte generale de implementare, antrenările se bazează pe biblioteca tensorflow pentru Windows. Deoarece ultimele versiuni de tensorflow, care folosesc versiuni mai recente de python, sunt exclusiv pentru Linux, programele pe care le-am dezvoltat eu cu tensorflow au fost scrise în python 3.9.17. Ca mediu de lucru am utilizat VSCode, care dispune opțiunea "Run in interactive window" prin extensia Jupyter, opțiune folosită extensiv în realizarea acestui proiect.

3.1 Modelele

Primul pas în antrenarea rețelelor neuronale este găsirea unui set de date. Primul set de date¹, pe care l-am utilizat la antrenarea de modele pentru detectarea genului și vîrstei, conține 26580 imagini color, de dimensiune 224x224px fiecare, cu imagini

¹Gender and age dataset

în diverse situații și locații de la 2284 de indivizi. Imaginele sunt etichetate. Cu toate acestea, grupele de vârstă din etichete au unele goluri și suprapuneri între ele. Prin urmare am desic să le ajustez pentru a acoperi toate vârstele fără suprapuneri, în final cele 8 grupe de vârstă fiind: 0-3, 4-7, 8-13, 14-20, 21-32, 33-43, 44-59, 60+. Setul de date conține, pe lângă imagini, 5 fișiere text cu următoarele informații despre 19370 din imagini:

- user_id numele folderului în care se gaseste.
- original_image numele imaginii / fișierului.
- face_id Face ID, conform README poate fi ignorat.
- age grupa de vârstă. Formatul cel mai comun este “(age1, age2)”, dar există un număr substanțial de exceptii. La citire au fost luate în considerare toate valorile care s-ar putea încadra într-o anumită grupă.
- gender genul, “f” sau “m”. 1099 imagini aveau valoarea “u”, de la unsure, și nu au fost utilizate în antrenare
- x, y, dx, dy chenarul de încadreare a feței în imaginea originală.
- tilt_ang, fiducial_yaw_angle poziția feței în imaginea originală.
- fiducial_score scor asociat imaginii de la un program folosit în prelucrare, conform README poate fi ignorat.

Setul de date pune la dispoziție 2 versiuni pentru fiecare imagine, originalul și una rotită cu un program al creatorilor setului de date, folosit pentru alinierea feței. Pentru modelele mele am folosit doar prima versiune.

În programul de antrenare, primul pas a fost citirea tuturor fișierelor text. Acestea erau în format csv, aşa că pentru citirea și memorarea eficientă a lor am folosit biblioteca pandas. Drept consecință al folosirii acestei biblioteci, datele au fost stocate în variabile de tip DataFrame, care funcționează asemenei unor maps în care cheia este stringul numelui unei coloane din fișier, iar valoarea asociată unei chei este lista tuturor valorilor din acea coloană. Am memorarat căile complete către fiecare imagine pe baza informațiilor din acele fișiere și am mutat toate informațiile din fișier pe care doream să le folosesc în continuare, anume doar etichetele de gen și vârstă și calea completă către imagine, în alt DataFrame. Am înlocuit etichetele inițiale de vârstă cu

varianta ajustată de mine și am elimitat toate imaginile care aveau etichete ambigue, cum ar fi "u" în cazul etichetelor de gen, sau goale, prin funcția dropna din pandas. În final au ramas 17452 imagini. Înlocuirea etichetelor de vîrstă cu cele uniformizate a fost realizată pe baza următoarei liste, care indică tuple (val1, val2), unde val1 reprezintă eticheta inițială și val2 cea din urma transformării:

```
[('(0, 2)', '0 – 3'), ('2', '0 – 3'), ('3', '0 – 3'), ('(4, 6)', '4 – 7'), ('(8, 12)', '8 – 13'), ('13', '8 – 13'), ('22', '21 – 32'), ('(8, 23)', '14 – 20'), ('23', '21 – 32'), ('(15, 20)', '14 – 20'), ('(25, 32)', '21 – 32'), ('(27, 32)', '21 – 32'), ('32', '21 – 32'), ('34', '33 – 43'), ('29', '21 – 32'), ('(38, 42)', '33 – 43'), ('35', '33 – 43'), ('36', '33 – 43'), ('42', '33 – 43'), ('45', '44 – 59'), ('(38, 43)', '33 – 43'), ('(38, 42)', '33 – 43'), ('(38, 48)', '33 – 43'), ('46', '44 – 59'), ('(48, 53)', '44 – 59'), ('55', '44 – 59'), ('56', '44 – 59'), ('(60, 100)', '60+'), ('57', '44 – 59'), ('58', '44 – 59')]
```

Valorile exacte din val1 au fost determinate prin afișarea tuturor valorilor distincte din lista de etichete de vîrstă. După această ajustare am declarat maps pentru cele 2 etichete, dorind să le asociiez valori numerice pentru a putea aplica funcția de pierdere

sparse_categorical_crossentropy în antrenare. Etichetele finale sunt: pentru gen, f=0, m=1; iar la vîrstă sunt cele 8 valori, în ordine precizată, de la 0 la 7.

După ce toate datele necesare au fost astfel reorganizate, a urmat pregătirea datelor pentru antrenare. Primul model realizat a fost cel pentru gen. Am preluat toate căile către imagini într-o listă x și toate etichetele pentru gen în alta listă y, după care am aplicat funcția *train_test_split*, din biblioteca *sklearn.model_selection*. Parametrii exacti folosiți au fost *test_size* = 0.15 și un *random_state* fix, anume 42, pentru a asigura faptul că fiecare încercare va produce aceleași rezultate, dar și că voi avea acces la același set pentru testare mai târziu. Am utilizat și parametrul *stratify* = y pentru a asigura că raportul dintre numărul de date pentru fiecare etichetă este același și în setul de antrenare și în cel de testare. Split-ul este mai mic decât în mod obișnuit, valoarea cea mai comună fiind 0.3. Am considerat că modelul ar putea fi îmbunătățit de această schimbare, după câteva încercări cu split mai mare care au rezultat în acuratețe la validare mai mică, prin comparație cu cea la antrenare, cât și cea a versiunilor cu split mai mic. Prin această modificare, modelele s-au antrenat pe un număr mai mare de imagini, modificare facilitată și de dimensiunea setului de date, fiind suficient de mare pentru a conferi validitate și testelor realizate doar pe 15% din numărul total de imagini. În final au fost selectate 14834 imagini pentru antrenare și 2618 pentru vali-

dare.

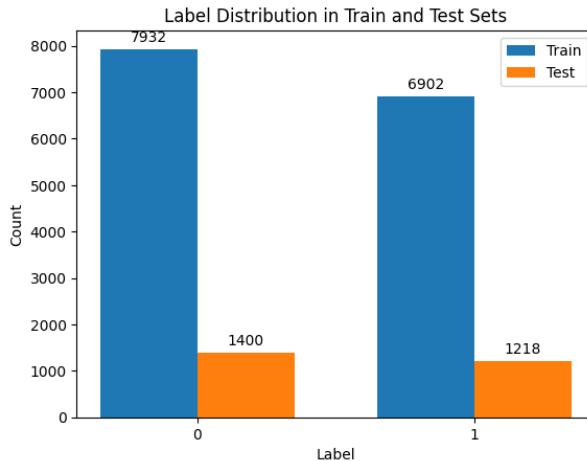


Figura 3.1: Gender Split

Dupa împărțirea setului de date în seturi de antrenament și test, imaginile au fost încărcate în memorie cu Image din biblioteca PIL, redimensionate la dimensiunea așteptată pentru siguranță și transformate în numpy arrays. Am creat o listă de imagini de antrenare și una de imagini de testare, liste care la rândul lor au fost transformate în numpy arrays.

După realizarea programului de prelucrare a capturilor de pe camera web am decis, însă, să mă reîntorc la acest pas. Am adăugat aici încă o etapă de prelucrare a imaginilor, pentru a simula modul de prelucrare a capturilor de ecran în timpul jocului și a asigura specializarea acestor modele pentru aplicația finală. În funcția *sim_unity_comm* utilizez *face_locations* din biblioteca *face_recognition* pentru a detecta față din fiecare imagine și pentru a lărgi zona detectată cu 5% din înălțimea imaginii. Motivația și alte detalii se găsesc în următorul subcapitol, care discută programul de preluare și prelucrare a capturilor de la cameră.

Întrucât dispozitivul pe care a fost realizată antrenarea beneficiază de un GPU cu doar 3GB VRAM, am adăugat aici un pas optional de optimizare: am convertit anterior antrenarii lista cu imaginile și cea cu etichetele de antrenare în tensori, care ocupă mult mai puțin spațiu. Acest lucru este posibil prin funcția *convert_to_tensor* din tensorflow. O altă optimizare pentru minimizarea spațiului ocupat pe GPU a fost stergerea variabilelor devenite inutile înainte de antrenare, prin apelarea manuală a *del*. În unele cazuri a fost necesară și apelarea manuală a garbage collector, cum ar fi după antrenarea unui model, pentru a putea antrena un al doilea fără resetarea kernel ului.

Deși în procesul de învățare și acomodare cu tensorflow am încercat mai multe arhitecturi, în final voi compara cea mai reușită variantă din cele 2 tipuri de abordări pe care le-am încercat. Prima abordare se referă la realizarea unei arhitecturi secvențiale de la 0. Sumarul modelului la care am ajuns este următorul:

| Layer (type) | Output Shape | Param # |
|--|----------------------|------------------|
| conv2d (Conv2D) | (None, 112, 112, 32) | 2432 |
| batch_normalization (BatchNormalization) | (None, 112, 112, 32) | 128 |
| max_pooling2d (MaxPooling2D) | (None, 56, 56, 32) | 0 |
| conv2d_1 (Conv2D) | (None, 56, 56, 64) | 51264 |
| batch_normalization_1 (BatchNormalization) | (None, 56, 56, 64) | 256 |
| max_pooling2d_1 (MaxPooling2D) | (None, 28, 28, 64) | 0 |
| conv2d_2 (Conv2D) | (None, 28, 28, 128) | 73856 |
| batch_normalization_2 (BatchNormalization) | (None, 28, 28, 128) | 512 |
| max_pooling2d_2 (MaxPooling2D) | (None, 14, 14, 128) | 0 |
| conv2d_3 (Conv2D) | (None, 14, 14, 256) | 295168 |
| batch_normalization_3 (BatchNormalization) | (None, 14, 14, 256) | 1024 |
| max_pooling2d_3 (MaxPooling2D) | (None, 7, 7, 256) | 0 |
| flatten (Flatten) | (None, 12544) | 0 |
| dense (Dense) | (None, 256) | 3211520 |
| dropout (Dropout) | (None, 256) | 0 |
| dense_1 (Dense) | (None, 256) | 65792 |
| dropout_1 (Dropout) | (None, 256) | 0 |
| dense_2 (Dense) | (None, 2) | 514 |
| Total params: | | 3,702,466 |
| Trainable params: | | 3,701,506 |
| Non-trainable params: | | 960 |

Tabela 3.1: Model1 Summary

După cum se poate observa, acesta este bazat pe layere de tip Conv2D, Batch-Normalization și MaxPooling2D. Conv2D este utilizat deoarece este conceput pentru învățarea pe bază de imagini. Numărul de filtre în cazul primului strat Conv2D este de 32, număr care se dublează la fiecare strat următor de acest tip. De asemenea, am considerat că strides, numărul de pași, la primul strat poate fi 2, dar pentru acuratețe ridicată a fost setat la 1 în celelalte straturi. Mărimea kernelului este de (5, 5) în primele 2 straturi, unde datele de intrare sunt de dimensiuni mai mari, și de (3,3) la celelalte două, mărimea fiind redusă pentru a putea rula eficient pe date de intrare de dimensiuni mai mici. Pentru pierdere minimă de informații am utilizat *padding = same*. Ca funcție de activare am utilizat *relu*, alegere populară datorită faptului că este cea mai eficientă modalitate de a introduce non linearitate în procesul de antrenare, aspect deosebit de important pentru învățarea de lucruri complexe de către rețea, dar și pentru că

ajută în combaterea problemei gradientului care dispare. Încă o măsura de combatere a problemelor ce pot apărea, cum ar fi destabilizarea valorilor din datele de ieșire ale straturilor COnv2D, am decis să utilizez BatchNormalization, funcție de normalizare potrivită tipului de antrenare pe care intentionam să o realizez, anume în batches mici. Înainte de introducerea unui strat Conv2D am utilizat un strat MaxPooling2D pentru a reduce dimensiunea datelor, util în simplificarea treptată a problemei pentru identificarea și păstrarea doar a celor mai importante aspecte învățate. După analiza imaginilor se găsește un strat Flatten, menit să reconfigureze datele într-o formă acceptată de straturile Dense, necesare predicției finale. Pentru a combate overfitting ul am introdus o serie de straturi Dense cu straturi Dropout între ele înainte de stratul final. Dropout are rata setată la 0.25, indicând faptul că renunță aleator, la fiecare iteratie, la un sfert din valorile din neuroni. Neuronii aleși fiind alții la fiecare iteratie, operația asigură tuturor neuronilor șansa de a învăța corespunzător.

În etapa de compilare am ales optimizerul standard pentru tensorflow, Adam, cu o rată de învățare de 0.0001. Funcția de pierdere este cea dedicată task urilor de categorizare cu etichete de la 0 la n-1, unde n e numărul de etichete:

sparse_categorical_crossentropy. Metrica urmărită este acuratețea, preferată în favoarea minimizării funcției de pierdere, care ar fi asigurat un nivel mai mare de certitudine a rețelei în realizarea predicțiilor. În schimb consider că un număr cât mai ridicat de predicții corecte este de preferat.

În etapa de fitting am decis utilizarea a *batch_size* = 32, un număr suficient de mare dar care nu cauza probleme de tipul Out Of Memory, și 25 de epoci, întrucât am sesizat că modelul nu mai evoluă în epoci ulterioare. M-am utilizat și de funcționalitatea de callbacks din tensorflow, mai exact de *ReduceLROnPlateau* și *ModelCheckpoint*. Prima dintre ele reduce rata de antrenare după un număr de epoci în care metrica urmărită nu evoluează, fiind în acest caz setată la 5 epoci. A doua asigură salvarea celui mai bun model din cadrul antrenării, chiar și în situația în care acesta nu este ultimul. Mai exact a fost setat să salveze modelul din starea / epoca în care acuratețea la validare era maximă, întrucât consider că aceasta este metrica cea mai importantă, indicând abilitatea modelului de a estima pe imagini diferite de cele pe care s-a antrenat.

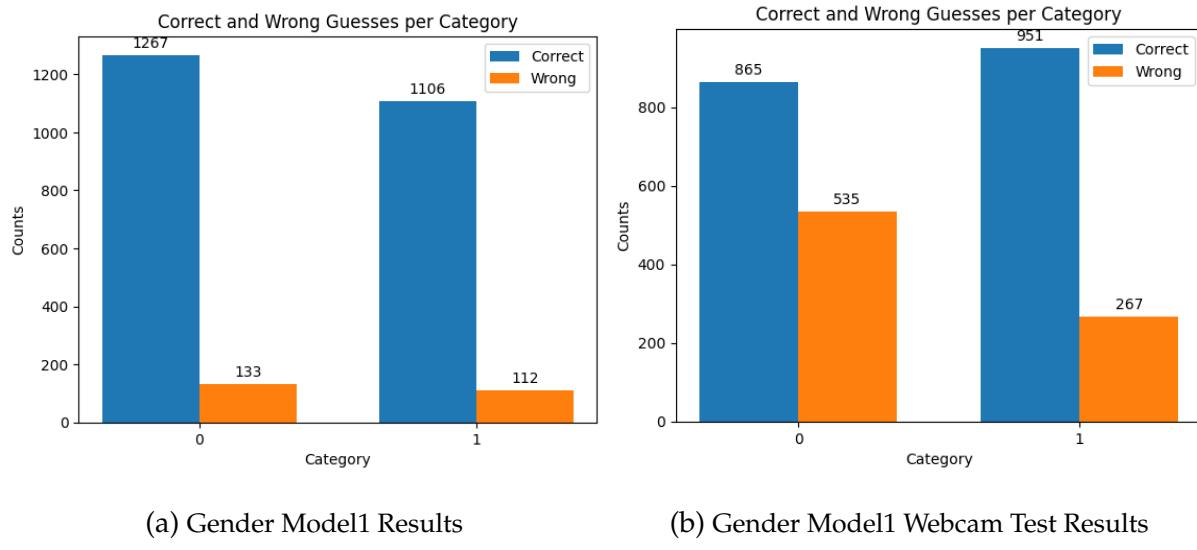
Prima antrenare, cea dinaintea prelucrării pentru camera web, a decurs după cum urmează:

| Epoch | Loss | Accuracy | Val_Loss | Val_Accuracy | Time_Per_Epoch | Time_Per_Step |
|-------|--------|----------|----------|--------------|----------------|---------------|
| 1 | 0.6704 | 0.6513 | 0.5360 | 0.7200 | 140s | 303ms/step |
| 2 | 0.5171 | 0.7399 | 0.4821 | 0.7651 | 20s | 44ms/step |
| 3 | 0.4238 | 0.8042 | 0.4100 | 0.8067 | 21s | 45ms/step |
| 4 | 0.3388 | 0.8483 | 0.3824 | 0.8212 | 21s | 45ms/step |
| 5 | 0.2698 | 0.8835 | 0.3426 | 0.8568 | 21s | 45ms/step |
| 6 | 0.2108 | 0.9104 | 0.3256 | 0.8675 | 21s | 44ms/step |
| 7 | 0.1601 | 0.9386 | 0.3471 | 0.8625 | 19s | 42ms/step |
| 8 | 0.1305 | 0.9489 | 0.3294 | 0.8682 | 21s | 44ms/step |
| 9 | 0.1006 | 0.9595 | 0.3451 | 0.8766 | 20s | 44ms/step |
| 10 | 0.0828 | 0.9670 | 0.3220 | 0.8915 | 21s | 45ms/step |
| 21 | 0.0060 | 0.9983 | 0.4079 | 0.9057 | 19s | 41ms/step |
| 22 | 0.0047 | 0.9987 | 0.4107 | 0.9057 | 19s | 41ms/step |
| 23 | 0.0043 | 0.9988 | 0.4135 | 0.9053 | 20s | 42ms/step |
| 24 | 0.0053 | 0.9987 | 0.4157 | 0.9064 | 23s | 49ms/step |
| 25 | 0.0049 | 0.9987 | 0.4174 | 0.9045 | 20s | 42ms/step |

Tabela 3.2: Gender Model1 Train Summary

Antrenarea a durat 10 minute. Acuratețea la validare este de 90.64%, fiind relativ identică pentru ambele etichete analizate individual.

Am decis să testezi aceste versiuni inițiale și pe seturile de validare prelucrate de simulatorul de webcam. Pentru acest model rezultatele indică inabilitatea modelului de a generaliza. Acuratețea este de 69.36%:



A urmat antrenarea în care toate imaginile au fost trecute prin funcția de simulare:

| Epoch | Loss | Accuracy | Val_Loss | Val_Accuracy | Time_Per_Epoch | Time_Per_Step |
|-------|--------|----------|----------|--------------|----------------|---------------|
| 1 | 0.6977 | 0.6344 | 0.5741 | 0.7086 | 52s | 113ms/step |
| 2 | 0.5501 | 0.7240 | 0.5227 | 0.7334 | 22s | 48ms/step |
| 3 | 0.4877 | 0.7740 | 0.4988 | 0.7777 | 22s | 48ms/step |
| 4 | 0.4408 | 0.8019 | 0.4637 | 0.7922 | 23s | 48ms/step |
| 5 | 0.3957 | 0.8285 | 0.4344 | 0.8125 | 22s | 48ms/step |
| 6 | 0.3514 | 0.8511 | 0.4081 | 0.8254 | 22s | 48ms/step |
| 7 | 0.3152 | 0.8714 | 0.4208 | 0.8251 | 22s | 46ms/step |
| 8 | 0.2857 | 0.8839 | 0.4450 | 0.8151 | 21s | 46ms/step |
| 9 | 0.2584 | 0.8994 | 0.4509 | 0.8422 | 22s | 48ms/step |
| 10 | 0.2314 | 0.9129 | 0.4534 | 0.8239 | 22s | 46ms/step |
| 21 | 0.0971 | 0.9601 | 0.4648 | 0.8701 | 22s | 46ms/step |
| 22 | 0.0976 | 0.9598 | 0.4654 | 0.8697 | 21s | 46ms/step |
| 23 | 0.0988 | 0.9580 | 0.4655 | 0.8705 | 22s | 46ms/step |
| 24 | 0.0984 | 0.9575 | 0.4652 | 0.8697 | 21s | 46ms/step |
| 25 | 0.0981 | 0.9584 | 0.4653 | 0.8690 | 21s | 46ms/step |

Tabela 3.3: Gender Model1 Webcam Train Summary

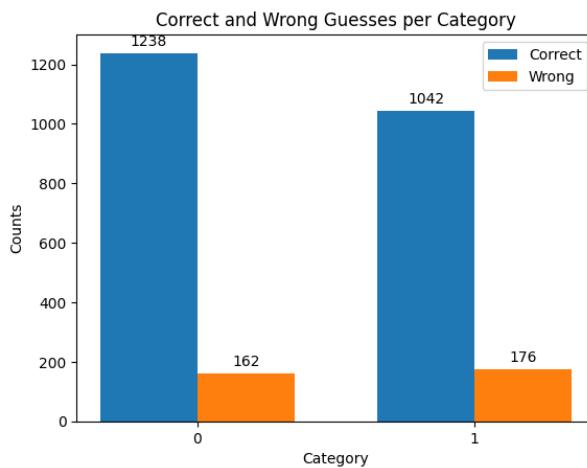


Figura 3.3: Gender Model1 Webcam Train Results

Acuratețea este de 87.09%, puțin mai mică decât cea la validare a primei versiuni pe imagini neprelucrate, dar considerabil mai bună decât pe cele prelucrate.

O problemă considerabilă a primului model a fost overfitting-ul, aşa că am căutat modele preexistente care să gestioneze bine această problemă, cu intenția de a le folosi într-o abordare de tip transfer learning. ResNet50 a dat rezultate comparabile și cu alte modele încercate, aşa că am decis să îl folosesc de-a lungul întregului proiect, ca al doilea tip de arhitectură.

Un aspect până acum nediscutat este augmentarea datelor. Am încercat abordări de această natură, dar din păcate imaginile ocupau prea multă memorie, rezultând în eroarea Out Of Memory. Acest lucru se datorează și dimensiunii imaginilor, 224x224px

fiind destul de mult pentru un set cu mii de imagini. Am luat în considerare antrenare în 2 etape, o dată pe setul inițial de antrenare și o dată doar pe imagini augmentate, variantă care a adus rezultate extrem de asemănătoare, motiv pentru care nu a fost utilizată. Augmentarea respectivă a fost realizată cu ajutorul ImageDataGenerator din tensorflow. Specificațiile au fost *zoom_range* de 0.1, *channel_shift_range* de 10, *rotation_range* de 0.2 și *horizontal_flip*. Tot din cauza erorii OOM nu am putut genera decât o singură imagine augmentată per imagine originală. În orice caz, un motiv pentru care augmentarea nu a avut un efect semnificativ ar putea fi faptul că setul de date conține deja multe imagini asemănătoare, care par să fie mici variații din aceeași situație și moment, similar rezultatelor unei augmentări.

Revenind la ResNet50, arhitectura este următoarea:

| Layer (type) | Output Shape | Param # |
|--|--------------|-------------------|
| resnet50 (Functional) | (None, 2048) | 23,587,712 |
| layer_normalization_1 (LayerNormalization) | (None, 2048) | 4096 |
| dense_3 (Dense) | (None, 256) | 524,544 |
| dropout_2 (Dropout) | (None, 256) | 0 |
| dense_4 (Dense) | (None, 256) | 65,792 |
| dropout_3 (Dropout) | (None, 256) | 0 |
| dense_5 (Dense) | (None, 2) | 514 |
| Total params: | | 24,182,658 |
| Trainable params: | | 24,129,538 |
| Non-trainable params: | | 53,120 |

Tabela 3.4: Model2 Summary

Am ales să nu includ ultimele straturi Dense din acesta deoarece doream o abilitate mai mare de customizare a modelului, pentru a putea testa cât mai multe variante, aşa că am ales *include_top = False*. Pentru *pooling* am utilizat *avg*, pentru a reduce dimensiunea datelor cu pierdere minimă de informații. Am decis și preluarea ponderilor din setul de date pe care a fost inițial antrenat ResNet50, ImageNet, rezultând într-o acuratețe inițială mai ridicată decât obisnuitul 50/50. LayerNormalization realizează normalizarea outputului inițial al ResNet50, dar pentru fiecare imagine în mod individual, asigurând că trăsăturile exceptionale extrase nu au un efect excesiv de puternic asupra rezultatului final. Straturile Dense și Dropout funcționează în manieră similară celor din arhitectura anterioară și au același scop.

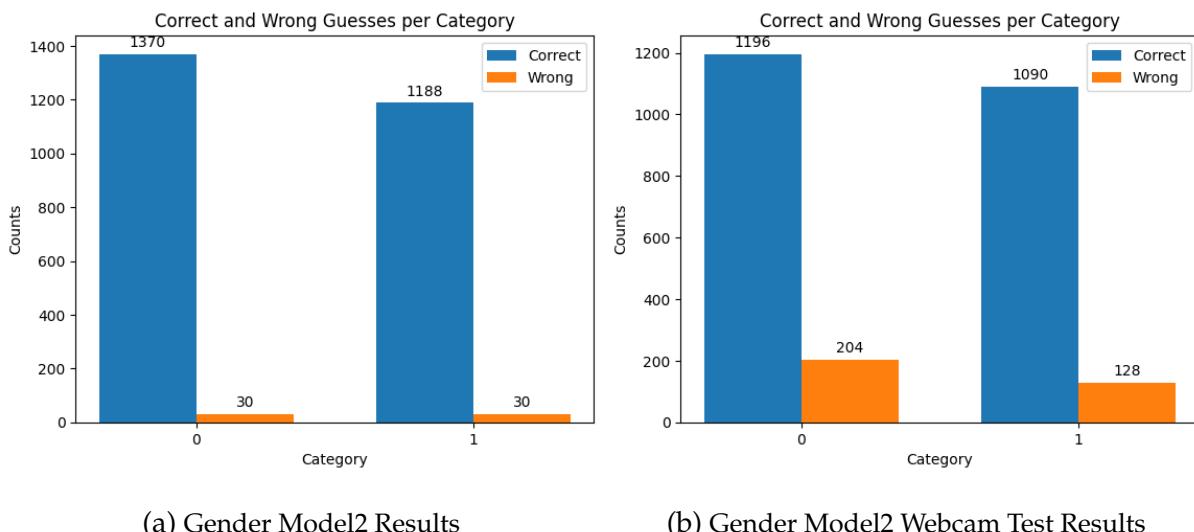
Specificațiile de antrenare sunt în mare neschimbate. Singura diferență este că, datorită mărimii considerabil mai mari a acestei arhitecturi din cauza utilizării ResNet50, necesarul de memorie este mult mai ridicat. Pentru a putea antrena rețeaua a

fost necesară reducerea semnificativă a *batch_size*, de la 32 în versiunea anterioară, la 8. Acest aspect cuplat cu dimensiunea modelului au crescut drastic timpul de antrenare la aproximativ 2 ore și 10 minute. Antrenarea a mers în felul următor:

| Epoch | Loss | Accuracy | Val_Loss | Val_Accuracy | Time_Per_Epoch | Time_Per_Step |
|-------|--------|----------|----------|--------------|----------------|---------------|
| 1 | 0.3479 | 0.8529 | 0.2088 | 0.9148 | 330s | 178ms/step |
| 2 | 0.1715 | 0.9315 | 0.1693 | 0.9305 | 333s | 179ms/step |
| 3 | 0.1230 | 0.9529 | 0.3777 | 0.8720 | 319s | 172ms/step |
| 4 | 0.0987 | 0.9647 | 0.1444 | 0.9458 | 330s | 178ms/step |
| 5 | 0.0724 | 0.9738 | 0.0987 | 0.9660 | 329s | 177ms/step |
| 6 | 0.0558 | 0.9802 | 0.1316 | 0.9557 | 318s | 171ms/step |
| 7 | 0.0484 | 0.9836 | 0.2297 | 0.9293 | 318s | 171ms/step |
| 8 | 0.0456 | 0.9840 | 0.1376 | 0.9561 | 318s | 171ms/step |
| 9 | 0.0403 | 0.9856 | 0.1353 | 0.9515 | 318s | 171ms/step |
| 10 | 0.0339 | 0.9879 | 0.1377 | 0.9584 | 318s | 171ms/step |
| 21 | 0.0002 | 0.9998 | 0.1422 | 0.9767 | 319s | 172ms/step |
| 22 | 0.0003 | 0.9998 | 0.1426 | 0.9759 | 319s | 172ms/step |
| 23 | 0.0002 | 0.9999 | 0.1421 | 0.9763 | 318s | 172ms/step |
| 24 | 0.0003 | 0.9999 | 0.1395 | 0.9771 | 328s | 177ms/step |
| 25 | 0.0003 | 0.9998 | 0.1408 | 0.9767 | 317s | 171ms/step |

Tabela 3.5: Gender Model2 Train Summary

Se poate observa că modelul a învățat aproape perfect setul de antrenare, dar obține rezultate extram de bune și în validare. Rezultatele testării indică o acuratețe de 97.71%. Testarea pe imaginile trecute prin simulator a adus rezultate relativ bune, de 87.31%.



(a) Gender Model2 Results

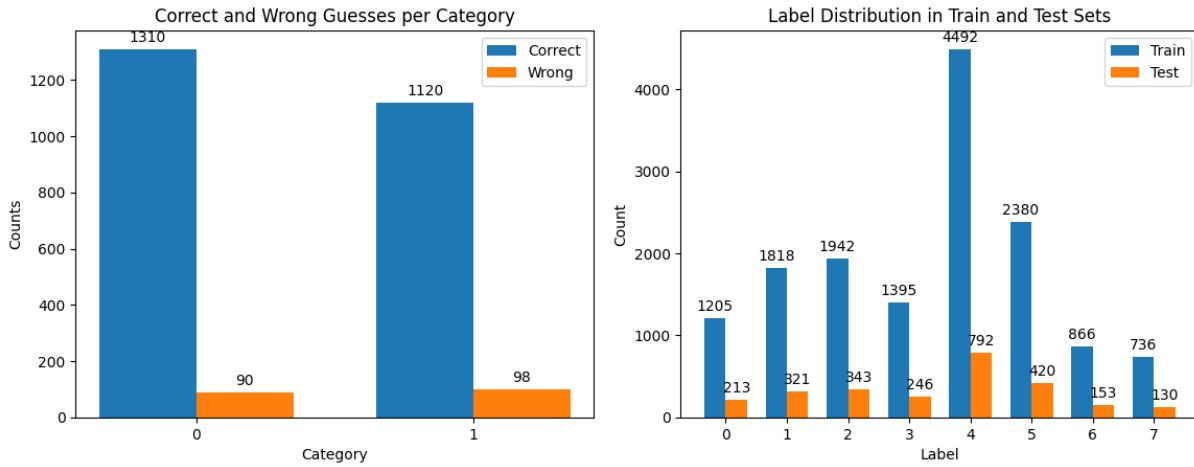
(b) Gender Model2 Webcam Test Results

A urmat și de această dată reantrenarea pe imaginile deja prelucrate:

| Epoch | Loss | Accuracy | Val_Loss | Val_Accuracy | Time_Per_Epoch | Time_Per_Step |
|-------|--------|----------|----------|--------------|----------------|---------------|
| 1 | 0.5155 | 0.7671 | 0.4536 | 0.7953 | 342s | 184ms/step |
| 2 | 0.3740 | 0.8536 | 0.4102 | 0.8266 | 333s | 180ms/step |
| 3 | 0.3080 | 0.8874 | 0.3724 | 0.8438 | 332s | 179ms/step |
| 4 | 0.2746 | 0.9035 | 0.3394 | 0.8678 | 333s | 180ms/step |
| 5 | 0.2462 | 0.9208 | 0.3478 | 0.8625 | 323s | 174ms/step |
| 6 | 0.2290 | 0.9267 | 0.3121 | 0.8900 | 333s | 179ms/step |
| 7 | 0.2136 | 0.9327 | 0.2955 | 0.8904 | 332s | 179ms/step |
| 8 | 0.1950 | 0.9395 | 0.3770 | 0.8560 | 323s | 174ms/step |
| 9 | 0.1922 | 0.9416 | 0.3262 | 0.8736 | 323s | 174ms/step |
| 10 | 0.1719 | 0.9453 | 0.3000 | 0.9007 | 334s | 180ms/step |
| 21 | 0.0525 | 0.9660 | 0.5652 | 0.9209 | 323s | 174ms/step |
| 22 | 0.0527 | 0.9647 | 0.5822 | 0.9213 | 323s | 174ms/step |
| 23 | 0.0524 | 0.9660 | 0.5949 | 0.9213 | 323s | 174ms/step |
| 24 | 0.0525 | 0.9647 | 0.5919 | 0.9206 | 321s | 173ms/step |
| 25 | 0.0525 | 0.9660 | 0.6032 | 0.9202 | 320s | 173ms/step |

Tabela 3.6: Gender Model2 Webcam Train Summary

Și în acest caz acuratețea finală a fost mai mică decât la antrenarea fără prelucrare dar a rezultat în rezultate mult mai bune în practică, anume 92.81%.



(a) Gender Model2 Webcam Train Results

(b) Age Split

Pentru clasificarea în grupe de vîrstă am utilizat aceleasi imagini, cu precizarea că am utilizat un alt random state la split. De asemenea, am modificat straturile de ieșire ale arhitecturilor, pentru a avea 8 clase în loc de 2, în rest fiind identice.

În loc de clasificare propriu zisă, o altă posibilă abordare ar fi una pe bază de regresie pentru determinarea vîrstei. Cu toate acestea, consider că acea abordare ar fi mai potrivită pentru un set de date care precizează vîrstă exactă a fiecărui individ, în loc de grupa de vîrstă. În practică rezultatele au fost satisfăcătoare și prin această metodă.

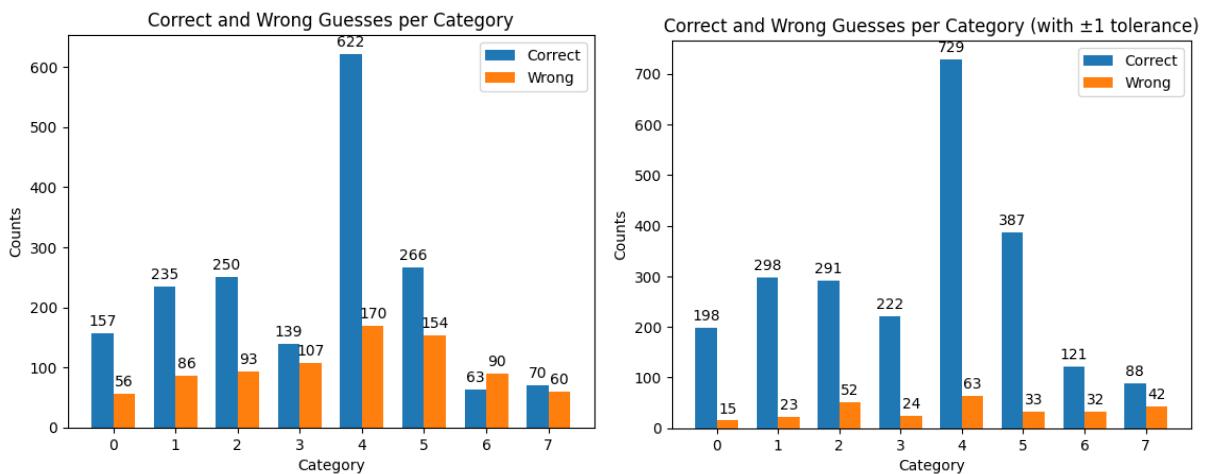
În cele ce urmează voi trece prin aceeași pașă ca la modelele pentru clasificarea genului. Prin urmare, prima versiune este cea pe bază de Conv2D, fără simularea camerei web.

Testele ajung la accuratețea de 68.83%. În practică, însă, am observat că modelele de vîrstă tind să prezică o grupă apropiată de vîrstă atunci când gresesc. Prin urmare, am rulat și o testare cu toleranță de o clasă, în care au fost considerate corecte și predicțiile ± 1 față de eticheta corectă. După acest standard, acuratețea este de 89.15%.

La testare pe imagini simulate, rezultatele au fost dezamăgitoare, de doar 27.04%. Își aici am rulat o testare cu toleranță, în care rezultatele au fost mai bune, de 60.65%.

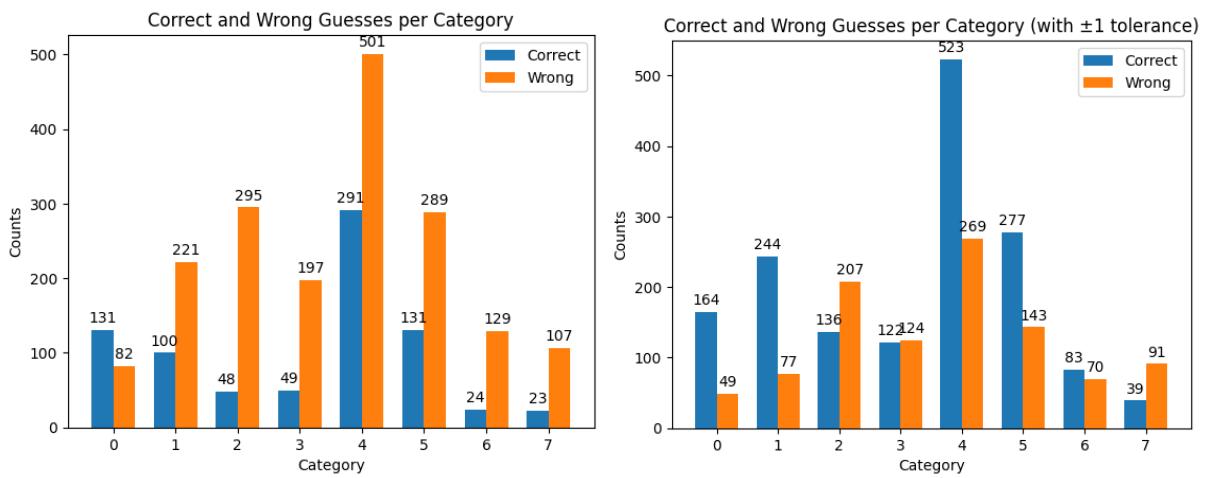
| Epoch | Loss | Accuracy | Val_Loss | Val_Accuracy | Time_Per_Epoch | Time_Per_Step |
|-------|--------|----------|----------|--------------|----------------|---------------|
| 1 | 1.8811 | 0.3168 | 1.6503 | 0.3808 | 47s | 101ms/step |
| 2 | 1.6203 | 0.3925 | 1.5100 | 0.4324 | 22s | 48ms/step |
| 3 | 1.4408 | 0.4445 | 1.4201 | 0.4729 | 22s | 48ms/step |
| 4 | 1.2865 | 0.5096 | 1.2783 | 0.5149 | 22s | 48ms/step |
| 5 | 1.1249 | 0.5724 | 1.1695 | 0.5630 | 22s | 48ms/step |
| 6 | 0.9820 | 0.6243 | 1.1034 | 0.5890 | 22s | 48ms/step |
| 7 | 0.8409 | 0.6806 | 1.0637 | 0.6146 | 22s | 48ms/step |
| 8 | 0.6991 | 0.7385 | 1.0891 | 0.6173 | 22s | 48ms/step |
| 9 | 0.5598 | 0.7901 | 1.0753 | 0.6241 | 22s | 48ms/step |
| 10 | 0.4442 | 0.8380 | 1.1081 | 0.6379 | 22s | 48ms/step |
| 21 | 0.0981 | 0.9705 | 1.1516 | 0.6841 | 21s | 46ms/step |
| 22 | 0.0980 | 0.9695 | 1.1490 | 0.6830 | 21s | 46ms/step |
| 23 | 0.0933 | 0.9729 | 1.1502 | 0.6830 | 21s | 46ms/step |
| 24 | 0.0961 | 0.9699 | 1.1526 | 0.6849 | 21s | 46ms/step |
| 25 | 0.0956 | 0.9718 | 1.1507 | 0.6841 | 21s | 46ms/step |

Tabela 3.7: Age Model1 Train Summary



(a) Age Model1 Results

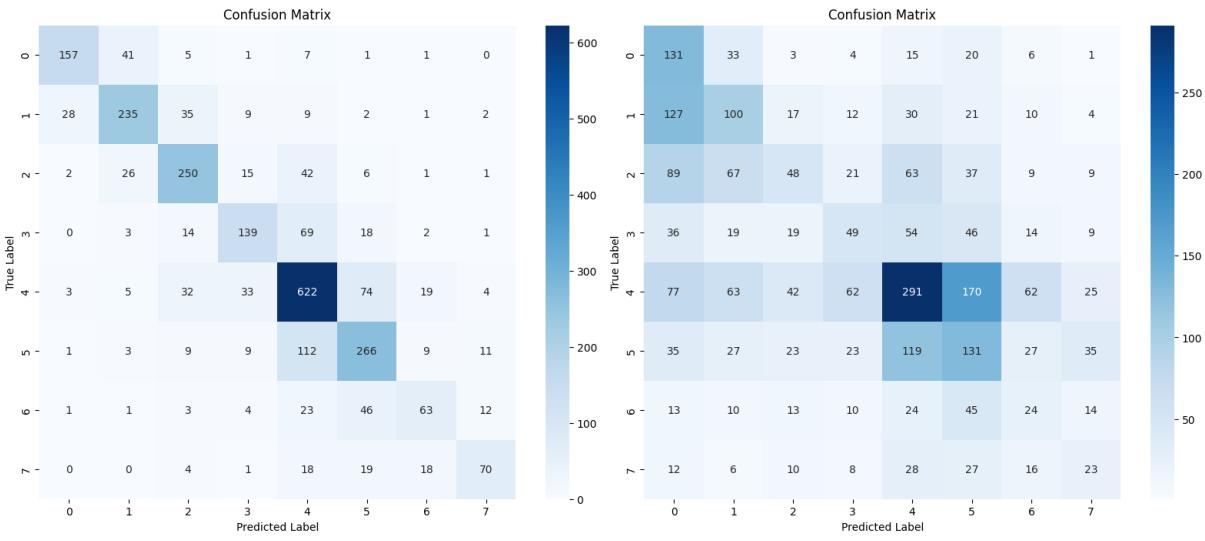
(b) Age Model1 Tolerance Results



(a) Age Model1 Webcam Test Results

(b) Age Model1 Webcam Tolerance Results

Mai exact, matricile de confuzie în cele 2 situații arată după cum urmează:



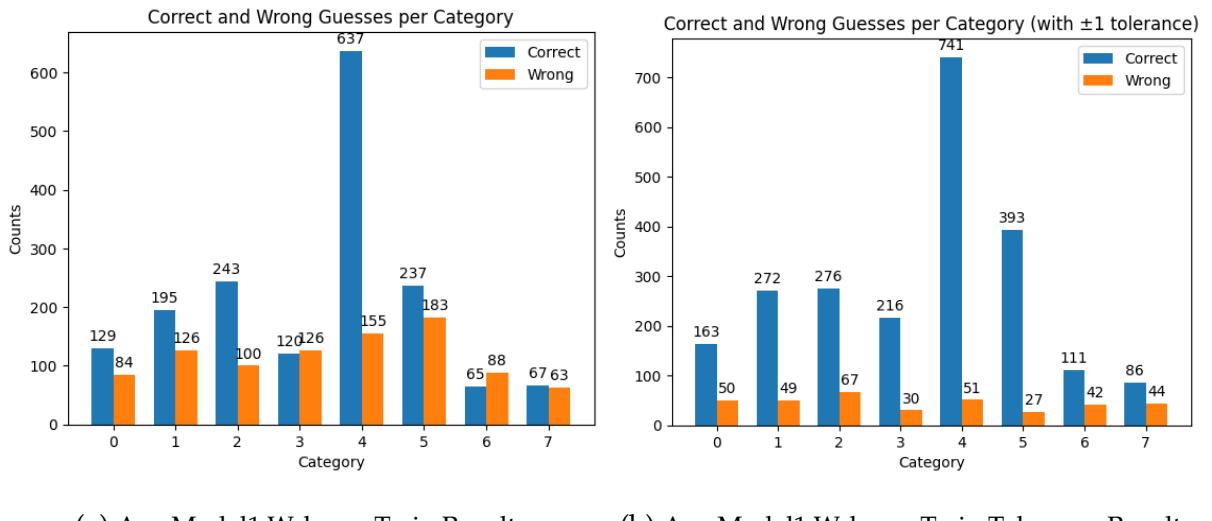
(a) Age Model1 Confusion Matrix

(b) Age Model1 Webcam Confusion Matrix

A urmat antrenarea pe imagini deja prelucrate. Rezultatele indică o acuratețe de 64.67%, respectiv 86.25% cu tolerantă:

| Epoch | Loss | Accuracy | Val_Loss | Val_Accuracy | Time_Per_Epoch | Time_Per_Step |
|-------|--------|----------|----------|--------------|----------------|---------------|
| 1 | 1.9509 | 0.2865 | 1.7497 | 0.3625 | 37s | 79ms/step |
| 2 | 1.7247 | 0.3599 | 1.6511 | 0.3930 | 20s | 43ms/step |
| 3 | 1.6002 | 0.4027 | 1.5371 | 0.4370 | 20s | 42ms/step |
| 4 | 1.4930 | 0.4446 | 1.4449 | 0.4725 | 20s | 42ms/step |
| 5 | 1.3823 | 0.4920 | 1.3923 | 0.4912 | 20s | 43ms/step |
| 6 | 1.2806 | 0.5275 | 1.3328 | 0.5153 | 20s | 42ms/step |
| 7 | 1.1664 | 0.5820 | 1.3760 | 0.5073 | 19s | 40ms/step |
| 8 | 1.0609 | 0.6246 | 1.2775 | 0.5474 | 19s | 42ms/step |
| 9 | 0.9552 | 0.6625 | 1.2484 | 0.5672 | 20s | 43ms/step |
| 10 | 0.8576 | 0.7062 | 1.2128 | 0.5886 | 20s | 42ms/step |
| 21 | 0.2930 | 0.8994 | 1.3376 | 0.6394 | 18s | 40ms/step |
| 22 | 0.2810 | 0.9003 | 1.3275 | 0.6432 | 19s | 40ms/step |
| 23 | 0.2637 | 0.9062 | 1.3304 | 0.6467 | 19s | 42ms/step |
| 24 | 0.2603 | 0.9052 | 1.3390 | 0.6429 | 18s | 40ms/step |
| 25 | 0.2430 | 0.9138 | 1.3430 | 0.6432 | 18s | 40ms/step |

Tabela 3.8: Age Model1 Webcam Train Summary



(a) Age Model1 Webcam Train Results

(b) Age Model1 Webcam Train Tolerance Results

Matricea de confuzie în acest caz este:

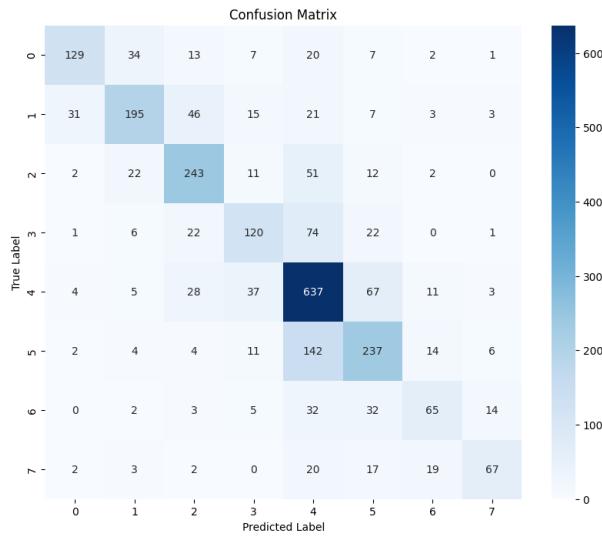
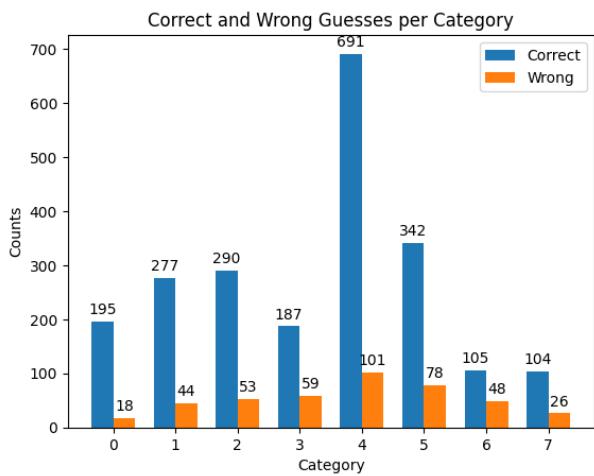


Figura 3.10: Age Model1 Webcam Train Confusion Matrix

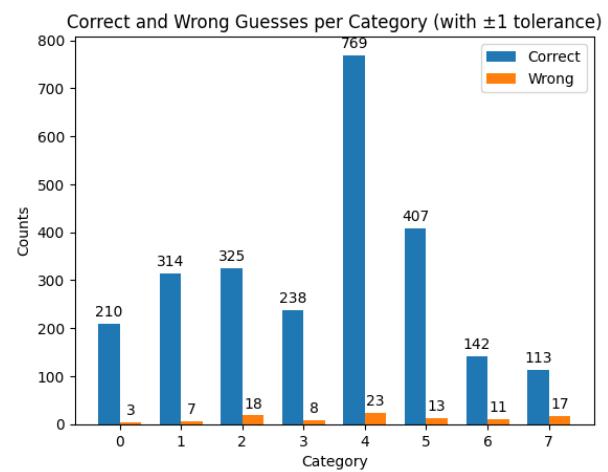
Aceleași operații au fost efectuate și pentru al doilea model, cel cu ResNet50. Acesta a atins accuratetea de 83.69%, respectiv 96.18% cu tolerantă. La testare pe imagini prelucrate de simulator, rezultatele au fost și de această dată semnificativ mai rele, de doar 56.80%. Și aici rezultatele cu tolerantă au fost mai bune, de 84.72%.

| Epoch | Loss | Accuracy | Val_Loss | Val_Accuracy | Time_Per_Epoch | Time_Per_Step |
|-------|--------|----------|----------|--------------|----------------|---------------|
| 1 | 1.3661 | 0.4821 | 1.1161 | 0.5798 | 334s | 180ms/step |
| 2 | 0.9483 | 0.6404 | 0.8448 | 0.6658 | 333s | 179ms/step |
| 3 | 0.7183 | 0.7264 | 0.8879 | 0.6872 | 329s | 177ms/step |
| 4 | 0.5526 | 0.7939 | 0.7537 | 0.7193 | 326s | 176ms/step |
| 5 | 0.4065 | 0.8523 | 0.8230 | 0.7108 | 317s | 171ms/step |
| 6 | 0.3115 | 0.8915 | 0.7614 | 0.7540 | 334s | 180ms/step |
| 7 | 0.2363 | 0.9175 | 0.8977 | 0.7422 | 316s | 170ms/step |
| 8 | 0.2016 | 0.9291 | 0.7535 | 0.7674 | 328s | 177ms/step |
| 9 | 0.1605 | 0.9477 | 0.8632 | 0.7716 | 327s | 176ms/step |
| 10 | 0.1512 | 0.9492 | 0.9029 | 0.7273 | 316s | 170ms/step |
| 21 | 0.0009 | 0.9999 | 0.9460 | 0.8335 | 325s | 175ms/step |
| 22 | 0.0010 | 0.9997 | 0.9480 | 0.8350 | 327s | 176ms/step |
| 23 | 0.0010 | 0.9999 | 0.9551 | 0.8369 | 327s | 177ms/step |
| 24 | 0.0007 | 0.9999 | 0.9589 | 0.8346 | 318s | 171ms/step |
| 25 | 0.0005 | 1.0000 | 0.9599 | 0.8354 | 318s | 171ms/step |

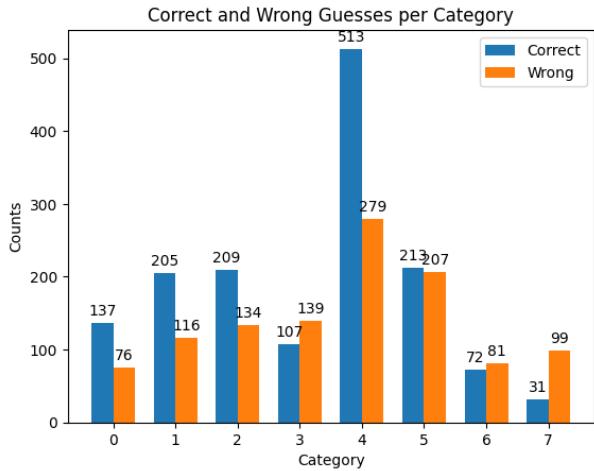
Tabela 3.9: Age Model2 Train Summary



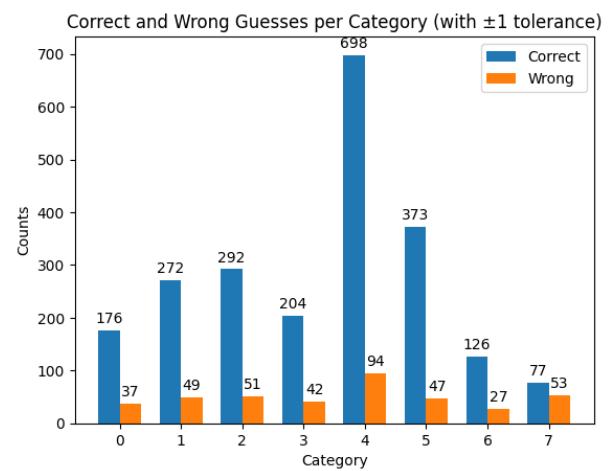
(a) Age Model2 Results



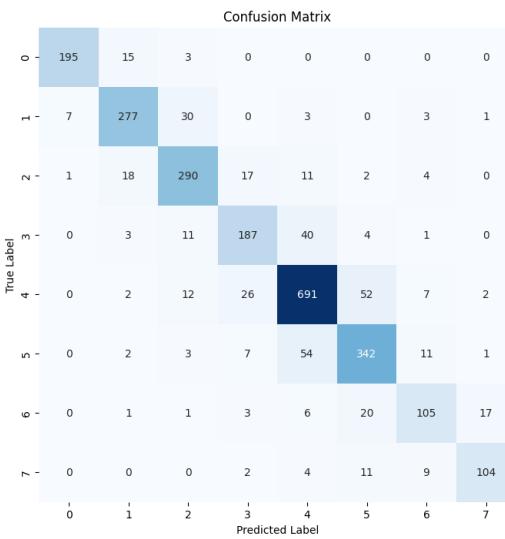
(b) Age Model2 Tolerance Results



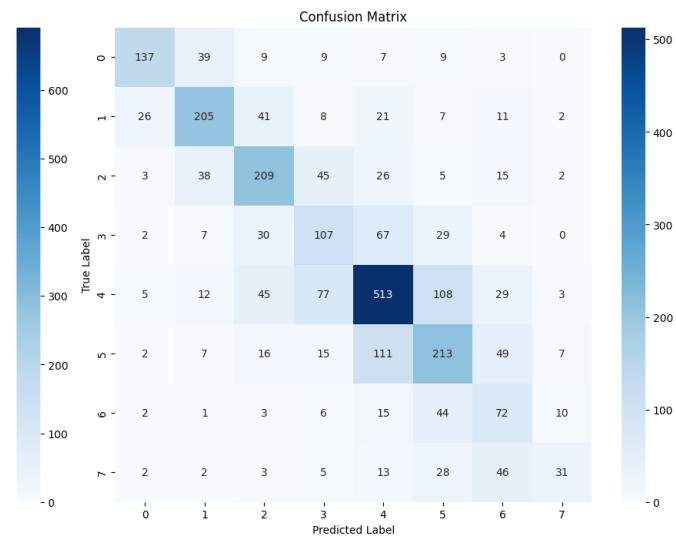
(a) Age Model2 Webcam Test Results



(b) Age Model2 Webcam Tolerance Results



(a) Age Model2 Confusion Matrix



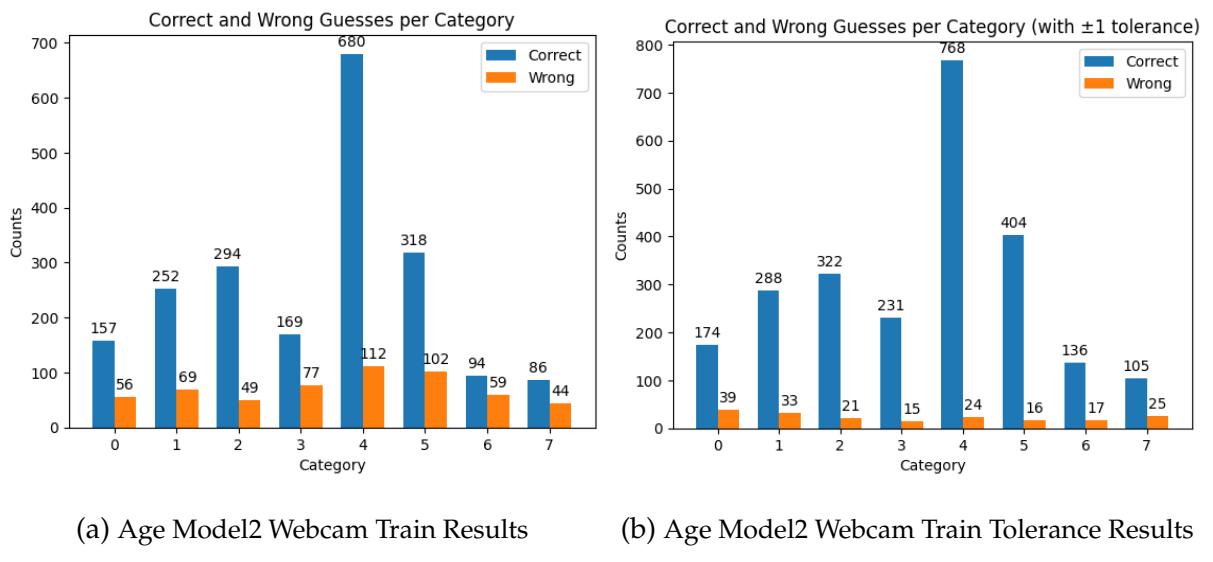
(b) Age Model2 Webcam Confusion Matrix

Antrenarea pe imaginile deja procesate de simulator a arătat în felul următor.

Acuratețea este de 78.30%, respectiv 92.74% cu toleranță:

| Epoch | Loss | Accuracy | Val_Loss | Val_Accuracy | Time_Per_Epoch | Time_Per_Step |
|-------|--------|----------|----------|--------------|----------------|---------------|
| 1 | 1.6122 | 0.4168 | 1.2933 | 0.5099 | 354s | 191ms/step |
| 2 | 1.2772 | 0.5385 | 1.1801 | 0.5707 | 352s | 190ms/step |
| 3 | 1.0997 | 0.6238 | 1.1352 | 0.6012 | 353s | 190ms/step |
| 4 | 0.9588 | 0.6796 | 1.0607 | 0.6375 | 350s | 189ms/step |
| 5 | 0.8276 | 0.7403 | 0.9700 | 0.6811 | 350s | 189ms/step |
| 6 | 0.7235 | 0.7822 | 1.0222 | 0.6772 | 340s | 183ms/step |
| 7 | 0.6309 | 0.8168 | 0.9170 | 0.6994 | 357s | 193ms/step |
| 8 | 0.5481 | 0.8476 | 0.9026 | 0.7112 | 352s | 190ms/step |
| 9 | 0.4897 | 0.8654 | 1.0429 | 0.6864 | 340s | 183ms/step |
| 10 | 0.4438 | 0.8734 | 0.9275 | 0.7074 | 338s | 182ms/step |
| 21 | 0.1347 | 0.9379 | 1.2059 | 0.7830 | 346s | 187ms/step |
| 22 | 0.1334 | 0.9393 | 1.1974 | 0.7800 | 335s | 181ms/step |
| 23 | 0.1322 | 0.9379 | 1.2187 | 0.7807 | 335s | 181ms/step |
| 24 | 0.1282 | 0.9409 | 1.2153 | 0.7792 | 336s | 181ms/step |
| 25 | 0.1292 | 0.9401 | 1.2138 | 0.7800 | 335s | 181ms/step |

Tabela 3.10: Age Model2 Webcam Train Summary



(a) Age Model2 Webcam Train Results

(b) Age Model2 Webcam Train Tolerance Results

Matricea de confuzie finală este:

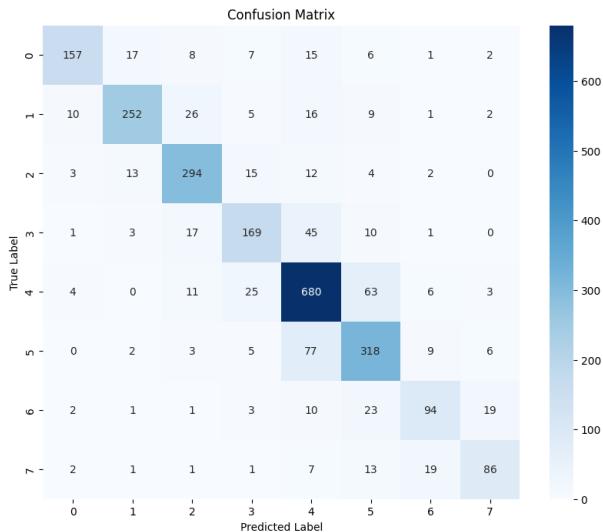


Figura 3.15: Age Model2 Webcam Train Confusion Matrix

Ultimul model este cel de detectare al emoțiilor pe baza expresiilor faciale. Lipsind o astfel de etichetare a primului set de date, am căutat un altul. Am reușit să găsesc 2 seturi, mai exact samples din seturi mai mari, pe platforma kaggle².

Primul set de date încercat a fost din Fer-2013³. Imaginele din acest set sunt alb negru, cu dimensiunea 48x48px, semnificativ mai mică decât cea a imaginilor din setul anterior. Cu toate acestea, am încercat să adaptez modul de antrenare deja cunoscut pentru a funcționa și în acest caz, dorind să văd ce rezultate pot obține.

În locul păstrării etichetelor în fișiere text, de data aceata setul de date conține 2 foldere, train și validation, fiecare cu 7 foldere denumite după 7 emoții, continând imaginile reprezentative emoției respective. Pentru antrenare sunt oferite 28821 imagini, iar pentru validare 7066. Această trăsătură mi-a permis citirea setului de date prin ImageDataGenerator din tensorflow, care poate genera o listă de tuple (imagină, etichetă). Prin setarea parametrului *class_mode* la *categorical* au rezultat etichete one hot encoded⁴, potrivite funcției de pierdere *categorical_crossentropy* la antrenare. Pe lângă acest parametru am specificat și modul culoare așteptat, anume *grayscale*, dimensiunea așteptată a imaginilor și m-am folosit și de *batch_size* pentru citirea mai rapidă, cât și de *shuffle* pentru a amesteca imaginile de etichete diferite într ele. Pentru a putea rula în continuare teste, însă, am separat imaginile și etichetele astfel citite în np arrays.

Emoțiile dispuse de acest set de date sunt angry, disgust, fear, happy, neutral,

²Kaggle

³Fer-2013

⁴One hot encoding

sad, surprise, în această ordine. Distribuirea lor în subseturi de antrenare și validare este:

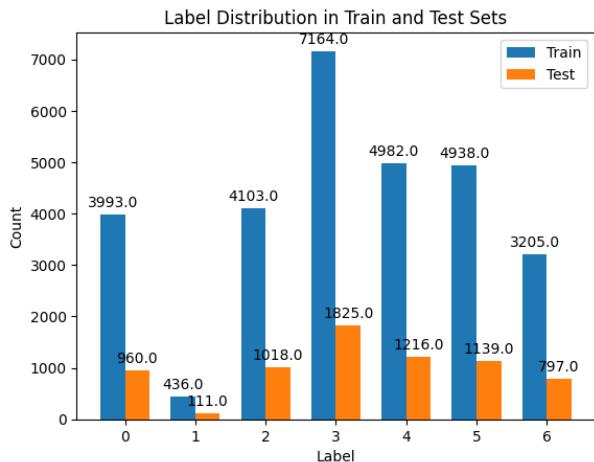


Figura 3.16: Emotion Set1 Split

Prima arhitectură încercată este asemănătoare cu prima pentru modelele anterioare, cu 3 modificări. În primul rând, am modificat *input_shape* la 48, 48, 1, din cauza dimensiunii și modului culoare ale acestor imagini. Din motive similare am setat și *kernel_size* al traturilor Conv2D la dimensiuni mai mici, anume 3, 3 unde anterior erau 5, 5, respectiv 2, 2 unde anterior erau 3, 3. În final am modificat și numărul de unități din ultimul strat, pentru a se potrivi clasificării în 7 clase.

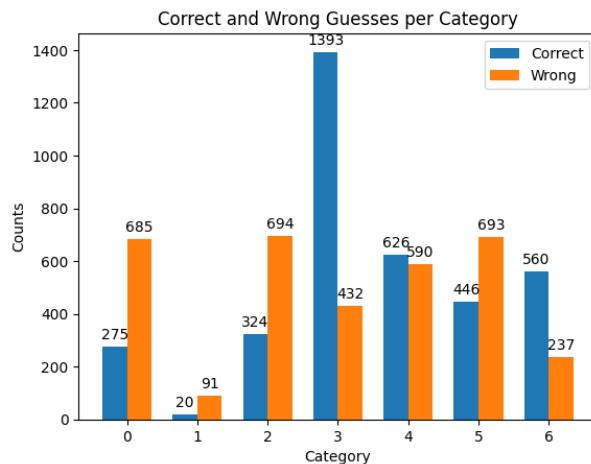
| Epoch | Loss | Accuracy | Val.Loss | Val_Accuracy | LR | Time_Per_Epoch | Time_Per_Step |
|-------|--------|----------|----------|--------------|------------|----------------|---------------|
| 1 | 1.8950 | 0.2517 | 1.7327 | 0.3374 | 1.0000e-04 | 17s | 75ms/step |
| 2 | 1.6729 | 0.3356 | 1.5984 | 0.3872 | 1.0000e-04 | 13s | 59ms/step |
| 3 | 1.5471 | 0.3984 | 1.5005 | 0.4232 | 1.0000e-04 | 13s | 59ms/step |
| 4 | 1.4563 | 0.4368 | 1.4401 | 0.4462 | 1.0000e-04 | 13s | 59ms/step |
| 5 | 1.3728 | 0.4722 | 1.4126 | 0.4601 | 1.0000e-04 | 13s | 59ms/step |
| 6 | 1.2986 | 0.5051 | 1.4019 | 0.4677 | 1.0000e-04 | 13s | 58ms/step |
| 7 | 1.2204 | 0.5376 | 1.3561 | 0.4843 | 1.0000e-04 | 13s | 59ms/step |
| 8 | 1.1440 | 0.5706 | 1.3466 | 0.4902 | 1.0000e-04 | 13s | 59ms/step |
| 9 | 1.0674 | 0.6022 | 1.3482 | 0.4939 | 1.0000e-04 | 14s | 60ms/step |
| 10 | 0.9857 | 0.6323 | 1.3537 | 0.5067 | 1.0000e-04 | 13s | 59ms/step |
| 46 | 0.5274 | 0.8165 | 1.4613 | 0.5130 | 1.0000e-11 | 12s | 54ms/step |
| 47 | 0.5272 | 0.8200 | 1.4616 | 0.5127 | 1.0000e-11 | 12s | 54ms/step |
| 48 | 0.5298 | 0.8180 | 1.4617 | 0.5125 | 1.0000e-11 | 12s | 54ms/step |
| 49 | 0.5298 | 0.8183 | 1.4619 | 0.5127 | 1.0000e-11 | 12s | 54ms/step |
| 50 | 0.5275 | 0.8176 | 1.4619 | 0.5130 | 1.0000e-11 | 12s | 54ms/step |

Tabela 3.11: Emotion Model1 Train Summary

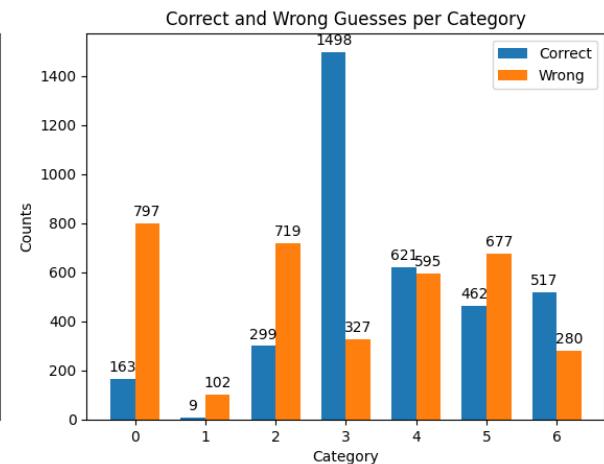
Modelele pentru clasificarea emoțiilor au fost cele mai grele de perfectionat. Prin urmare, pentru a le oferi șanse cât mai bune de a învăța, am decis dublarea numărului

de epoci, ajungând la 50.

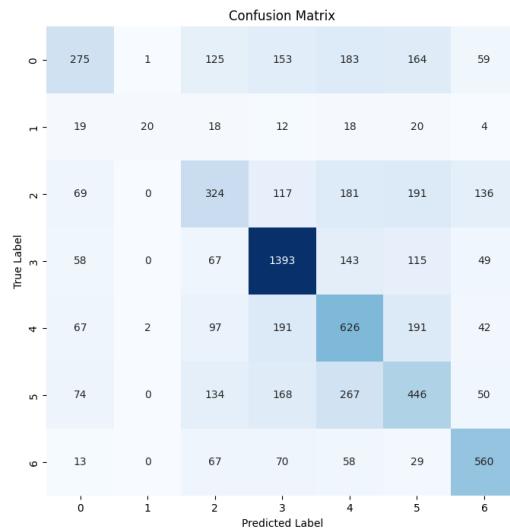
După cum indică diferența de 30% dintre acuratețea la antrenare și cea la validare, modelul este extrem de overfitted. Prin urmare, abordarea prezentată nu este cea mai potrivită pentru acest set de date. În orice caz, rezultatele testului pe clase sunt de 51.57%. Am testat și de această dată modelul pe imagini procesate de simulatorul camerei web, ajungând la o acuratețe de 50.51%.



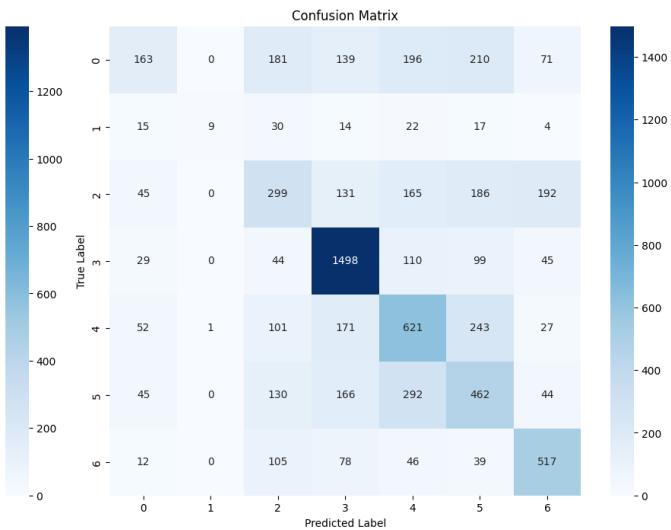
(a) Emotion Model1 Results



(b) Emotion Model1 Webcam Test Results



(a) Emotion Model1 Confusion Matrix

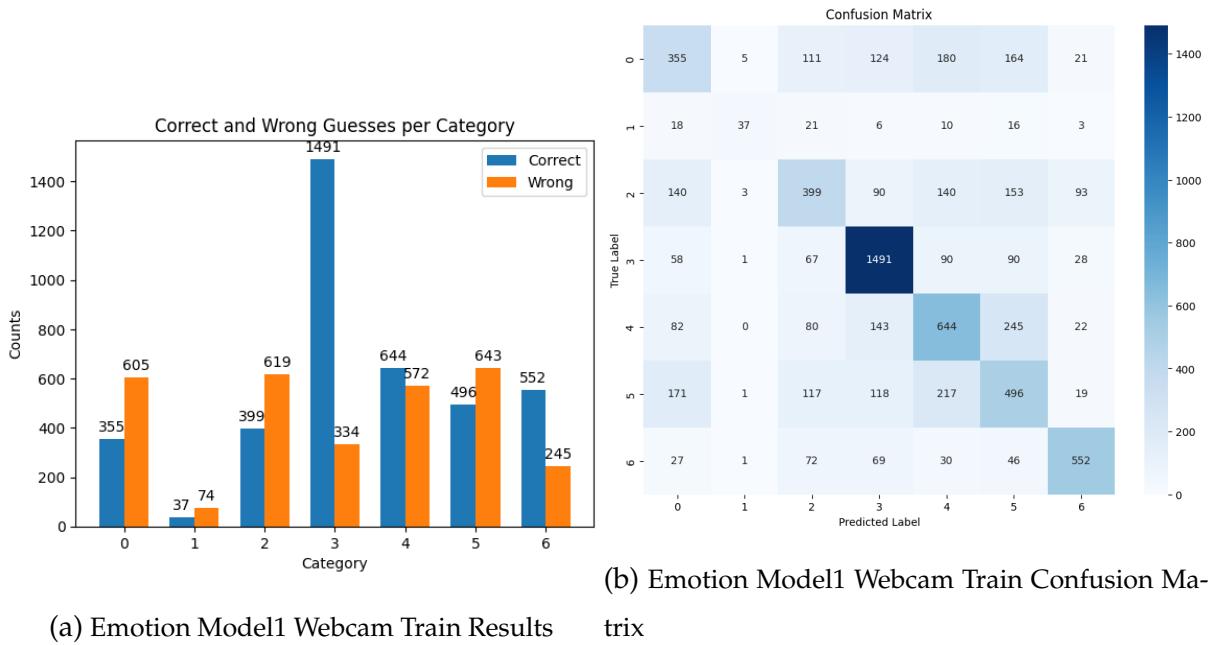


(b) Emotion Model1 Webcam Confusion Matrix

Antrenarea modelului pe imaginile deja prelucrate a dus la o acuratețe de 56.24%:

| Epoch | Loss | Accuracy | Val_Loss | Val_Accuracy | LR | Time_Per_Epoch | Time_Per_Step |
|-------|--------|----------|----------|--------------|------------|----------------|---------------|
| 1 | 1.8686 | 0.2489 | 1.6507 | 0.3510 | 1.0000e-04 | 16s | 18ms/step |
| 2 | 1.7182 | 0.3156 | 1.5433 | 0.4205 | 1.0000e-04 | 11s | 13ms/step |
| 3 | 1.6550 | 0.3534 | 1.4925 | 0.4510 | 1.0000e-04 | 12s | 13ms/step |
| 4 | 1.6118 | 0.3790 | 1.4389 | 0.4648 | 1.0000e-04 | 13s | 14ms/step |
| 5 | 1.5714 | 0.4020 | 1.3973 | 0.4805 | 1.0000e-04 | 11s | 12ms/step |
| 6 | 1.5344 | 0.4224 | 1.3627 | 0.5034 | 1.0000e-04 | 11s | 12ms/step |
| 7 | 1.4979 | 0.4419 | 1.3413 | 0.5034 | 1.0000e-04 | 10s | 11ms/step |
| 8 | 1.4636 | 0.4540 | 1.3044 | 0.5226 | 1.0000e-04 | 11s | 12ms/step |
| 9 | 1.4235 | 0.4705 | 1.2956 | 0.5161 | 1.0000e-04 | 10s | 11ms/step |
| 10 | 1.3831 | 0.4887 | 1.2960 | 0.5337 | 1.0000e-04 | 11s | 12ms/step |
| 46 | 0.8382 | 0.6798 | 1.2587 | 0.5595 | 1.0000e-10 | 10s | 11ms/step |
| 47 | 0.8401 | 0.6788 | 1.2592 | 0.5601 | 1.0000e-10 | 10s | 11ms/step |
| 48 | 0.8378 | 0.6788 | 1.2578 | 0.5599 | 1.0000e-10 | 10s | 11ms/step |
| 49 | 0.8367 | 0.6803 | 1.2579 | 0.5596 | 1.0000e-10 | 10s | 11ms/step |
| 50 | 0.8375 | 0.6807 | 1.2574 | 0.5594 | 1.0000e-10 | 10s | 11ms/step |

Tabela 3.12: Emotion Model1 Webcam Train Summary

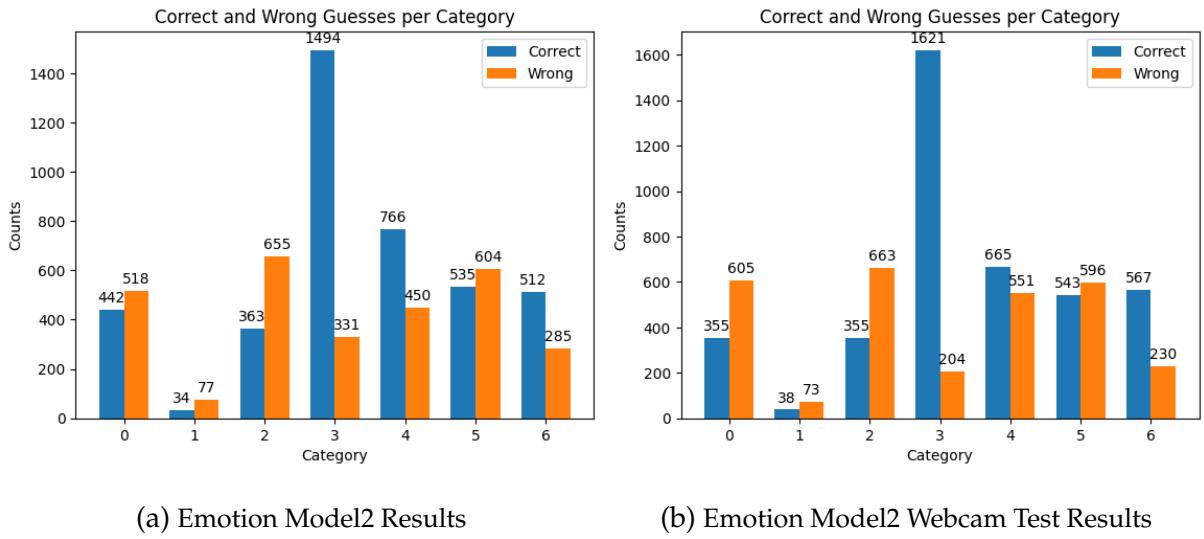


Din motive similare primei situații în care am folosit ResNet50, am încercat utilizarea lui și aici. Am întâmpinat însă o problemă: ResNet50 necesită imagini color, sau mai bine zis de forma $(x, x, 3)$. Am reușit să combat această limitare prin adăugarea unui strat Conv2D înaintea includerii ResNet50. De asemenea, dimensiunea extrem de redusă a imaginilor mi-a permis augmentarea datelor. Pentru augmentarea cât mai eficientă și de cost minim, am ales adăugarea de 2 straturi care modifică proprietățile imaginilor la fiecare epocă, aspect care ajută și la remedierea problemei de overfitting. Aceste straturi noi sunt poziționate înaintea celui Conv2D. Primul este

RandomRotation(factor = 0.20, seed = 100), care pe baza valorii seed rotește imaginile cu până la +- 0.2 din maximul de 360 de grade, adică aproximativ 70 de grade în ambele sensuri. Al doilea este *RandomFlip(mode = "horizontal", seed = 100)*, care pe baza valorii seed decide rotirea sau nu a imaginii pe orizontală. Antrenarea a dus la o acuratețe de 58.67%, respectiv 58.65% pe imagini procesate de simulator.

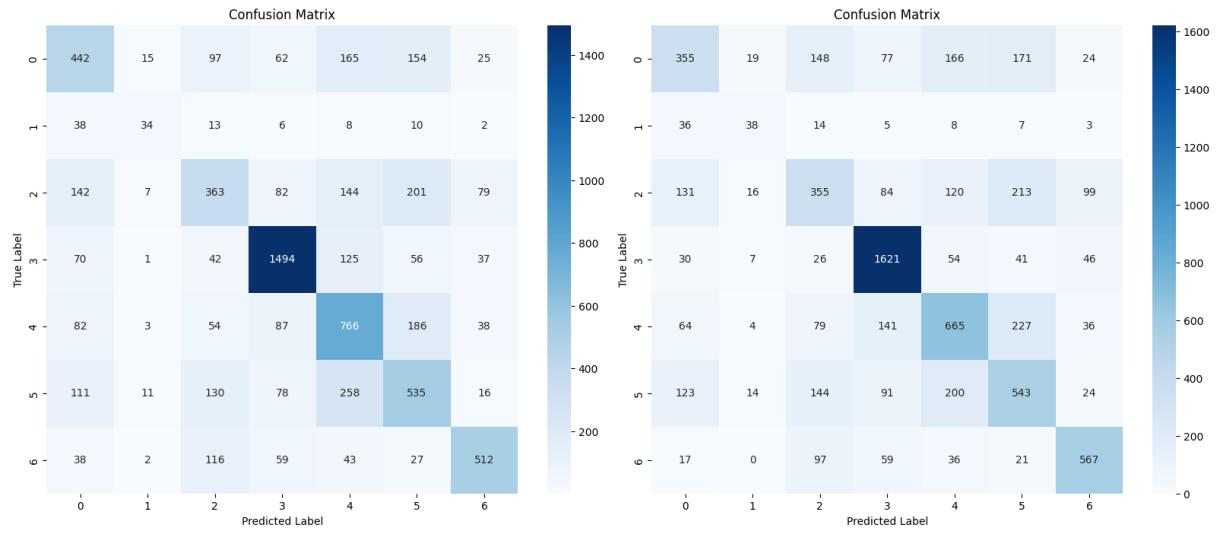
| Epoch | Loss | Accuracy | Val_Loss | Val_Accuracy | LR | Time_Per_Epoch | Time_Per_Step |
|-------|--------|----------|----------|--------------|------------|----------------|---------------|
| 1 | 2.0170 | 0.1973 | 1.8876 | 0.2344 | 1.0000e-04 | 303s | 1s/step |
| 2 | 1.8650 | 0.2203 | 1.8688 | 0.2041 | 1.0000e-04 | 283s | 1s/step |
| 3 | 1.8226 | 0.2376 | 1.8150 | 0.2426 | 1.0000e-04 | 301s | 1s/step |
| 4 | 1.8069 | 0.2460 | 1.7844 | 0.2639 | 1.0000e-04 | 300s | 1s/step |
| 5 | 1.7953 | 0.2519 | 1.7828 | 0.2570 | 1.0000e-04 | 290s | 1s/step |
| 6 | 1.7886 | 0.2569 | 1.7859 | 0.2605 | 1.0000e-04 | 367s | 2s/step |
| 7 | 1.7825 | 0.2609 | 1.7716 | 0.2757 | 1.0000e-04 | 506s | 2s/step |
| 8 | 1.7770 | 0.2613 | 1.7685 | 0.2678 | 1.0000e-04 | 291s | 1s/step |
| 9 | 1.7705 | 0.2658 | 1.7622 | 0.2738 | 1.0000e-04 | 293s | 1s/step |
| 10 | 1.7635 | 0.2717 | 1.7577 | 0.2799 | 1.0000e-04 | 302s | 1s/step |
| 46 | 0.7344 | 0.7317 | 1.1737 | 0.5868 | 1.0000e-06 | 303s | 1s/step |
| 47 | 0.7209 | 0.7366 | 1.1747 | 0.5860 | 1.0000e-06 | 292s | 1s/step |
| 48 | 0.7264 | 0.7338 | 1.1739 | 0.5851 | 1.0000e-06 | 293s | 1s/step |
| 49 | 0.7199 | 0.7363 | 1.1760 | 0.5845 | 1.0000e-06 | 292s | 1s/step |
| 50 | 0.7207 | 0.7385 | 1.1757 | 0.5846 | 1.0000e-06 | 292s | 1s/step |

Tabela 3.13: Emotion Model2 Train Summary



(a) Emotion Model2 Results

(b) Emotion Model2 Webcam Test Results



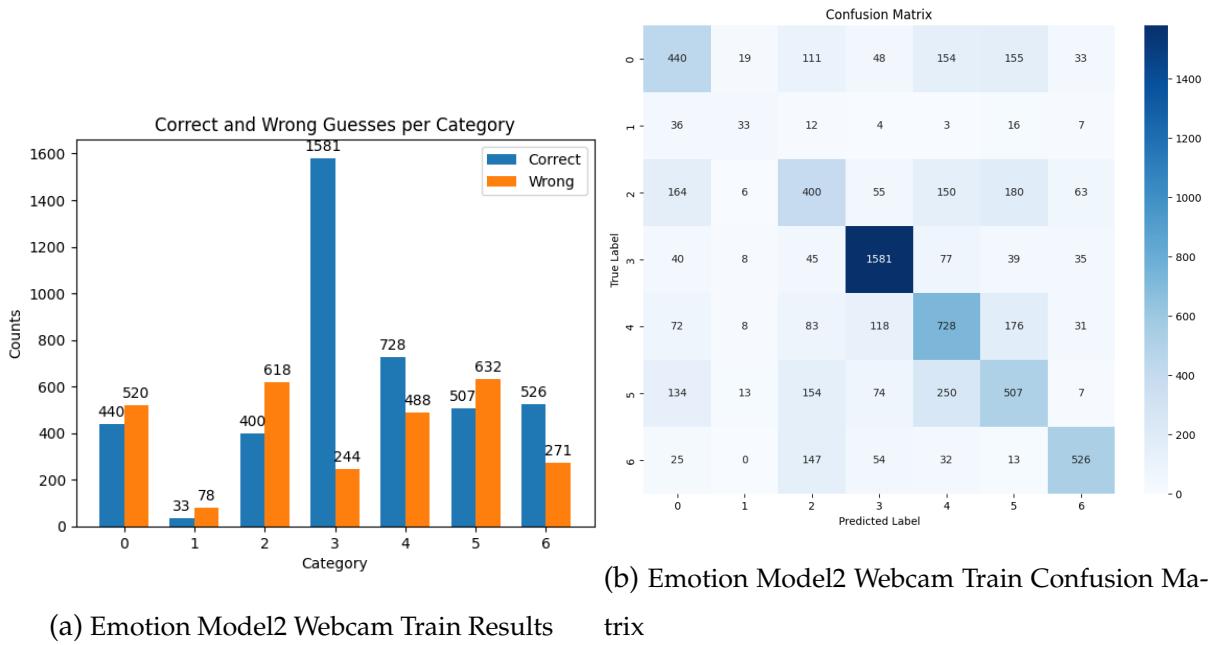
(a) Emotion Model2 Confusion Matrix

(b) Emotion Model2 Webcam Confusion Matrix

Antrenarea aceleiași arhitecturi pe imaginile deja procesate a avut următoarele rezultate, ducând la cea mai bună versiune obținută din acest set de date, cu o acuratețe așteptată de 59.65% în practică.

| Epoch | Loss | Accuracy | Val_Loss | Val_Accuracy | LR | Time_Per_Epoch | Time_Per_Step |
|-------|--------|----------|----------|--------------|------------|----------------|---------------|
| 1 | 1.8956 | 0.2378 | 1.6206 | 0.3640 | 1.0000e-04 | 341s | 95ms/step |
| 2 | 1.7612 | 0.3007 | 1.5882 | 0.3882 | 1.0000e-04 | 321s | 89ms/step |
| 3 | 1.7069 | 0.3288 | 1.5001 | 0.4526 | 1.0000e-04 | 322s | 89ms/step |
| 4 | 1.6737 | 0.3512 | 1.4636 | 0.4723 | 1.0000e-04 | 320s | 89ms/step |
| 5 | 1.6387 | 0.3755 | 1.4922 | 0.4676 | 1.0000e-04 | 310s | 86ms/step |
| 6 | 1.6118 | 0.3839 | 1.4681 | 0.4684 | 1.0000e-04 | 311s | 86ms/step |
| 7 | 1.5955 | 0.3954 | 1.4202 | 0.5050 | 1.0000e-04 | 322s | 89ms/step |
| 8 | 1.5755 | 0.4054 | 1.3901 | 0.5106 | 1.0000e-04 | 321s | 89ms/step |
| 9 | 1.5612 | 0.4127 | 1.4451 | 0.4870 | 1.0000e-04 | 312s | 86ms/step |
| 10 | 1.5505 | 0.4211 | 1.3849 | 0.5269 | 1.0000e-04 | 320s | 89ms/step |
| 46 | 1.3428 | 0.5345 | 1.2276 | 0.5882 | 1.0000e-05 | 312s | 87ms/step |
| 47 | 1.3410 | 0.5359 | 1.2212 | 0.5907 | 1.0000e-05 | 312s | 87ms/step |
| 48 | 1.3380 | 0.5350 | 1.2275 | 0.5877 | 1.0000e-05 | 312s | 87ms/step |
| 49 | 1.3362 | 0.5368 | 1.2324 | 0.5896 | 1.0000e-05 | 313s | 87ms/step |
| 50 | 1.3294 | 0.5391 | 1.2345 | 0.5825 | 1.0000e-05 | 312s | 87ms/step |

Tabela 3.14: Emotion Model2 Webcam Train Summary



După cum am enunțat, am încercat și antrenarea pe un alt set de date. Am găsit un sample din setul extensiv pentru clasificarea emoțiilor numit AffectNet⁵. Acesta conține imagini color, în rezoluție 96x96px. Versiunea găsită⁶ are 29042 imagini pentru 8 emoții: anger, contempt, disgust, fear, happy, neutral, sad, surprise. Deoarece ele nu vin împărțite în subseturi de antrenare și validare, am folosit încă o dată *train_test_split* cu aceleași specificații, dar split de 0.2 în loc de 0.15. Împărțirea astfel a imaginilor a rezultat în următoarea distribuție pe clase, cu un total de 23233 imagini de antrenare și 5809 de validare:

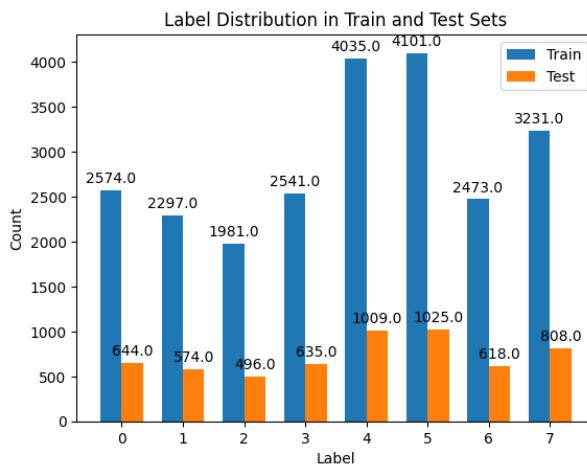


Figura 3.23: Emotion Set2 Split

Urmează aceeași pași ca în cazul setului anterior de date. Singurele diferențe sunt

⁵AffectNet

⁶AffectNet Sample

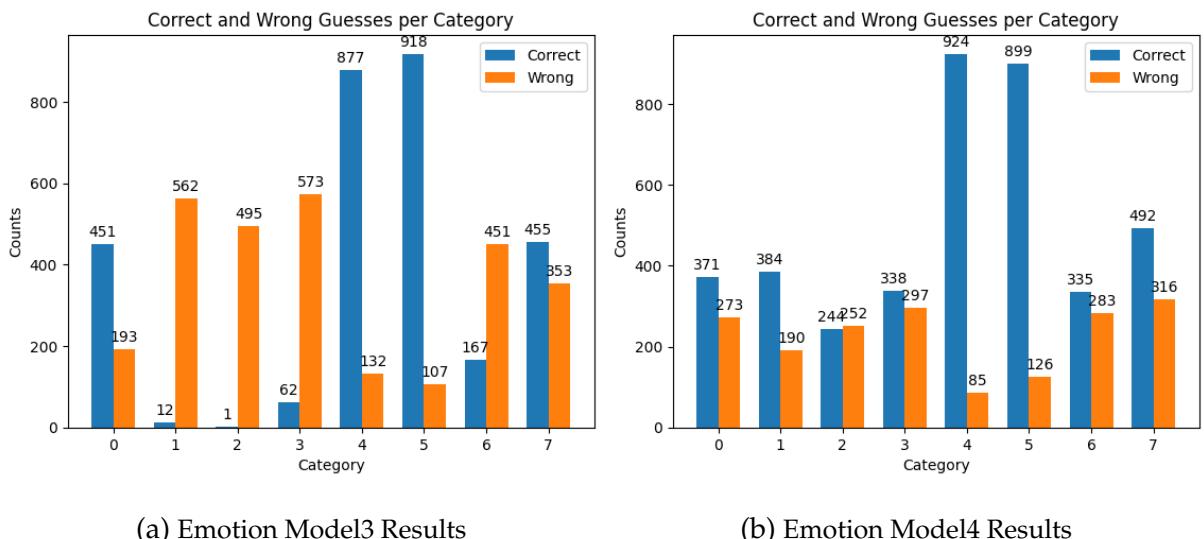
următoarele: pentru arhitectura pe baza de Conv2D, model3, folosesc și de această dată versiunea cu dimensiunile de kernel reduse, modificând doar *input_shape* la (96, 96, 3). La antrenarea acestui model am ales o rată de învățare de 0.001 în locul celei de 0.0001 folosite pentru toate celelalte, deoarece am observat că modelul învăță foarte încet în primele epoci. În cazul modelului pe bază de ResNet50, model4, am revenit la arhitectura folosită pentru modelele de clasificare a genului și vîrstei. De asemenea, am constatat că acest ultim model stagna după 25 de epoci, motiv pentru care nu l-am antrenat pentru 50. În următorul rînd, similar setului de date anterior, etichetele au fost prelucrate pentru a fi one hot, motiv pentru care și de această dată am utilizat funcția de pierdere *categorical_crossentropy*.

| Epoch | Loss | Accuracy | Val.Loss | Val.Accuracy | LR | Time_Per_Epoch | Time_Per_Step |
|-------|--------|----------|----------|--------------|------------|----------------|---------------|
| 1 | 1.9290 | 0.2592 | 2.5564 | 0.2054 | 0.0010 | 23s | 32ms/step |
| 2 | 1.6026 | 0.3465 | 1.6095 | 0.3880 | 0.0010 | 21s | 28ms/step |
| 3 | 1.5263 | 0.3763 | 1.3962 | 0.4378 | 0.0010 | 21s | 29ms/step |
| 4 | 1.4628 | 0.4016 | 1.4756 | 0.4243 | 0.0010 | 20s | 27ms/step |
| 5 | 1.4348 | 0.4130 | 1.3656 | 0.4471 | 0.0010 | 21s | 28ms/step |
| 6 | 1.3950 | 0.4241 | 1.4014 | 0.4126 | 0.0010 | 20s | 27ms/step |
| 7 | 1.3922 | 0.4318 | 1.3782 | 0.4488 | 0.0010 | 21s | 29ms/step |
| 8 | 1.3306 | 0.4497 | 1.2636 | 0.4682 | 0.0010 | 21s | 29ms/step |
| 9 | 1.3096 | 0.4533 | 1.5128 | 0.3786 | 0.0010 | 20s | 27ms/step |
| 10 | 1.2752 | 0.4677 | 1.4432 | 0.4483 | 0.0010 | 20s | 27ms/step |
| 46 | 1.0034 | 0.5680 | 1.2650 | 0.5058 | 1.0000e-09 | 20s | 27ms/step |
| 47 | 1.0005 | 0.5705 | 1.2642 | 0.5054 | 1.0000e-09 | 20s | 27ms/step |
| 48 | 0.9991 | 0.5701 | 1.2639 | 0.5054 | 1.0000e-09 | 21s | 28ms/step |
| 49 | 1.0011 | 0.5675 | 1.2647 | 0.5059 | 1.0000e-09 | 20s | 27ms/step |
| 50 | 0.9986 | 0.5674 | 1.2656 | 0.5054 | 1.0000e-09 | 21s | 28ms/step |

Tabela 3.15: Emotion Model3 Train Summary

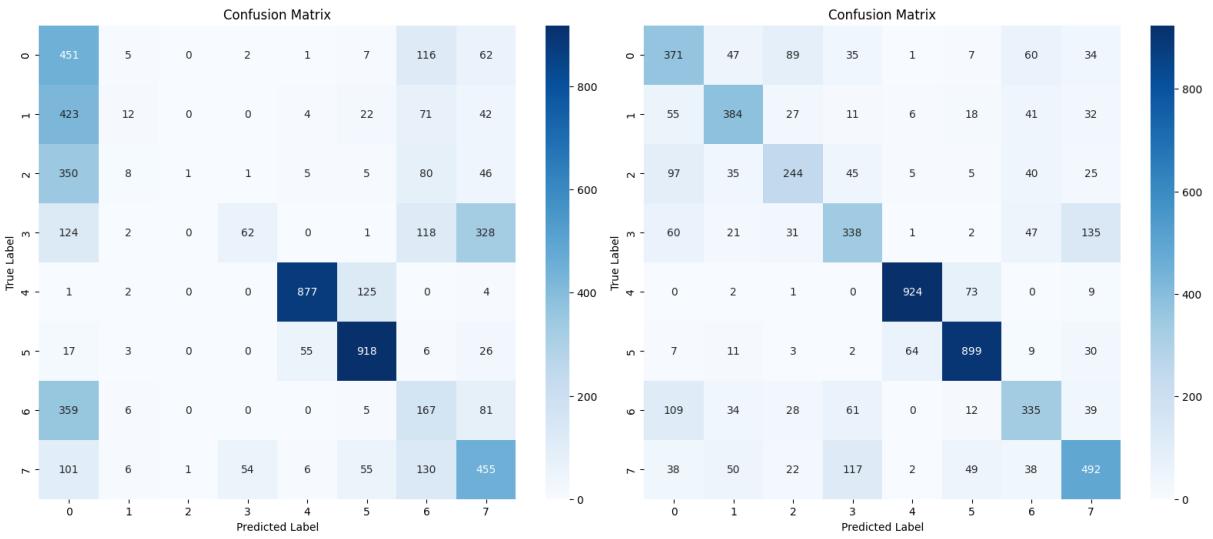
| Epoch | Loss | Accuracy | Val_Loss | Val_Accuracy | LR | Time_Per_Epoch | Time_Per_Step |
|-------|--------|----------|----------|--------------|------------|----------------|---------------|
| 1 | 1.5005 | 0.4510 | 1.1497 | 0.5739 | 1.0000e-04 | 270s | 93ms/step |
| 2 | 1.1445 | 0.5767 | 1.0342 | 0.6151 | 1.0000e-04 | 254s | 87ms/step |
| 3 | 0.9713 | 0.6486 | 1.0033 | 0.6283 | 1.0000e-04 | 260s | 90ms/step |
| 4 | 0.8321 | 0.7034 | 1.0086 | 0.6307 | 1.0000e-04 | 258s | 89ms/step |
| 5 | 0.6907 | 0.7577 | 1.1131 | 0.6017 | 1.0000e-04 | 249s | 86ms/step |
| 6 | 0.5485 | 0.8096 | 1.0184 | 0.6480 | 1.0000e-04 | 259s | 89ms/step |
| 7 | 0.4349 | 0.8489 | 1.1279 | 0.6688 | 1.0000e-04 | 257s | 88ms/step |
| 8 | 0.3498 | 0.8788 | 1.1269 | 0.6629 | 1.0000e-04 | 247s | 85ms/step |
| 9 | 0.1738 | 0.9424 | 1.2441 | 0.6853 | 1.0000e-05 | 260s | 90ms/step |
| 10 | 0.1080 | 0.9634 | 1.4440 | 0.6855 | 1.0000e-05 | 259s | 89ms/step |
| 21 | 0.0439 | 0.9830 | 1.9078 | 0.6853 | 1.0000e-07 | 244s | 84ms/step |
| 22 | 0.0434 | 0.9830 | 1.9192 | 0.6850 | 1.0000e-07 | 244s | 84ms/step |
| 23 | 0.0445 | 0.9828 | 1.9216 | 0.6848 | 1.0000e-07 | 244s | 84ms/step |
| 24 | 0.0431 | 0.9836 | 1.9177 | 0.6841 | 1.0000e-08 | 244s | 84ms/step |
| 25 | 0.0445 | 0.9819 | 1.9190 | 0.6848 | 1.0000e-08 | 244s | 84ms/step |

Tabela 3.16: Emotion Model4 Train Summary



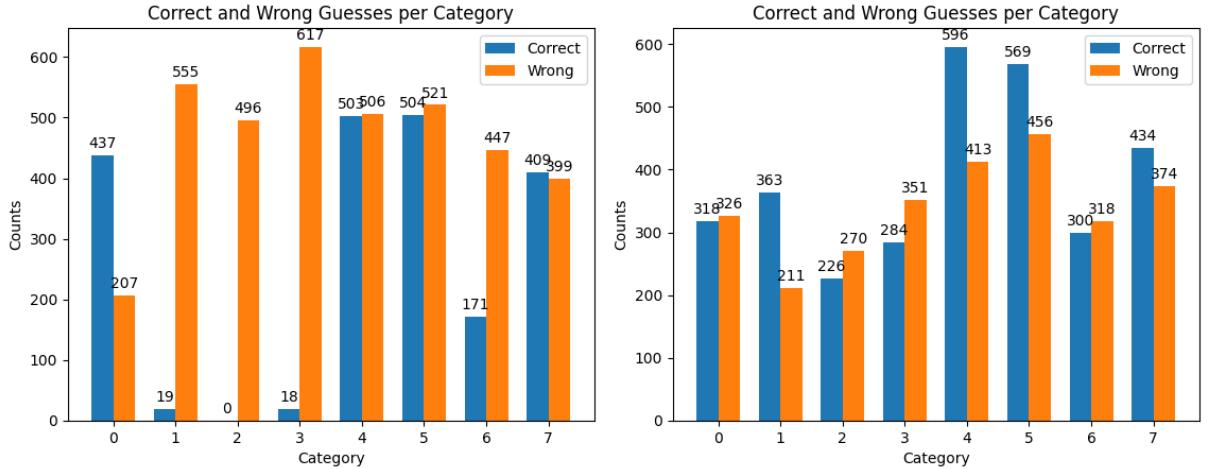
(a) Emotion Model3 Results

(b) Emotion Model4 Results



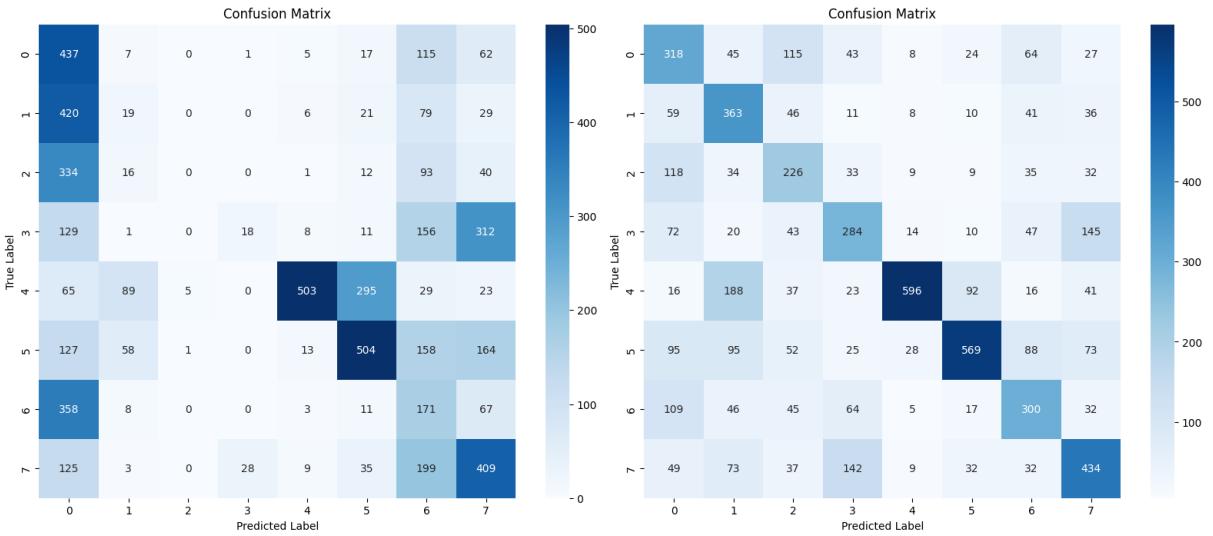
(a) Emotion Model3 Confusion Matrix

(b) Emotion Model4 Confusion Matrix



(a) Emotion Model3 Webcam Test Results

(b) Emotion Model4 Webcam Test Results



(a) Emotion Model3 Webcam Confusion Matrix

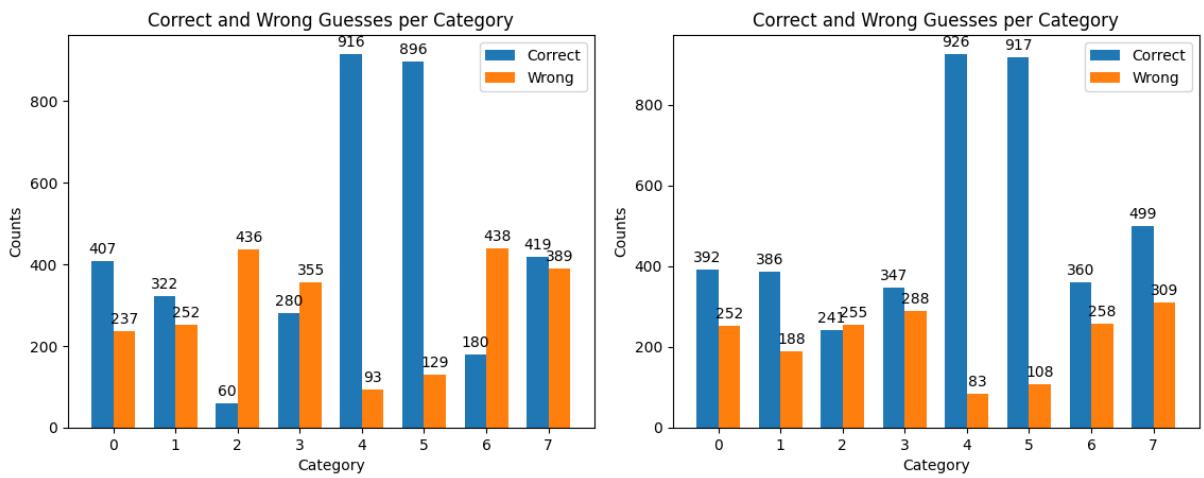
(b) Emotion Model4 Webcam Confusion Matrix

| Epoch | Loss | Accuracy | Val_Loss | Val_Accuracy | LR | Time_Per_Epoch | Time_Per_Step |
|-------|--------|----------|----------|--------------|----------|----------------|---------------|
| 1 | 1.9151 | 0.2702 | 1.9222 | 0.2699 | 0.0010 | 25s | 35ms/step |
| 2 | 1.6071 | 0.3619 | 1.6839 | 0.3481 | 0.0010 | 21s | 29ms/step |
| 3 | 1.5121 | 0.3790 | 1.6761 | 0.3255 | 0.0010 | 19s | 27ms/step |
| 4 | 1.4786 | 0.3886 | 1.5096 | 0.3983 | 0.0010 | 20s | 28ms/step |
| 5 | 1.4485 | 0.3953 | 1.7574 | 0.3474 | 0.0010 | 19s | 27ms/step |
| 6 | 1.4074 | 0.4101 | 2.0200 | 0.2892 | 0.0010 | 20s | 27ms/step |
| 7 | 1.4246 | 0.4038 | 1.4028 | 0.4219 | 0.0010 | 21s | 28ms/step |
| 8 | 1.3705 | 0.4245 | 1.6913 | 0.3636 | 0.0010 | 19s | 27ms/step |
| 9 | 1.3564 | 0.4205 | 1.3951 | 0.4166 | 0.0010 | 19s | 27ms/step |
| 10 | 1.3838 | 0.4157 | 1.5009 | 0.3954 | 0.0010 | 19s | 27ms/step |
| 46 | 0.7245 | 0.7012 | 1.2266 | 0.5981 | 0.000001 | 20s | 28ms/step |
| 47 | 0.7192 | 0.7018 | 1.2263 | 0.5982 | 0.000001 | 20s | 28ms/step |
| 48 | 0.7216 | 0.7018 | 1.2274 | 0.5982 | 0.000001 | 20s | 27ms/step |
| 49 | 0.7222 | 0.7020 | 1.2269 | 0.5981 | 0.000001 | 20s | 28ms/step |
| 50 | 0.7202 | 0.7018 | 1.2267 | 0.5982 | 0.000001 | 20s | 28ms/step |

Tabela 3.17: Emotion Model3 Webcam Train Summary

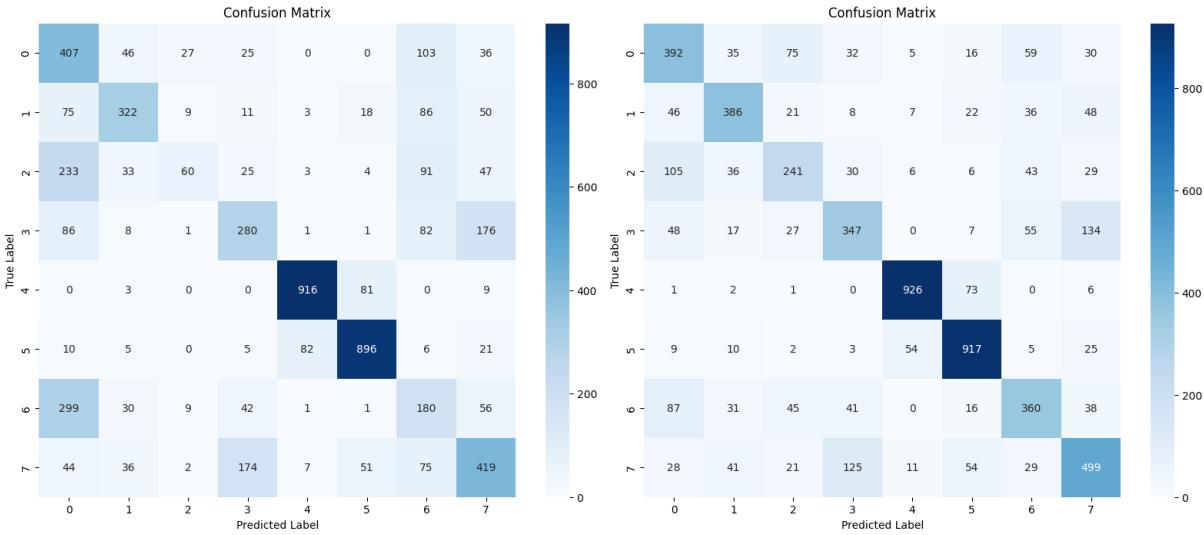
| Epoch | Loss | Accuracy | Val_Loss | Val_Accuracy | LR | Time_Per_Epoch | Time_Per_Step |
|-------|--------|----------|----------|--------------|-----------|----------------|---------------|
| 1 | 1.4508 | 0.4692 | 1.2913 | 0.5374 | 0.0001 | 271s | 93ms/step |
| 2 | 1.0891 | 0.5998 | 1.3055 | 0.5411 | 0.0001 | 259s | 89ms/step |
| 3 | 0.9384 | 0.6636 | 1.1749 | 0.5980 | 0.0001 | 259s | 89ms/step |
| 4 | 0.7903 | 0.7229 | 1.0044 | 0.6402 | 0.0001 | 258s | 89ms/step |
| 5 | 0.6522 | 0.7718 | 0.9866 | 0.6512 | 0.0001 | 259s | 89ms/step |
| 6 | 0.5212 | 0.8211 | 0.9918 | 0.6707 | 0.0001 | 259s | 89ms/step |
| 7 | 0.4089 | 0.8608 | 1.0268 | 0.6676 | 0.0001 | 249s | 86ms/step |
| 8 | 0.3139 | 0.8954 | 1.1601 | 0.6722 | 0.0001 | 259s | 89ms/step |
| 9 | 0.2564 | 0.9174 | 1.1768 | 0.6684 | 0.0001 | 250s | 86ms/step |
| 10 | 0.2189 | 0.9291 | 1.2927 | 0.6597 | 0.0001 | 248s | 85ms/step |
| 21 | 0.0078 | 0.9970 | 2.0257 | 0.7001 | 0.000001 | 242s | 83ms/step |
| 22 | 0.0074 | 0.9970 | 2.0513 | 0.6972 | 0.000001 | 242s | 83ms/step |
| 23 | 0.0064 | 0.9977 | 2.0438 | 0.6991 | 0.000001 | 242s | 83ms/step |
| 24 | 0.0078 | 0.9972 | 2.0353 | 0.6979 | 0.000001 | 242s | 83ms/step |
| 25 | 0.0067 | 0.9972 | 2.0491 | 0.6975 | 0.0000001 | 242s | 83ms/step |

Tabela 3.18: Emotion Model4 Webcam Train Summary



(a) Emotion Model3 Webcam Train Results

(b) Emotion Model4 Webcam Train Results

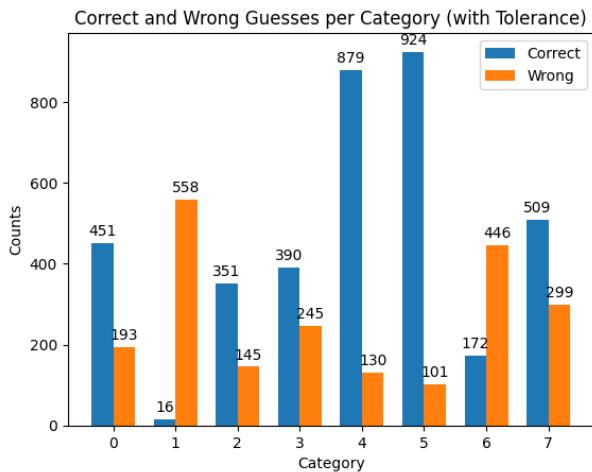


(a) Emotion Model3 Webcam Train Confusion Matrix (b) Emotion Model4 Webcam Train Confusion Matrix

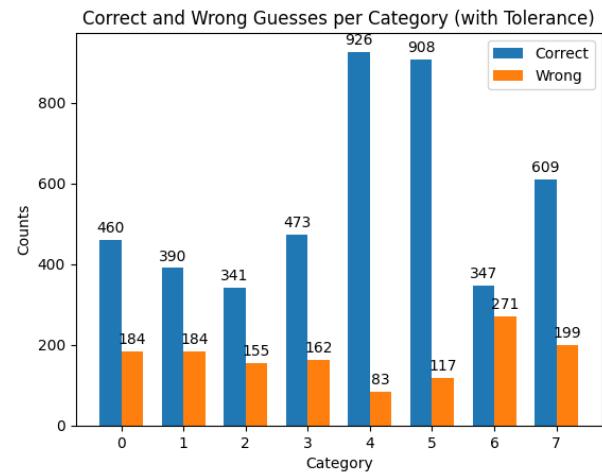
Acuratețea la validare a modelului 3 este 50.66%, iar a modelului 4 68.63%. La testarea pe imagini trecute prin simulator acuratețile sunt de 35.48% și 53.19%, iar la antrenarea pe imagini deja prelucrate rezultatele sunt de 59.90% și 70.02%.

Rezultatele la care am ajuns cu acest model pot părea semnificativ mai reale decât ale celoralte modele, dar este important de recunoscut faptul că problema detectării emoțiilor este mult mai complexă, iar o acuratețe de 70% în acest caz este foarte bună. Alte observații din testele manuale efectuate ar fi faptul că sentimentele par să fie grupate câte 2 în cazul celui de al doilea set de date: happy și contempt, neutral și sad, anger și disgust, fear și surprise. Sentimentele din aceste tuple sunt uneori confundate una pentru celalătă, dar extrem de rar pentru alt sentiment. Profitând de acest fapt am reușit să iau emoțiile în considerare ca perechi în cadrul jocului și să minimizez astfel

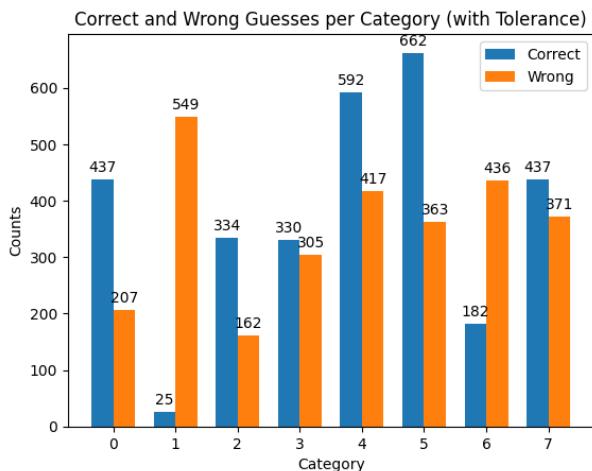
problema predicțiilor greșite în timpul jocului. Deși această constatare poate fi dedusă și din matricea de confuzie, am rulat o testare cu toleranță asupra acestor modele, pentru a vizualiza mai clar acuratețea lor în practică. Prin toleranță mă refer la faptul că au fost considerate corecte predicțiile contempt pentru imagini happy și invers, analog celorlalte perechi enumerate. Acuratețile cu toleranță ale modelelor sunt 63.55% și 76.67% pentru versiunile anterioare simulării camerei web, 51.62% și 67.32% pentru testarea lor pe imagini simulate, respectiv 70.63% și 78.10% pentru versiunile finale.



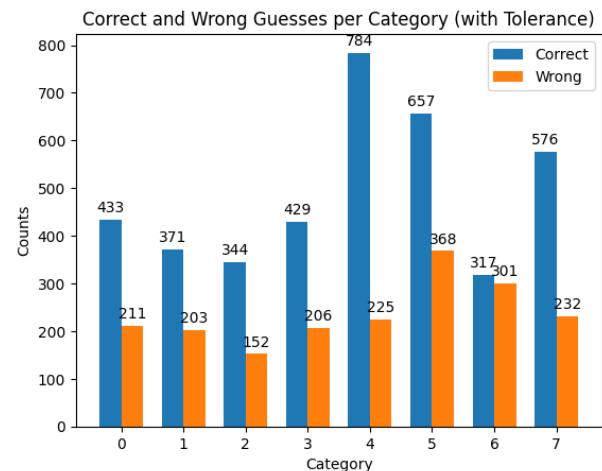
(a) Emotion Model3 Tolerance Results



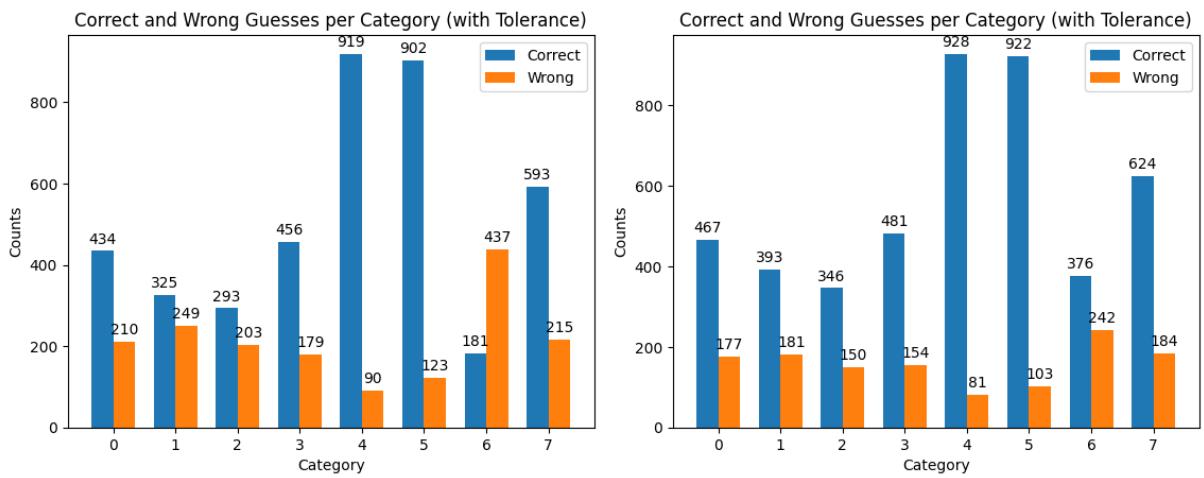
(b) Emotion Model4 Tolerance Results



(a) Emotion Model3 Webcam Tolerance Results



(b) Emotion Model4 Webcam Tolerance Results



(a) Emotion Model3 Webcam Train Tolerance Results (b) Emotion Model4 Webcam Train Tolerance Results

Modelele folosite în proiectul final pentru toate cele 3 trăsături sunt cele antrenate pe imagini trecute prin simulator, cu ResNet50, întrucât aveau acuratețea la testare cea mai mare.

3.2 Legătura

După finalizarea modelelor, a apărut problema realizării unei modalități de a rula predicțiile în timp real în timpul jocului și de a procesa rezultatele în Unity. Inițial am sperat că există un plugin sau o bibliotecă specializată pentru acest scop, dar din păcate am aflat că, deși se pot rula predicții nativ în C# și, printr-o extensie, Unity nu pot rula toate bibliotecile / funcțiile pe care doream să le folosesc în realizarea predicțiilor. Soluția la care am ajuns a fost de a realiza un program tot în Python, limbaj în care știam că vor funcționa toate lucrurile de care aveam nevoie, și de a realiza comunicarea dintre Python și C# la un moment ulterior. Își pentru această comunicare existau mai multe variante, dar am considerat comunicarea locală prin intermediul unui fișier text ca fiind o soluție suficient de bună și simplă de implementat, așa că nu am analizat alternativele.

În scriptul de predicții, pe care l-am numit *unity_comms.py*, aveam de ales ce informații doresc să transmit către joc. Am considerat urmatoarele informații ca fiind utile: dacă dispozitivul are sau nu cameră web, dacă este sau nu detectată o față în captura de ecran aleasă spre interpretare și următoarele aspecte ale rezultatelor predicțiilor: genul aparent al jucătorului, indexul grupei de vârstă, grupa de vârstă

estimata și emoția indicată de expresia facială. Aceste informații sunt salvate în fișierul *predictions.txt*, creat și salvat în aceeași locație ca executabilul aplicației finale. În cazul în care captura de ecran este realizată și interpretată cu succes, în fișierul menționat se scriu detaliile menționate legate de predicție în format json, în acea ordine. Nu am considerat că procente exacte de probabilitate merită transmise, dar pentru predicția emoțiilor, în scop de debug, le afișez în consolă. Tot aici doresc să menționez că rularea predicțiilor, mai exact progresul fiecărei rulări și timpul de executare, sunt afișate în mod default în consolă de funcția de predicție din tensorflow, dar nu am considerat această setare ca fiind nedorită, prin urmare, am păstrat-o. Fișierul conține un json cu aceleași câmpuri dar valori nule atunci cand nu este detectata nici o față în captura de ecran analizată și conține doar mesajul *NOWEBCAM* atunci când dispozitivul nu are deloc cameră și, deci, nu se pot prelua capturi.

În dezvoltarea acestui program intermediu am întâmpinat urmatoarele probleme:

În primul rând, încercarea de a rula în mod continuu, fără a aștepta între preluarea spre analiză a capturilor, producea rezultate nesatisfăcătoare din mai multe puncte de vedere. Dispozitivul fiind suprasolicităt, rezulta în încetinirea camerei web și potențial a altor programe, lucru care ar fi putut afecta performanța jocului. Pentru a rezolva această problemă am decis impunerea unui delay mic, anume de o secundă, între prelucrarea a două capturi. Această soluție a rezolvat și o altă problemă, anume de sincronizare a scrierii în fișier cu interpretarea fișierului text în timpul jocului.

În al doilea rand, transmiterea directă a capturilor de ecran nu semăna suficient cu imaginile pe care le-am utilizat inițial în procesul de antrenare. Inițial nu am rea- lizat prelucrarea imaginilor cu *face_recognition*, așa că în această etapă a dezvoltării aplicației am încercat mai multe opțiuni de redimensionare și ajustare a capturii. Am observat că modul de prelucrare a capturii pentru acuratețe optimă era ușor diferit la modelele pentru gen și vîrstă față de cel pentru emoții. Prima soluție aparentă pentru a rezolva această dificultate ar fi fost prelucrarea imaginilor în mai multe moduri complet distințe, pornind în ambele cazuri de la imaginea originală, și specializând acele prelucrări pentru un model anume. Această soluție ar fi cauzat, însă, probleme în ceea ce privește viteza de rulare a programului. Prin urmare, am căutat o soluție mai eficientă. Pentru a uniformiza rezultatele, adică a asigura o performanță satisfăcătoare a tuturor modelelor, dar în manieră optimizată, am decis să grupez pe cât posibil pașii de prelucrare. În primul rând din cauza naturii setului de date folosit pentru primele 2 modele acestea păreau specializate pe fețe orientate drept față de

cameră, nerotide. Pentru a trece peste această limitare am realizat o funcție de rotire a capturilor pentru a îndrepta fețele detectate înainte de orice alte prelucrări. Această funcție se bazează pe locația ochilor persoanei din imagine, aşa cum sunt detectate de *shape_predictor_68_face_landmarks.dat* din libraria dlib și rotește imaginea astfel încât linia imaginară dintre ochi să fie orizontală. În al doilea rând, pentru toate modelele este necesară tăierea capturii pentru a analiza doar față, lucru realizat cu biblioteca *face_recognition*. Urmează, tot în cazul ambelor modele, să preiau și putin din fundamental, deoarece am considerat că marginile detectate de către funcția din acea bibliotecă sunt prea apropiate și elimină unele trăsaturi. Aici, însă, a intervenit prima diferență: modelul pentru emoții preferă imagini mai apropiate decât celelalte două. Soluția în această versiune a programului a fost să separ aici prelucrarea imaginii în două etape: una pentru gen și vîrstă, în care imaginea este lărgită cu 25 de pixeli în toate direcțiile, și una pentru emoții, în care imaginea este mărită cu 10 pixeli în toate direcțiile. Tot aici doresc să menționez că încercarea de a lărgi zona considerată ca față din imagine uneori ducea la rezultate out of bounds. Ajustarea a fost simplă: în condițiile în care (0,0) este considerat stânga sus, am preluat maximul dintre coordonatele ajustate și 0 pentru coordonatele sus și stânga, respectiv minimul dintre ajustare și dimensiunea originală pentru jos și dreapta.

Ajustarea pe bază de pixeli, însă, nu s-ar fi comportat la fel pe toate dispozitivele, deoarece camerele folosite de utilizatori diferiți pot avea rezoluții diferite. Prin urmare am ajustat codul pentru a realiza lărgirea imaginii pe bază de procent. Intenționând să reantrenez modelele să includă aceeași prelucrare, pentru a unifica și acest pas al prelucrării capturilor, procentul în versiunea finală este același pentru toate modelele: 5% din înălțimea capturii.

După realizarea noilor modele am decis, totuși, să păstreze funcția de rotire a camerei doar în acest program. În primul rând doream ca modelele mele să aibă o oarecare abilitate de generalizare, chiar și când imaginile nu sunt drepte. Cu toate acestea am considerat că, în practică, rezultatele vor da randament maxim dacă păstrează această optimizare, care de altfel nu pare să fie costisitoare din punct de vedere al timpului de execuție.

Ultimul pas de prelucrare, separat pentru ambele imagini obținute până în acest moment, este o simplă redimensionare pentru a se potrivi cu dimensiunea așteptată a datelor de intrare a arhitecturilor fiecărui model, anume de (224, 224, 3), respectiv (96, 96, 3).

În ceea ce privește tehnologii folosite dar până acum nementionate, pentru utilizarea camerei am folosit VideoCapture și pentru rotirea imaginii warpAffine, amândouă din biblioteca cv2. VideoCapture are ca rezultat preluarea și nevoia controlului complet asupra camerei dispozitivului, lucru care înseamnă că va eșua preluarea capturilor în cazul în care camera este deschisă în alt program.

O problemă pentru care nu am găsit nici o soluție este faptul că *face_recognition* are uneori șansa să nu detecteze prezența unei fețe dacă aceasta este rotită la un unghi neobișnuit sau dacă luminozitatea este prea slabă sau camera are anumite defecte. Nu am găsit o alternativă mai bună, așa că aplicația finală folosește în continuare această bibliotecă.

După realizarea programului a urmat etapa de asigurare a ușurinței de utilizare, exportare și compatibilitate cu alte dispozitive. Cu alte cuvinte, a fost necesara reabilitarea unui executabil, în primul rând pentru programul discutat mai sus, dar mai târziu și unul care să lanseze simultan și jocul și acest program auxiliar. Executabilele au fost realizate, în ambele cazuri, cu pylauncher. Acesta asigură exportarea tuturor dependențelor necesare rulării programului, dar necesită o mică modificare manuală. După realizarea executabilelor am mutat modelele în folderul care conține executabilul, deoarece ele sunt importate în program utilizând calea relativă și altfel nu erau detectate.

În plus mentionez și faptul că nu am impus utilizarea CUDA în acest script. Consider că instalarea CUDA pe un dispozitiv nu este ceva ce ar trebui să vină o dată cu instalarea unui joc. Predicțiile rulează în aproximativ 60ms/predictie pe un intel i7 8th gen, viteză suficient de mare pentru scopurile aplicației.

3.3 Jocul

Un pas important în dezvoltarea aplicației a fost alegerea unui concept realist realizabil de o singură persoană într-un timp scurt. Ca și jocul are doar 15 nivele și estimez că poate fi completat în 2 ore. Ca idei generale, este un platformer 2D împărțit în 3 zone care încep cu nivele scurte, introductory, în care jucătorul are destule resurse pentru a testa efectele abilității pe inamici și are șansa de a vedea și testa mecanica principală a zonei. La mijlocul zonei este introdusă o mecanică secundară, iar ultimul nivel este contra cronometru pentru a testa abilitățile pe care le-a învățat jucătorul. Scopul jucatorului este de a descoperi calea de parcurgere a nivelului, folosindu-se de

mecanici și de inamici prin abilitatea specială controlată prin mimica fetei. La final jucatorul este readus la meniul principal, dar cu felicitările meritate.



Figura 3.33: Main menu

Primul aspect abordat a fost meniul principal, acela care apare în urma deschiderii jocului. Acesta conține opțiunea de a începe un joc nou, de a continua de unde s-a rămas și de a schimba setările. În momentul închiderii jocului sunt salvate într-un fișier de tip json setările jucătorului și numele ultimului nivel început. Aceste operații sunt realizate în scriptul SaveManager, aflat în fiecare nivel, prin apelarea funcției OnApplicationQuit. Astfel este posibilă continuarea jocului, însă nu există un buton manual de salvare. La continuarea jocului se reia de la început nivelul la care rămăsese anterior jucătorul și se aplică setările. Inabilitatea de a selecta un nivel anume și de a avea mai multe salvări este combatată de faptul că jocul este relativ scurt. Setările includ abilitatea de a forța neutilizarea camerei, de a alege aspectul protagonistului și de a selecta dificultatea. Cu alte cuvinte, setările au fost dezvoltate din necesitatea compatibilității jocului cu dispozitive care nu dispun de cameră sau în eventualitatea în care nu poate fi folosită temporar. Astfel, toate aspectele pe care jucătorul le poate impune manual prin meniuri sunt în mod ușor controlate de predicții. Aceste setări sunt salvate ca PlayerPrefs și accesate de scriptul de interpretare a predicțiilor, pentru a verifica dacă funcționalitățile vor fi bazate pe predicții sau pe setările alese de jucător. În eventualitatea lipsei unei camere se setează automat forțarea în PlayerPrefs ca adevărată, indiferent de opțiunea selectată de jucător în această privință, luându-se în considerare doar opțiunile jucătorului legate de aspect și dificultate.

Fiecare nivel se află într-o scenă separată care are un obiect cu scriptul numit InterpretWebcam. La inițializare acesta citește conținutul fișierului cu predicții și populează cu el un istoric de 10 predicții de gen și vîrstă, mai exact indexii grupelor de vîrstă. Prin urmare, inițial acest istoric este fals, fiind luată în considerare doar o instantă de predicții pentru realizarea sa, care este plasată pe toate cele 10 poziții.

Acest istoric este actualizat corespunzător pe parcus. O dată pe secundă este recitat conținutul fișierului de comunicare și sunt schimbate valorile din istoricul celor două atribută, câte una per atribut. Dacă opțiunea de forțare este dezactivată, cu alte cuvinte setată drept fals, și este detectată o cameră conform programului auxiliar, sunt actualizate valorile expuse public astfel: genul pe baza majorității din istoric, vârsta pe baza mediei indexilor din istoric și în cazul emoțiilor este luată în considerare doar ultima valoare citită.

Aspectul protagonistului este primul lucru afectat de predictiile IA. În orice moment, în scenă se află și protagonistul cu aspect feminin și unul cu aspect masculin. Funcționalitatea acestei implementări se bazează pe faptul că cele două obiecte nu sunt niciodată active simultan. Există un script care urmărește schimbarea valorii din InterpretWebcam și activează, respectiv dezactivează, modelele când este cazul, poziționând modelul anterior inactiv, care nu putea fi mișcat, în poziția exactă în care se află modelul anterior activ. Tranzitia poate fi realizată în orice moment, inclusiv în mijlocul nivelului dacă vine o altă persoană în fața camerei. Camera urmărește acel obiect din cele 2 care este activ. Funcționalitatea obiectelor nu diferă în nici un fel, având aceleași componente pe lângă cele care afectează aspectul.

La fiecare inițiere de nivel se aleg valorile HP și MP pe baza dificultății indicate de InterpretWebcam. Dificultatea este decisă pe baza vîrstei. Mai exact, pentru cei cu vîrstă sub 13 sau peste 60 inclusiv, dificultatea este setată pe "easy", caz în care jucătorul începe cu 5 HP și 5 MP, respectiv 3 și 3 în caz contrar, la dificultatea numită "normal". Verificarea vîrstei este efectuată și în momentul ridicării poțiunilor roșii pentru HP, respectiv albastre pentru MP, care cresc rezerva respectivei resurse cu 2 pentru easy și 1 pentru normal. Acestea sunt toate diferențele dintre cele dificultăți.

Semnificația resurselor este următoarea: în momentul în care un jucătorul intră în contact cu un inamic, acesta pierde 1 HP, acronim pentru hit sau health point. În momentul în care ajunge la 0 HP personajul "moare" și nivelul reîncepe. În ultima zonă există un pericol adăugat, deoarece inamicii pot ataca de la distanță. Termenul MP reprezintă magic sau mana points, care sunt consumate câte 1 pentru fiecare utilizare a abilității speciale. Consumarea 1 MP este necesară activării abilității. Drept consecință, când jucătorul rămâne fără MP, abilitatea nu se activează în momentul apăsării butonului aferent.

Ultima parte de IA este detectarea emoțiilor, care sunt folosite în cadrul jocului pentru abilitatea specială. Aceasta reprezintă elementul cheie al jocului. În momentul

în care jucătorul activează abilitatea se întâmplă simultan mai multe lucruri.

În primul rând, timpul se oprește pentru toate elementele jocului în afară de InterpretWebcam. Prin aceasta mă refer la faptul că *Time.timeScale* este setat la 0 cât timp se asteaptă inputul jucătorului și înapoi la 1 după. Această abordare este posibilă datorită faptului că InterpretWebcap analizează trecerea timpului pe baza *Time.realtimeSinceStartup*, iar toate celelalte elemente ale jocului folosesc *Time.time*, *Time.deltaTime* și *WaitForSeconds()*, elemente afectate de *Time.timeScale*.

În al doilea rând, camera se îndepărtează pentru a permite jucătorului să vadă mai departe, lucru menit să îl ajute să vadă mai clar inamicii. Un efect secundar al acestei funcționalități este că jucătorul poate alege să utilizeze abilitatea doar pentru a vedea mai departe, posibilitate care nu are efecte adverse în ceea ce privește funcționalitatea jocului. Cu toate acestea, resursele sunt intentionat destul de limitate, fapt care poate descuraja acest comportament.

În ultimul rând se pornește o cortină care așteaptă următorul click stânga al jucătorului. În momentul detectării acestuia, se aruncă un *Raycast* din punctul apăsat și se verifică dacă primul *collider* atins este al unui inamic, mai exact al unui obiect cu tag *Enemy*, caz în care se activează scriptul de afectare al inamicului respectiv. În caz contrar, nu se întamplă nimic, dar abilitatea tot este considerată ca fiind efectuată și încheiată. După acestea jocul revine la normal și este consumat 1 MP.



Figura 3.34: Ability with menu

Trebuie menționată și metoda de utilizare a abilității când nu se folosește camera. În momentul activării abilității se activează și un meniu auxiliar cu 8 butoane, unul pentru fiecare din cele 8 emoții pe care le poate detecta modelul. Se așteapă un click stânga pe unul din aceste butoane, de data aceasta așteptarea având loc până este apăsat un buton, spre deosebire de abordarea anterioară în care era doar verificată validitatea primului click. Aceste butoane schimbă emoția din InterpretWebcam cu

cea enunțată pe buton. După setarea astfel a emoției abilitatea decurge ca în versiunea fără meniu.

Există scripturi de afectare ale inamicilor ca urmare a abilității. Fiecare tip de inamic are propriul script, dar toate urmează convenția de nume AffectType. Fiecare inamic are scriptul corespunzător ca o componentă initial inactivă. Ea se activează doar când inamicul este selectat de abilitate și se dezactivează după o singură rulare. În el este realizată o simplă asociere: în funcție de emoția din InterpretWebcam se activează variabila bool asociată unei anumite stări posibile ale acelui tip de inamic.

Revenind la InterpretWebcam, un ultim aspect nemenționat este faptul că la încercarea citirii din fișier, există două situații în care conținutul nu este cel așteptat. În primul rând, dacă nu există cameră, conținutul fișierului nu este în format json. În această situație este suprascrisă valoarea *force*, de forțare a nefolosirii camerei, din PlayerPrefs. Cealaltă situație este atunci când nu este detectat nici un jucător în fața camerei, caz în care json-ul are valori nule. În această situație se setează un flag menit să oprească temporar partea de actualizare a variabilelor publice și a istoricului, astfel încât acestea să nu fie actualizate cu valori nule. Dacă jucătorul continuă să nu fie detectat pentru 2 secunde, atunci Time.Timescale e transformat în 0 și este setată o variabilă în PlayerPrefs pentru comunicarea cu script-urile abilității, singurul altul care poate opri timpul în joc. Pentru a nu cauza probleme de sincronizare, fiecare din cele 2 scripturi repornește timpul doar dacă celălalt nu impune continuarea opririi, lucru indicat de variabile în PlayerPrefs. În general, însă, jocul repornește când un jucător este iar detectat.

Jocul are 3 zone și 5 nivele, fiecare cu particularitățile ei. Zonele sunt denumite după culori. Inițial doream să realizez câte o zonă pentru culorile primare additive și subtractive ale luminii, anume roșu, verde și albastru, respectiv cyan, magenta și galben. În final am ales să mă limitez la primul set de culori. Încă o intenție inițială a fost ca abilitatea să aibă cu totul alte efecte în fiecare zonă, afectând comportamentul inamicilor doar în prima. Cu toate acestea, am considerat că această versiune a abilității, cea deja explicitată, aducea prea multă valoare jocului pentru a fi cu totul înlocuită.

În ceea ce privește resursele folosite, numite assets, cum ar fi cele vizuale pentru protagonist, inamici, platforme, fundal, poțiuni și decorări și resursele necesare creării de animații, cât și cele audio pentru sunetele asociate unor evenimente și muzica de

fundal, au fost cu toate preluate de pe Unity Asset Store⁷, itch.io⁸ și Pixabay⁹.

Singura modificare realizată de către mine a fost crearea unei a doua versiuni a protagonistului, în Photoshop, întrucât cea găsită online¹⁰ avea doar versiune feminină. Am realizat prelucrarea tilemaps și a character sheets pentru a putea fi plasate în nivel, respectiv utilizate în animații, pe care în mod predominant le-am reconstruit eu. Un alt aspect ce trebuie discutat este faptul că modelul pentru protagonist nu era pe bază de sprites, ca toate celelalte, fiind în schimb construit, cu schelet, în unity. Avea un demo funcțional cu două scripturi atașate pentru funcționalitățile de bază și demonstrarea animațiilor. Am decis să nu încep prelucrarea acestui asset de la zero, în schimb modificând scriptul principal pentru a permite toate funcționalitățile noi, dar și pentru a îmbunătății funcționalitățile de bază.

În ceea ce privește controlarea personajului principal de către jucător, scriptul principal *SimplePlayerController* are o serie de funcții care se apelează în funcția Update, deci la fiecare frame. În primul rând am ajustat viteza de mișcare și puterea săriturilor. De asemenea, am adăugat abilitatea de alergare, *1.5 mai rapidă decât mersul, prin ținerea apăsată a tastei LeftShift. Mișcarea propriu-zisă se realizează prin formula *transform.position += moveVelocity * movePower * Time.deltaTime;*, unde *moveVelocity* este vectorul direcție corespunzător tastei apăsate și *movePower* este viteza setată de mișcare. În funcția de realizare a mișcării sunt setate și valorile corespunzătoare în animator și sunt apelate sunetele asociate mersului, dacă este cazul. Pentru mișcările direcționale sunt acceptate și săgețile și wasd de la tastatură.

Atâtădată, în contextul acestui proiect, se referă la abilitatea specială, a cărei animație asociată a fost inițial numită Attack. Dacă abilitatea nu este activă la momentul apăsării uneia din tastele asociate, Z și E, și jucătorul are cel puțin 1 MP, abilitatea este marcată ca fiind activă, se declanșează sunetul caracteristic ei și se pornește scriptul abilității, descris mai sus.

Coliziunile sunt detectate cu *OnCollisionEnter2D*. Dacă jucătorul atinge un inamic, jucătorul e rănit și pierde 1 HP. Ce nu am discutat la explicarea acestei resurse a fost faptul că jucătorul este împins ușor în spate, relativ la orientarea sa, prin formula *rb.AddForce(newVector2(-5f, 1f), ForceMode2D.Impulse);*, unde rb reprezintă componenta Rigidbody2D a personajului. Când se ajunge la 0 HP personajul este con-

⁷Unity Asset Store

⁸Itch.io asset store

⁹Pixabay

¹⁰Player character asset

siderat mort, se schimbă animația și sunetul, după care se așteaptă 2 secunde înaintea reîncarcării nivelului. Alte coliziuni verificate sunt cele cu poțiunile pentru HP și MP, care au tagurile *Heal*, respectiv *Mana*. De asemenea, dacă jucătorul lovește un tavan se impune un scurt delay de 0.5 secunde până poate sări din nou.

Funcția de sărit a prezentat multe dificultăți. Nu toate platformele sau obiectele de pe care doream ca jucătorul să poate sări aveau același tag. Spre exemplu, uneori jucătorul ajungea deasupra unui perete, cu tag *Wall*, caz în care doream să aibă în continuare libertate de mișcare, dar verificarea unui simplu collision cu *Wall* ar fi rezultat în abilitatea jucătorului de a sări când atinge peretele din orice poziție, analog altor tipuri de obiecte. Prin urmare, detectarea validității săriturii a fost efectuată cu o serie de verificări:

```
(Input.GetKeyDown(KeyCode.UpArrow) || Input.GetKeyDown(KeyCode.W))  
&& !anim.GetBool("isJump") && Time.time - setDelayTime > 0.5f &&  
(!falling || Time.time - startFalling <= 0.3f) && rb.velocity.y <= 0.1f.
```

Apăsarea unei taste corespunzătoare, dacă nu este deja în săritură, dacă nu a lovit recent un tavan, dacă nu e în cădere de prea mult timp și dacă poziția sa pe y este relativ stabilă. Ultimele verificări indică faptul că jucătorul a stat recent pe un obiect. Am considerat că jucătorul ar trebui să se poate salva dacă abia a căzut de pe o platformă sau dacă, în încercarea de a se deplasa de pe o platformă pe alta, a sărit puțin prea târziu, mai exact cu maxim 0.3 secunde. Această practică este populară în jocurile de tip platformă și este cunoscută sub denumirea de *Coyotetime*. Metoda de verificare a finalității unei sărituri a fost tot prin *rb.velocity.y*, mai exact am verificat dacă aceasta rămâne între 0.01f și -0.01f pentru cel puțin 0.02 secunde, timp suficient pentru a evita eventualitatea velocității 0 în vârful săriturii. O altă modificare față de versiunea cu care a venit asset-ul este că am implementat abilitatea de a opri săritura prin ridicarea tastei asociate, cu alte cuvinte abilitatea de înălțime și durată variabilă a săriturilor pe baza lungimii apăsării tastei. Acest lucru l-am realizat prin setarea *rb.velocity.y* la 0, impunând virtual atingerea vârfului săriturii, în momentul ridicării tastei.

Am adăugat abilitatea de a mișca temporar în jos camera, prin afectarea offset-ului, la apăsarea tastei Săgeată Jos sau S. De asemenea, am considerat necesară o modalitate rapidă și simplă de a reîncerca un nivel, prin apăsarea tastei Escape, pentru momentele în care jucătorul nu a manevrat corespunzător resursele oferite. În ultimul rând, la deschiderea jocului, fie prin începerea unui joc nou, fie prin continuarea celui anterior, pe ecran apare un meniu minimalist care explică acțiunile pe care le poate lua

jucătorul și ce taste poate folosi. Acest meniu poate fi închis sau deschis oricând, prin apăsarea tastei Tab.

Urmează descrierea zonelor, aşa cum apar în jocul final.



Figura 3.35: Red Area

Prima zonă este desemnată cea roșie. Aspectul este de tip medieval, acțiunea având loc în jurul unui castel. Fundalul și platformele sunt luate de pe Unity Asset Store¹¹. Acest asset vine cu 5 imagini diferite pentru fundal, permitând implementarea unui efect de paralaxă între straturi. În cod, efectul se bazează pe distanța dintre poziția Z a jucătorului și cea a părții specifice de fundal. Fiecare din cele 5 layere are poziții Z ușor diferite, între 0 și -1, semnificând cât de apropiate sau depărtate sunt față de jucător, în comparație cu celelalte elemente de fundal. Pe baza acestei valori am calculat factorul de paralaxă, formula fiind:

```
parallaxFactor => -Mathf.Abs(distanceFromPlayer) / 10;
```

Un alt factor care influențează efectul de paralaxă este mișcarea camerei, mai exact distanța dintre poziția initială și cea actuală. Astfel, în Update, calculez noua poziție a fiecărui element din fundal cu formula:

```
newPos = startingPos + camMoveSinceStart * parallaxFactor
```

Formula exactă pentru factorul de paralaxă și poziția fiecărui layer a fost determinată prin testare.

Tipul de inamic găsit în aceste nivele este dragonul¹². Acțiunile pe care le poate lua sunt stat pe loc, mers și atacat, mai exact scuipat foc la distanță mică. Pot merge și ataca în același timp. Aceste stări sunt controlate de scriptul principal al fiecărui inamic, care în toate cazurile urmează convenția de nume TypeMovement. Prin urmare, în *DragonMovement* se găsesc două proprietăți publice bool *IsMoving* și *IsAttacking*.

¹¹Red area assets

¹²Dragon asset

Valoarea lor adevărată este stocată în câte o variabilă privată definită cu [SerializeField] pentru a fi editată cu ușurință în timpul dezvoltării, dar a păstra simultan încapsularea. Această abordare oferă flexibilitate în customizarea acțiunilor ce trebuie luate la fiecare schimbare a uneia din variabile, dar și claritate în citirea codului, acțiunile necesare fiind astfel grupate într-un mod ușor de înțeles și accesat. Ambele variabile au mai multe verificări și actualizări ce trebuie efectuate la fiecare setare a valorii. În cazul mersului, trebuie actualizată viteza de mișcare, animația și sunetul. În ceea ce privește atacul, pe lângă animație și sunet, în momentul în care dragonul începe să scuipe foc trebuie realizată o verificare cu potențiale triggers. Mai exact, dacă *collider* ul dragonului se suprapune cu *trigger* ul tortelor plasate de-a lungul unor nivele și este vizual orientat cu fața spre tortă, componenta de lumină, *Light2D*, a acestora este activată, având ca efect aparenta aprindere a tortelor. Tot în acest script se realizează și rotirea dragonului în momentul atingerii unui perete și stabilirea direcției de mers pe baza orientării. Acest lucru este realizat prin schimbarea valorii *gameObject.transform.localScale.x* între 1 pentru dreapta, orientarea inițială a asset ului, și -1 pentru stânga. Mișcarea modelului este realizată, similar săriturilor jucătorului, prin afectarea *rb.velocity*, setând valoarea x a vectorului la *walkSpeed * walkDirectionVector.x*. Determinarea atingerii unui perete este realizată în alt script atașat tuturor inamicilor, numit *TouchingDirections*, care conține 3 proprietăți: *IsOnWall*, *IsGrounded* și *IsOnCeiling*, folosite în diverse verificări și setate prin *OnCollisionEnter2D* și verificarea tagurilor obiectelor atinse.

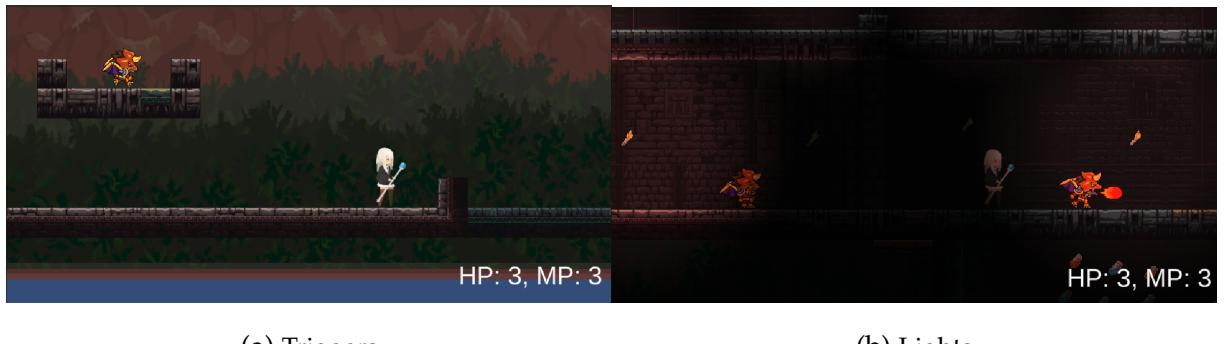
În ceea ce privește efectele abilității asupra lor, emoțiile “neutral” și “sad” opresc mersul, “happy” și “contempt” îi fac să meargă și restul îi fac să atace.

Mecanica principală a zonei rosii este aceea de *Triggers*. Mai exact, nivelele au zone de podea sau perete de altă culoare decât părtile înconjurătoare, pentru a putea fi identificate de către jucător. De exemplu, ele sunt de obicei albastru cu galben în loc de roșu sau albastru fără galben. Atunci când sunt atinse de jucator sau de un dragon activează sau dezactivează o altă secțiune de podea sau perete din nivel, denumită *Target*. De menționat este faptul că, deși unele triggers pot fi montate pe pereti, acestea nu sunt considerate pereti și, drept consecință, inamicii nu își vor schimba direcția la atingerea lor. Pentru funcționalitatea acestei mecanici am folosit 2 scripturi, unul pentru *Trigger* și unul care, dacă este atașat drept componentă unui obiect, permite acestora să declanșeze *Trigger* ele.

Primul script este numit *TriggerTarget* și primește ca variabilă publică, setată în editor, componenta *Target* pe care o poate controla. De asemenea ține minte când a

fost folosit ultima dată, cât și un cooldown, setat la 1 secundă. Ultima variabilă, *active*, este menită să indice dacă este sau nu deja cineva în coliziune cu *Trigger* ul respectiv. Oferă, de asemenea, o funcție publică de toggle a obiectului *Target*.

Al doilea script, cel folosit drept componentă pentru jucător și inamici, verifică intrarea și ieșirea din coliziune cu o componentă cu tag *Trigger*. La intrare, dacă nu este deja cineva în coliziune cu obiectul și acesta nu este în cooldown, adică a trecut cel puțin o secundă de la ultima utilizare, se setează că este în utilizare, se actualizează timpul ultimei utilizări și se apelează funcția de toggle. La ieșire se eliberează variabila *active*.



Drept mecanică auxiliară am utilizat luminile, ultimele fiind întunecate și necesitând utilizarea atacurilor dragonilor pentru a aprinde torte. O particularitate a luminilor 2D în Unity este faptul că acestea nu verifică coliziunea luminii cu potențiali pereți. Prin urmare, lumina pare să treacă în mod irealist prin anumite obiecte din scenă. Din păcate nu am găsit o alternativă sau soluție. De asemenea, pentru ca luminile să funcționeze în Unity este necesară setarea unui *UniversalRenderPipeline* în proprietățile proiectului, în submeniul *Graphics*. Aceasta poate fi creat simplu din meniul *Create*, ca toate celelalte obiecte oferite de Unity, dar necesită adăugarea pașchetului *UniversalRP* la proiect. Pe lângă aceste setări, singura problemă momentan nediscutată este natura aprinderii luminilor în *DragonMovement*. În acel script este verificată prezența unei torte în apropierea dragonului în momentul în care acesta începe să atace. Problema cu această abordare este faptul că uneori dragonul începe să atace în altă locație, după care se mișcă, fără a opri atacul, în zona tortei. Apar, de asemenea, potențiale probleme la încărcarea nivelului în cazul dragonilor care sunt initializați să atace în apropierea unei torte. Pentru a combate aceste situații am creat un script *LightToggle*, atașat tuturor tortelor. În funcția Start verifică toate *Collider2D* în contact cu componenta trigger a tortei. Dacă obiectul are tag *Enemy* și este valid, adică dra-

gonul este cu față spre tortă, atunci se verifică dacă dragonul este în atac, caz în care se activează lumina. Aceleasi verificări sunt efectuare în funcția *OnTriggerEnter2D*. Verificarea orientării dragonului relativ la tortă se realizează prin formula:

$$\begin{aligned} & Mathf.Sign(enemy.transform.localScale.x) != \\ & Mathf.Sign(enemy.transform.position.x - transform.position.x) \end{aligned}$$

Spre exemplu, dacă inamicul este orientat spre dreapta și se află la stânga tortei, atunci este valid.

Ultimul nivel are un timer, pentru a ridica dificultatea finalizării unei zone. Acesta testează nivelul de înțelegere al jucătorului asupra tuturor mecanicilor. Resursele sunt, de asemenea, mai limitate. Cu toate acestea, în testarea manuală realizată tot de către mine m-am asigurat ca nivelul să poate fi câștigat în jumătate din timpul alocat, dacă jucătorul știe exact ce are de făcut. Dificultatea ridicată nu ar trebui să cauzeze sentimente neplăcute, deoarece jucătorul are abilitatea de a reîncerca rapid și fără consecințe fiecare nivel. Elementul de UI al timer ului este prezent în toate scenele, adică toate nivelele, făcând parte din prefab, dar în majoritatea este dezactivat. Mai specific, timer ul are 5 minute și este afișat în format mm:ss utilizând formula *timerText.text = string.Format("0 : 00 : 1 : 00", minutes, seconds);*. Acest timer continuă să scadă și în timpul utilizării abilității, deoarece verificarea trecerii unei secunde se realizează prin *Time.realtimeSinceStartup - lastReadTime >= readInterval*, unde *readInterval* este de o secundă, semnalând faptul că textul din UI trebuie actualizat. Când timpul rămas este epuizat, adică atinge 0, nivelul este restartat automat, fără delay.

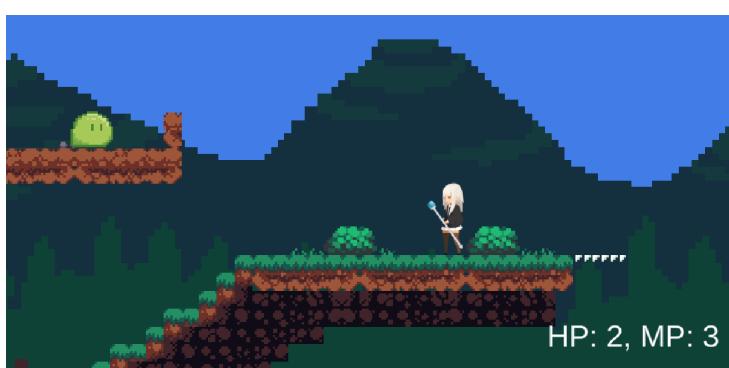


Figura 3.37: Green Area

A doua zonă este cea verde. De această dată nivelele au loc în natură, între munte. Asseturile pentru această zonă au fost luate tot de pe Unity Asset Store¹³. Si în acest pachet fundalul era separat în 5 imagini, aşa că efectul de parallaxă a rămas identic.

¹³Green area assets

Nivelele sunt populate de slimes. Deși pachetul de mai sus avea inclus acest tip de inamic, animațiile erau extrem de limitate. Prin urmare, pentru a putea confi și mai multe acțiuni acestor inamici am preluat un alt asset pentru ele, de data asta de pe itch.io¹⁴. Acestea pot sta pe loc, merge, sări sau ataca. În practică, ultima stare nu are nici o utilitate în jocul final, deși este funcțională. Pot merge și sări în același timp, combinatie care dublează viteza de mers pe durata săriturii. Activarea săriturilor nu rezultă în sărituri continue, starea fiind încheiată în momentul finalizării unei singure sărituri. În momentul săriturilor se modifică și collider-ul, care mimică aproximativ poziția din animație. Mai exact între unele frames ale animației este apelată o funcție care incrementează, respectiv decrementează offset-ul collider-ului. Acest lucru este realizat prin adăugare de events în animație, care apelează funcții regăsite în *SlimeMovement*. În animația de 11 frames, primele 2 nu generează acțiuni speciale, următoarele 4 cresc offsetul cu 0.2f, cât și poziția obiectului cu 1.2f, următoarele 4 decrementează valorile înapoi la cele inițiale, iar ultimul frame declanșează setarea proprietății *IsJumping* la false. Similar dragilor, *SlimeMovement* oferă 3 proprietăți: *IsMoving* și *IsAttacking* similar cazului anterior, dar fără afectarea luminilor, și *IsJumping*, care, pe lângă animație și sunet, crește temporar viteza. Încă un aspect similar cu dragonii este schimbarea direcției la atingerea unui perete, realizată prin aceeași modalitate. Am considerat crucială schimbarea poziției obiectului și collider-ului la săritură, pentru a oferi inamicilor mobilitate ridicată și intuitivă pentru jucător, elemente importante în rezolvarea nivelelor.

Pentru abilitatea jucătorului, sentimentele “neutral” și “sad” opresc orice acțiune, “happy” și “contempt” activează mersul, “fear” și “surprise” săritul și restul atacatul.



(a) Player Moving Platform

(b) Enemy Moving Platform

În această zonă există platforme care se mișcă atunci când sunt atinse de jucător. Ele sunt vizual distincte de restul platformelor, fiind albe, aproape ca niște triunghiuri

¹⁴Slime asset

lipite între ele, spre deosebire de cele cu aspect de pământ, iarba și pietriș care alcătuiesc majoritatea fiecărui nivel. Poziția jucătorului este mutată o dată cu platforma. Mișcarea platformei și efectul ei asupra jucătorului se oprește fie în momentul în care se ajunge la destinație, fie atunci când jucătorul pierde contactul cu platforma pentru puțin timp. Următoare atingere continuă mișcarea spre destinație. Dacă platforma ajunge la destinație, atunci următoarea atingere schimbă destinația înapoi la pozitia inițială și începe iar mișcarea, dar în direcția opusă.

La mijlocul zonei sunt introduse platforme similare, dar care pot fi mișcate doar de slimes. Vizual diferența este că textura platformei este rotită pe orizontală: pentru jucător partea plată este sus și cea ascuțită jos, pentru slimes partea ascuțita este sus și cea plată jos.

Logica acestor platforme se regăseste în scriptul atașat tuturor acestora, *MoveTile*. Acesta primește coordinatele destinației finale și tagul obiectelor care îl pot afecta, adică *Player* sau *Enemy*. *OnCollisionEnter2D* verifică în primul rând dacă au trecut mai puțin de 0.2 secunde de la ultima ieșire din coliziune cu obiectului cu tagul căutat în timpul mersului, verificare realizată în *OnCollisionExit2D*. Aceasta este rezolvarea la un bug întâlnit în testare, în care obiectul de pe platformă pierdea uneori contactul cu platforma, rezultând în oprirea imediată a mișcării. Un alt pas în rezolvarea acestei probleme a fost realizarea unei corutine de oprire a mișcării. Aceasta este pornită la ieșirea condiționată din coliziune și dacă nu este deja în rulare o astfel de corutină. Așteaptă 0.3 secunde înainte să verifice dacă în acest timp obiectul a reintrat în contact cu platforma. Oprirea platformei are loc doar dacă nu se reia contactul în acele 0.3 secunde. În continuare în *OnCollisionEnter2D* se verifică tagul, dacă platforma este deja în mișcare și un mic delay de 0.3 secunde minim necesar de la ultima oprire. După trecerea de aceste verificări se începe corutina asociată mișcării. Am considerat că această operațiune trebuie plasată într-o corutină datorită faptului că trebuie să efectueze operații la fiecare frame, dar doar în anumite intervale de timp, determinate de coliziuni. În această corutină se setează variabila *isMoving*, folosită pentru continuarea buclei, asigurând abilitatea de a opri din cealaltă corutină. Pe baza poziției actuale și celei finale se calculează direcția de deplasare. Se actualizează poziția platformei, cât și a obiectului care a declansat-o, prin formula *direction * moveSpeed * Time.deltaTime*, până platforma este la mai puțin sau exact 0.5f distanță de destinație sau se decide oprirea mișcării din cealaltă corutină. Dacă mișcarea se oprește din cauza ajungerii la destinație, atunci coordonatele destinației se schimbă în poziția inițială.



Figura 3.39: Blue Area

Ultima zona este cea albastra. Mediul este modern, chiar futurist, preluat de pe itch.io¹⁵. Efectul de paralaxă funcționează și aici în mod identic.

Nivelele sunt ocupate de roboti¹⁶, care pot sta pe loc, merge și ataca de la distanță. Aceștia nu pot merge și ataca simultan și au în plus față de celelalte tipuri de inamici un delay între oprirea și reactivarea acțiunilor.

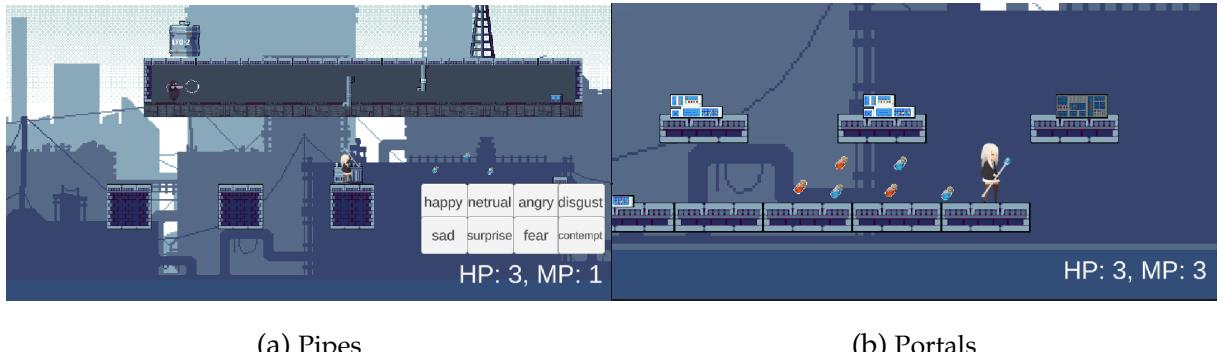
BotMovement controlează de această dată logica inamicilor. Toate precizările legate de schimbarea direcției la atingerea peretilor și structura proprietăților sunt valabile și aici. Proprietățile sunt IsSleeping, IsWaking, IsMoving și IsAttacking. Primele 2 reprezintă intrarea și ieșirea din starea de pauză. La începerea mișcării, IsMoving activează inițial iesirea din starea de pauză dacă botul nu era deja în mijlocul unei acțiuni, analog porning starea de intrare în pauză dacă botul nu ataca și a fost semnalată oprirea mersului. Si la schimbarea valorii IsAttacking sunt implementați acești pași auxiliari, dar setarea IsAttacking de la fals la adevarat activează o corutină ce simulează atacurile intermitente vizibile în animație. Această corutină așteaptă 0.8 secunde pentru tranziția animației, după care realizează verificarea obiectelor din față prin apelarea funcției *CheckForTargets*. În linie dreaptă de la *transform.position.x + 3*walkDirectionVector.x* la *transform.position.x+28*walkDirectionVector.x* este verificat cu *Linecast* care este cel mai apropiat collider. Dacă acesta aparține jucătorului, jucătorul este considerat atacat.

Pentru a doua mecanică a zonei am decis plasarea în scenă a unor țevi ce pot fi distruse de atacurile botilor. Prin urmare, funcția *CheckForTargets* verifică și dacă obiectul cel mai apropiat are tag *Breakable*, caz în care dezactivează obiectul, reprezentativ țevei, în scenă. Trebuie specificat și faptul că în procesul de schimbare a direcției

¹⁵Blue area assets

¹⁶Bot asset

inamicilor în momentul atingerii unui perete, aceste țevi sunt considerate pereti.



Revenind la prima mecanică a zonei, aceasta constă în prezența mai multor tipuri de portale în cadrul nivelerelor. Ele pot fi de 4 tipuri: pentru jucător sau inamic, care necesită sau nu activare. Fiecare din cele 4 variante arată ușor diferit. Cu alte cuvinte, unele pot teleporta doar jucătorul, altele doar botii, lucru decis în cod pe baza tag ului oferit în editor. În ceea ce privește activarea, este vorba despre necesitatea portalelor de a avea o sursă de curent. Unele au sursă de curent implicită și sunt active de la începerea nivelului, lucru indicat de culoarea lor mai deschisă. Atunci când un portal este inactiv, este mai închis la culoare și jucătorul trebuie să convingă un robot să atace o baterie asociată portalului, asemeni trigger urilor din prima zonă, pentru a activa portalul. Portalele funcționează doar când au curent. Destinația portalelor este fixă, oferită în editor, dar jucătorul o poate afla doar prin încercări.

Scriptul asocial portalelor, pentru realizarea repoziționării și verificărilor, este *TeleportTile*. Sursele de curent asociate portalelor au un alt script, *PowerTP*, în care este verificat în *Update* natura tuturor collider elor în contact cu cel al sursei, care este setat drept trigger. Când unul din aceste collidere aparține unui bot care atacă și are față spre sursă, portalul asociat este activat.

În ceea ce privește abilitatea jucătorului, sentimentele "neutral" și "sad" opresc atacurile, "happy" și "contempt" schimbă acțiune în mers, "fear" și "surprise" opresc mersul și restul schimbă acțiunea în atacat.

Au rămas câteva aspecte generale pe care nu le-am putut include până acum în acest subcapitol.

În ceea ce privește organizarea obiectelor în scenă, mai exact în Ierarhie, majoritatea obiectelor sunt grupate. Individuale se regăsesc modelele celor 2 jucători, Save-Manager, PythonComm care conține InterpretWebcam și scriptul de setare a protagonistului vizibil, camera 2D default și EventSystem necesar elementelor UI. Alături de

camera Cinemachine sunt grupate efectele sonore legate de jucător. În Background se regăsesc cele 5 părți ale fundalului. În Grid se află toate platformele, cum ar fi Ground, Wall, Ceiling, BG, dar și cele speciale cum ar fi Trigger, Target, Moving, Breakable, TP și Power. În BG sunt plasate mici obiecte decorative și perete care nu doream să aibă colliders. Toti inamicii se găsesc în Enemies, iar în final toate potiunile în Items.

În mare, fiecare nivel a fost gândit să servească unui scop specific. Nivelul 1 din fiecare zonă este unul introductiv. În zona roșie: al doilea nivel testează înțelegerea jucătorului asupra abilității, al treilea introduce tortele și crește dificultatea, al patrulea necesită folosirea intensă a tortelor, iar în ultimul nu este clar unde e finalul și cum se ajunge la el. În zona verde, al doilea nivel introduce platformele pentru inamici, al triilea are un singur slime ce trebuie adus până la finalul nivelului, al patrulea are multe sărituri ce trebuie executate cu precizie de slimes, iar ultimul necesită folosirea inamicilor pentru a construi calea spre final. În zona albastră, nivelul 2 îi arată jucătorului cum se poate feri de atacuri, nivelul 3 introduce abilitatea de spargere a țevilor, 4 are multe portale în zona de început, nu toate dintre care progresează nivelul, iar 5 necesită utilizarea abilității mai des ca orice alt nivel.

Am dorit ca inamicii să nu poată interacționa între ei. Acest lucru l-am realizat prin eliminarea coliziunilor dintre ei. Operația este efectuată într-un script atașat tuturor inamicilor de orice tip, numit *ManageEnemyCollisions*. În funcția Start se preiau toate obiectele cu tag *Enemy* din scenă, după care apelez *Physics2D.IgnoreCollision(entityCollider, enemyCollider)*.

Jocul nu folosește camera standard 2D. În schimb, este utilizată *CinemachineVirtualCamera* din pachetul *Cinemachine* din Unity. Aceasta oferă posibilitatea setării unui aşa numit *DeadZone*, în care jucătorul se poate mișca aproape de centrul ecranului fără a mișca poziția camerei, cât și un *SoftZone*, în care camera se mișcă mai încet decât dacă jucătorul ar fi exact pe marginea zonei vizibile. Când am menționat îndepărțarea camerei în momentul utilizării abilității mă refeream mai specific la dublarea *OrthoSize*, de la 10 la 20, în cadrul acestei camere.

O specificație legată de sunetele folosite este că am dorit ca acestea să aibă calitate stereo. Cu alte cuvinte, am considerat important ca jucătorul să poată determina sursa unui sunet, în funcție de direcția din care se audă. În plus, am dorit ca sunetele să poată fi auzite doar în apropierea sursei. Pentru a atinge aceste proprietăți am modificat valoarea *SpatialBlend* a surselor de sunet la 1, valoare tipică unui mediu 3D. Sunetele astfel setate funcționează și în jocuri 2D, cu specificația că, de exemplu, dacă sursa se

află într-o cameră închisă, jucătorul nu poate auda deloc sunetul din afara camerei.

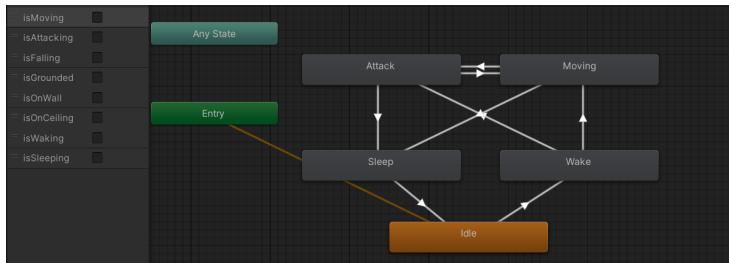


Figura 3.41: Bot Animator

În ceea ce privește animațiile, acestea sunt controlate, cum este ușual, printr-o componentă Animator asociată fiecărui inamic și jucătorului. Animațiile pentru jucător nu le-am modificat din versiunea inițială a asset ului. În cazul inamicilor, aceștia au venit cu o serie de imagini ce puteau fi legate într-o animație, dar animația în sine nu era încă realizată. Particularitățile legate de animații, cum ar fi viteza, au fost determinate prin testare. În animator se regăsesc o serie de variabile bool care condiționează tranzițiile între stări. Acestea reprezintă dacă obiectul atinge podeaua, tavanul sau un perete și dacă este în mijlocul unei acțiuni. Acțiunile exacte sunt cele discutate în secțiunile respective, mai exact paragrafele despre fiecare tip de inamic. În mare parte sunt posibile tranziții de la oricare la oricare stare și, implicit, animație. Singura excepție o fac botii din ultima zonă, care au animații pentru culcat și trezit. După ce începe culcarea este impusă trecerea prin Idle și prin trezire înaintea realizării unei alte acțiuni, anume mers sau atacat. De asemenea, oprirea mersului sau atacatului poate rezulta ori în începerea celeilalte acțiuni, ori în culcat.

Trecerea de la un nivel la următorul se realizează printr-un obiect din Grid, numit Finish. Acesta conține un script care doar încarcă scena a cărei denumire îi este dată în editor în momentul coliziunii cu jucătorul, determinat pe bază de Tag. Excepție face ultimul nivel din ultima zonă, unde acest obiect conține încă un script ce setează *PlayerPrefs.SetInt("Win", 1)*. Deoarece nu există SetBool și GetBool în PlayerPrefs, utilizez int cu valorile 1 pentru true și 0 pentru false pentru aceasta dar și pentru variabila folosită pentru forțarea nefolosirii camerei web. Valoarea Win din PlayerPrefs este verificată în momentul încărcării meniului principal, scena încărcată în urma ultimului nivel, de către componența *CheckWin* a obiectului *MainCanvas*. Există un submeniu, în mod obișnuit dezactivat, care se activează în cazul prezenței acestei valori și care conține felicitările câștigate în urma finalizării jocului.

Concluzii

Sunt de părere că inovațiile în IA pot aduce îmburătățiri și inovații extraordinare în multe domenii, inclusiv în cel al jocurilor. Ele nu pot doar eficientiza aspectele monoton din viața noastră, lasându-ne cu mai mult timp pentru ce vrem cu adevărat să facem, ci pot și deschide oportunități către lucruri anterior imposibile. Acest nivel de interacțiune cu jucătorul este un astfel de exemplu: înainte de IA, nu am fi avut nici o modalitate de a detecta automat sentimentele jucătorului și de a utiliza această informație în timpul jocului.

Pe viitor sunt sigură că vor apărea din ce în ce mai multe jocuri care folosesc IA de clasificare în gameplay pentru experiențe cât mai interactive și pentru a adapta lumea virtuală cât mai natural la ce face jucătorul, eliminând nevoie a sute de meniuri și butoane, sau de algoritmi și formule grele de dezvoltat manual. Desigur, varianta adusă de mine este relativ simplistă, dar aduce în discuție cum altfel am putea folosi camera web pentru a controla unele aspecte ale unei aplicații.

Se mai pot folosi lucruri legate de aspect pentru crearea instantă a unui personaj, mișcări ale capului, mâinilor sau ale întreg corpului pe baza unei singure camere, nu a mai multor senzori. Partea de detecție a sentimentelor ar fi cu siguranță extrem de utilă pentru experiențe narative, permitând naratorului și personajelor să se adapteze la reacțiile jucătorului în timp real.

Domeniul este nou și abia aștept să văd ce utilizări ingenioase apar în viitor, în special în interacțiunea cu utilizatorul, pentru a-i crea o experiență cât mai interesantă, plăcută și imersivă.

Bibliografie

- Eran Eidinger, Roei Enbar, and Tal Hassner, *Age and Gender Estimation of Unfiltered Faces*, Transactions on Information Forensics and Security (IEEE-TIFS), special issue on Facial Biometrics in the Wild, Volume 9, Issue 12, pages 2170 - 2179, Dec. 2014
- Harry H. Jiang, Lauren Brown, Jessica Cheng, Mehtab Khan, Abhishek Gupta, Deja Workman, Alex Hanna, Johnathan Flowers, and Timnit Gebru, *AI Art and its Impact on Artists.*, In Proceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society (AIES '23). Association for Computing Machinery, New York, NY, USA, 363–374.
- Sigalas Markos *Artificial intelligence: dangers to privacy and democracy.*, 2021
- Kobe Millet, Florian Buehler, Guanzhong Du, Michail D. Kokkoris, *Defending humankind: Anthropocentric bias in the appreciation of AI art.*, Computers in Human Behavior, Volume 143, 2023, 107707, ISSN 0747-5632.
- World Economic Forum *Stanford University AI Index Report*, 2024
- Wharton School of the University of Pennsylvania *AI and Machine Creativity: How Artistic Production Is Changing*, 2024
- McKinsey Global Institute *Jobs lost, jobs gained: Workforce transitions in a time of automation*, 2017
- World Economic Forum *The Future of Jobs Report 2020*, 2020
- Authors Guild *Concerns about AI and the Future of Writing*, 2020
- Parlamentul European *Texte adoptate - Regulamentul privind inteligența artificială*, 2024

- Priyanka Ranade, Anupam Joshi, Tim Finin *1 Texte adoptate - Regulamentul privind inteligența artificială*, 2024
- Noémi Bontridder, Yves Poulet *The role of artificial intelligence in disinformation*, 2021, Cambridge University Press
- Ethan Fast, Eric Horvitz *Long-Term Trends in the Public Perception of Artificial Intelligence*, 2017
- Tensorflow setup guide
<https://www.tensorflow.org/install/pip#windows-native>
- Deep Learning Course : TensorFlow - Python Deep Learning Neural Network API
<https://deeplizard.com/course/tfcpailzrd>
- Deep Learning Course : Deep Learning Fundamentals
<https://deeplizard.com/course/dlcpailzrd>
- Tensorflow tutorials <https://www.tensorflow.org/tutorials>
- Unity asset store <https://assetstore.unity.com>
- Itch.io asset store <https://itch.io/game-assets/free>
- Pixabay copyright free music & sfx <https://pixabay.com>
- Unity 2D Light
<https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@7.1/manual/2d-index.html>
- <https://learn.unity.com/course/create-with-code>
- <https://learn.unity.com/project/creator-kit-rpg>
- Kaggle Datasets <https://www.kaggle.com>
- <https://www.pewresearch.org/short-reads/2023/11/21/what-the-data-says-about-americans-views-of-artificial-intelligence/>

- Gender and age dataset <https://talhassner.github.io/home/projects/Adience/Adience-data.html>
- Fer-2013 sample <https://www.kaggle.com/datasets/jonathanohex/face-expression-recognition-dataset/data>
- AffectNet sample <https://www.kaggle.com/datasets/noamsegal/affectnet-training-data>
- VSCode Jupyter extension <https://code.visualstudio.com/docs/datascience/jupyter-notebooks>
- <https://www.python.org/downloads/release/python-3917/>
- Conda environments <https://conda.io/projects/conda/en/latest/user-guide/install/index.html>
- Transfer learning with keras https://keras.io/guides/transfer_learning/
- ResNet50 <https://keras.io/api/applications/resnet/>
- CUDA for GPU optimization of tensorflow
<https://docs.nvidia.com/cuda/cuda-installation-guide-microsoft-windows/index.html>
- cv2 VideoCapture
https://docs.opencv.org/3.4/d8/dfe/classcv_1_1VideoCapture.html
- Using checkpoints in training
https://keras.io/api/callbacks/model_checkpoint/
- Reduce learning rate on plateau
https://keras.io/api/callbacks/reduce_lr_on_plateau/