UNIVERSITATEA BABEŞ-BOLYAI CLUJ-NAPOCA

FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ

SPECIALIZAREA INFORMATICA

Metode Inteligente de Rezolvare a Problemelor Reale

EMOINATOR

Raport

Borza Catalin

Mariei Andrei

Panaite Cristian

Parcas Rares

2021-2022

Conținut

Introducere

Succesul sau eșecul unei aplicații interactive este determinată de utilizabilitatea produsului. Una din componentele utilizabilitătii este satisfacția utilizatorului. Evaluarea satisfacției pentru utilizatorii adulți se realizează prin observație, interviuri post-interacțiune sau metode cantitative (chestionare) care însă nu pot fi aplicate copiilor preșcolari, care au capacitate limitată de autoanaliză și de comunicare, nu pot citi și nu pot scrie. Astfel, pe langă observație (care poate fi subiectivă, depinzând de modul de interpretare a reacțiilor preșcolarului de către expertul care realizează aplicația), se dorește o măsurare obiectivă a emoțiilor pe care le traiesc copiii în timpul interacțiunii. Pentru aceasta este nevoie de dezvoltarea unei aplicații care să permită identificarea stărilor emoționale ale unui preșcolar în timpul derulării unei activități.

Pentru o analiză cât mai completă, identificarea stărilor emoționale trebuie făcută într-o manieră cât mai obiectivă, bazată pe numere și calcule rezultate dintr-un proces de învățare elaborat, și în același timp, această identificare trebuie facută in timp real. Ca aceste două aspecte să fie îndeplinite, concepera analizei ar implica supravegherea constantă de către un specialist cu experientă, care să monitorizeze stările emoționale ale preșcolarilor, implicație care aduce cu ea costuri foarte mari atât financiare, cât si resurse umane. Aceste costuri și resurse pot fi reduse prin folosirea unei aplicații care să înglobeze un proces automat de monitorizare a stărilor emoționale prin utilizarea metodologiilor inteligenței artificiale.

Ceea ce își propune Emoinator este livrarea unei aplicații pe dispozitivele mobile care să identifice preșcolarul, să îi identifice stările emotionale din diferite surse (audio, video) care ne pot sugera care sunt emoțiile lui la un moment, iar apoi să salveze aceste date pentru a fi ușor de analizat de către profesorul care coordonează grupul de preșcolari. Pe baza analizei, persoana responsabilă poate decide care a fost impactul activității asupra copiilor și îl va ajuta în ceea ce priveste modul de organizare a viitoarelor activități pentru a maximiza impactul pozitiv si a face experința cât mai plăcută pentru participanți.

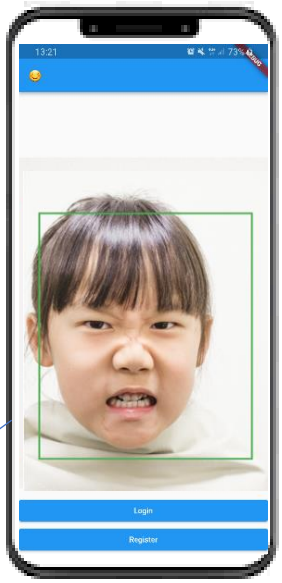
Problema științifică abordată

Pentru această aplicație se pot identifica 3 probleme: recunoașterea facială, identificarea emoțiilor din imagini și identificarea emoțiilor din sunete.

Recunoașterea facială este o tehnologie capabila să facă legătura între fața unei persoane și un cadru vizual sau o secvență video. (Wikipedia, Wikipedia 2021)

Recunoașterea emoțiilor este procesul de identificare a emoțiilor umane. Folosirea tehnologiei pentru a ajuta oamenii în recunoașterea emoțiilor este o arie în curs de cercetare. În general, tehnologia funcționează cel mai bine dacă folosește mai multe modalități în context. Până în prezent, s-au efectuat cele mai multe lucrări privind automatizarea recunoașterii expresiilor faciale din video, expresii vorbite din audio, expresii scrise din text. (Wikipedia, Emotion_recognition 2021)

Face recognition



Send results from image

Send result from sound

Real time data

Emotion recognition from image

Emotion recognition from sound

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 48, 48, 64)	640
batch_normalization (BatchNo	(None, 48, 48, 64)	256
activation (Activation)	(None, 48, 48, 64)	0
max_pooling2d (MaxPooling2D)	(None, 24, 24, 64)	0
dropout (Dropout)	(None, 24, 24, 64)	0
conv2d_1 (Conv2D)	(None, 24, 24, 128)	204928
batch_normalization_1 (Batch	(None, 24, 24, 128)	512
activation_1 (Activation)	(None, 24, 24, 128)	0
max_pooling2d_1 (MaxPooling2	(None, 12, 12, 128)	0
dropout_1 (Dropout)	(None, 12, 12, 128)	0
conv2d_2 (Conv2D)	(None, 12, 12, 512)	590336
batch_normalization_2 (Batch	(None, 12, 12, 512)	2048
activation_2 (Activation)	(None, 12, 12, 512)	0
max_pooling2d_2 (MaxPooling2	(None, 6, 6, 512)	0
dropout_2 (Dropout)	(None, 6, 6, 512)	0
conv2d_3 (Conv2D)	(None, 6, 6, 512)	2359808
batch_normalization_3 (Batch	(None, 6, 6, 512)	2048
activation_3 (Activation)	(None, 6, 6, 512)	0
max_pooling2d_3 (MaxPooling2	(None, 3, 3, 512)	0
dropout_3 (Dropout)	(None, 3, 3, 512)	0
flatten (Flatten)	(None, 4608)	0
dense (Dense)	(None, 256)	1179904
batch_normalization_4 (Batch	(None, 256)	1024
activation_4 (Activation)	(None, 256)	0
dropout_4 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 512)	131584
batch_normalization_5 (Batch	(None, 512)	2048
activation_5 (Activation)	(None, 512)	0
dropout_5 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 7)	3591

Total params: 4,478,727
Trainable params: 4,474,759
Non-trainable params: 3,968

Model: "sequential_10"

Layer (type)	Output Shape	Param #
dense_35 (Dense)	(None, 180, 256)	512
dense_36 (Dense)	(None, 180, 512)	131584
dropout_10 (Dropout)	(None, 180, 512)	0
dense_37 (Dense)	(None, 180, 512)	262656
dense_38 (Dense)	(None, 180, 256)	131328
flatten_8 (Flatten)	(None, 46080)	0
dense_39 (Dense)	(None, 4)	184324

Total params: 710,404
Trainable params: 710,404
Non-trainable params: 0

Metode existente de rezolvare a problemei

Recunoaștere facială

Pentru această problemă există modele și metode foarte performante.

Identificarea prin recunoaștere facială presupune:

- Detecția feței dintr-o imagine sursă, pentru care există mai multe arhitecturi:
    - "Cascade classifiers" (Paul Viola 2001): se iterează prin imagine folosind o fereastră glisantă, fiecare porțiune fiind comparată cu mai multe trăsături dreptunghiulare, organizate pe stadii care cresc în complexitate; dacă la un stadiu nu se trece testul, următoarele nu sunt evaluate.
        - Cod și modele deja antrenate există în biblioteca OpenCV
        - Fiind o metodă de detecție a obiectelor în imagini, se pot folosi modele pentru față și pentru ochi pentru a îndrepta imaginea, ceea ce ajută algoritmul de recunoaștere
        - Filmare cu camera web (640x480) + detecție live (cuda) ~ 20 fps
        - Probleme: unele poziții ale feței nu sunt detectate
    - MTCNN ("Multi-task cascaded convolutional networks") (Kaipeng, et al. n.d.)
        - P-Net ("Proposal network") generează zone dreptunghiulare din imagine care s-ar putea să conțină o față;
        - R-Net ("Refine network") elimină o mare parte dintre propunerile primei rețele, le combină pe cele suprapuse și le îmbunătățește pe cele existente prin regresie cu casetă de delimitare;
        - O-Net ("Output network") este similară cu R-Net, îmbunătățește rezultatele acesteia și găsește 5 puncte de interes pe fețele detectate (ochii, nasul, colțurile gurii)
            - Există diverse implementări, inclusiv pentru Tensorflow Lite, care poate fi rulat pe dispozitive mobile
            - Filmare (640x480) + detecție live (Tensorfow, cuda) ~ 2 fps
    - DLIB
        - Filmare (640x480) + detecție live (cuda) ~ 11 fps
- O metodă de a măsura similaritatea a 2 fețe
    - FaceNet (Florian , Dmitry and James n.d.), dlib creează un vector cu 128 elemente care identifică o față într-un spațiu vectorial, astfel că similaritatea a 2 fețe este decisă în funcție de distanța euclidiană dintre vectori. Metoda este fezabilă deoarece, într-o bază de date, se pot salva acești vectori în locul unor imagini cu fețe.
        - Filmare (640x480) + detecție live (160x240, din 2 în 2 cadre, cuda) + encoding ~ 30 fps

Recunoașterea emoțiilor din imagini

Înainte de pandemie, Ka Tim Chu, profesor și director adjunct al Colegiului True Light din Hong Kong, s-a uitat la fețele elevilor săi pentru a evalua modul în care aceștia au răspuns la munca de clasă. Acum, cu majoritatea lecțiilor sale online, tehnologia îl ajută pe Chu să observe stundenții prin intermediul camerei. O platformă de învățare bazată pe inteligență artificială monitorizează emoțiile elevilor săi în timp ce învață acasă.

Software-ul, 4 Little Trees, a fost creat de startup-ul Find Solution AI din Hong Kong. În timp ce utilizarea IA de recunoaștere a emoțiilor în școli și în alte setări a provocat îngrijorare, fondatorul Viola Lam spune că poate face clasa virtuală la fel de bună ca – sau mai bună decât – lucrul real.

Elevii lucrează la teste și teme pe platformă ca parte a curriculum-ului școlar. În timp ce studiază, IA măsoară punctele musculare de pe fețele lor prin intermediul camerei de pe computer sau tabletă și identifică emoții, inclusiv fericire, tristețe, furie, surpriză și frică. (Lam fără an)

Această problemă a recunoașterii emoțiilor din imagini este abordată și la nivelul adulților, unde cercetarea a facut mai multe încercări. Tabelul de mai jos ilusteaza arhitecturile folosite, datele pe care s-a antrenat și performanțele obținute.

Model	Dataset	Accuracy	Paper Link
Multi-task EfficientNet-B2	AffectNet	66.34	https://arxiv.org/pdf/2103.17107.pdf
DAN	AffectNet	65.69	https://arxiv.org/pdf/2109.07270v3.pdf
Distilled student	AffectNet	65.4	https://arxiv.org/pdf/2103.09154v2.pdf
Ensemble ResMaskingNet with 6 other CNNs	FER2013	76.82	https://arxiv.org/pdf/1307.0414v1.pdf
LHC-Net	FER2013	74.42	https://arxiv.org/pdf/2111.07224v2.pdf
Residual Masking Network	FER2013	74.14	https://arxiv.org/pdf/1307.0414v1.pdf
FER-VT	FERPlus	90.04	https://www.sciencedirect.com/science/article/abs/pii/S0020025521008495
LResNet50E-IR	FERPlus	89.257	https://arxiv.org/pdf/2012.13912v1.pdf
RAN (VGG-16)	FERPlus	89.16	https://arxiv.org/pdf/1905.04075v2.pdf

Recunoașterea emoțiilor din semnale audio

O prima încercare de detecție a emoțiilor din semnale audio este descrisa în lucrarea MULTIMODAL SPEECH EMOTION RECOGNITION USING AUDIO AND TEXT, care folosește pe lângă semnalele audio și text. Ca si arhitectură neuronală, paperul descrie folosirea unui RNN cu care, pe dataset-ul IEMOCAP, se obține o acuratețe de 71.8%.

Alte paper-uri care încearcă identificarea emoțiilor din semnale audio sunt prezentate în tabelul de mai jos:

Model	Dataset	Accuracy	Paper Link
ERANN-0-4	RAVDESS	74.8	https://arxiv.org/pdf/2106.01621v4.pdf

Pentru detecții ale emoțiilor din semnale audio la copii, paperul EmoReact: A Multimodal Approach and Dataset for Recognizing Emotional Responses in Children. Pentru cuantificarea performanței lucrării aceștia folosesc fbeta si AUC/ROC, cea mai mare valoare obținută doar din audio pentru fbeta fiind 0.69 si 0.61 pentru AUC/ROC.

Metode efectiv folosite pentru rezolvarea problemei

Recunoaștere facială

Datorită existenței unor performanțe foarte bune pe partea de recunoaștere facială, am adoptat una dintre metodologiile prezentate în capitolul anterior.

Versiunea implementată în cadrul aplicației este Opencv haar cascade.

Recunoașterea emoțiilor din imagini

Pentru această problemă de clasificare, Emoinator se folosește de un model convolutiv, prin care retunrează probabilitățile celor 7 emoții pe care le consideram (angry, disgust, fear, happy, neutral, sad, surprise).

Arhitectura folosită este una proprie, prezentată în detaliu în imaginea de mai jos, arhitectură dezvoltată folosind biblioteca keras.

Layer (type)	Output Shape	Param #
conv2d_8 (Conv2D)	(None, 48, 48, 64)	256
conv2d_9 (Conv2D)	(None, 48, 48, 64)	12352
batch_normalization_6 (Batch	(None, 48, 48, 64)	256
activation_7 (Activation)	(None, 48, 48, 64)	0
max_pooling2d_4 (MaxPooling2	(None, 24, 24, 64)	0
dropout_6 (Dropout)	(None, 24, 24, 64)	0
conv2d_10 (Conv2D)	(None, 24, 24, 128)	24704
conv2d_11 (Conv2D)	(None, 24, 24, 128)	49280
batch_normalization_7 (Batch	(None, 24, 24, 128)	512
activation_8 (Activation)	(None, 24, 24, 128)	0
max_pooling2d_5 (MaxPooling2	(None, 12, 12, 128)	0
dropout_7 (Dropout)	(None, 12, 12, 128)	0
conv2d_12 (Conv2D)	(None, 12, 12, 256)	98560
conv2d_13 (Conv2D)	(None, 12, 12, 256)	196864
batch_normalization_8 (Batch	(None, 12, 12, 256)	1024
activation_9 (Activation)	(None, 12, 12, 256)	0
max_pooling2d_6 (MaxPooling2	(None, 6, 6, 256)	0
dropout_8 (Dropout)	(None, 6, 6, 256)	0
conv2d_14 (Conv2D)	(None, 6, 6, 512)	393728
conv2d_15 (Conv2D)	(None, 6, 6, 512)	786944
batch_normalization_9 (Batch	(None, 6, 6, 512)	2048
activation_10 (Activation)	(None, 6, 6, 512)	0
max_pooling2d_7 (MaxPooling2	(None, 3, 3, 512)	0
dropout_9 (Dropout)	(None, 3, 3, 512)	0
flatten_1 (Flatten)	(None, 4608)	0
dense_3 (Dense)	(None, 512)	2359808
batch_normalization_10 (Batc	(None, 512)	2048
activation_11 (Activation)	(None, 512)	0
dropout_10 (Dropout)	(None, 512)	0
dense_4 (Dense)	(None, 256)	131328
batch_normalization_11 (Batc	(None, 256)	1024
activation_12 (Activation)	(None, 256)	0
dropout_11 (Dropout)	(None, 256)	0
dense_5 (Dense)	(None, 7)	1799
activation_13 (Activation)	(None, 7)	0

Total params: 4,062,535
Trainable params: 4,059,079
Non-trainable params: 3,456

Ca și input, rețeaua accepta imagini de dimensiune 48x48 și returnează un șir de 7 valori reale între 0 și 1 care reprezintă probabilitatea pentru fiecare tip de emoție.

Rețeaua a fost antrenată pe setul de date CAFE, care conține 1192 de imagini cu copii de diferite rase și în diferite emoții.




Optimizatorul folosit este Adam, loss-ul folosit este categorical_crossentropy și ca și metrici de evaluare am folosit fbeta și acuratețea, iar în timpul antrenării, după un anumit număr de epoci, learning_rate-ul a scăzut.

După antrenare și optimizare, acuratețea obținută este de 67.19%.

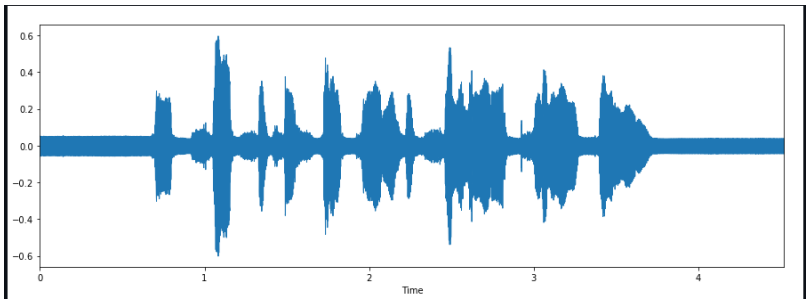
Recunoașterea emoțiilor din semnale audio

Pentru detectarea emoțiilor din semnale audio s-a folosit o rețea neuronală convolutivă.

Dataset-urile pe care s-a făcut antrenarea sunt The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) și SAVEE.

Ca și input pentru rețeaua neuronlă avem un array de 216 numere reale, ce reprezintă feature-uri extrase din fișiere de tip waveform.

Waveform Audio File Format (WAVE, sau WAV datorită extensiei numelui de fișier; pronunțat „wave”) este un standard de format de fișier audio, dezvoltat de IBM și Microsoft , pentru stocarea unui flux de biți audio pe computere. Este formatul principal utilizat pe sistemele Microsoft Windows pentru audio necomprimat. Codificarea obișnuită a fluxului de biți este formatul LPCM (liniar pulse-code modulation). (Wikipedia 2021).



Featurile sunt extrase folosind libraria LibROSA, care este una dintre librăriile folosite pentru analiza audio.

Arhitectura rețelei neuronale implementată folosind biblioteca keras și se poate observa în imaginea de mai jos.

```
Model: "sequential_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_40 (Conv1D)           (None, 216, 256)          1536
_________________________________________________________________
activation_50 (Activation)   (None, 216, 256)          0
_________________________________________________________________
conv1d_41 (Conv1D)           (None, 216, 128)          163968
_________________________________________________________________
activation_51 (Activation)   (None, 216, 128)          0
_________________________________________________________________
dropout_10 (Dropout)         (None, 216, 128)          0
_________________________________________________________________
max_pooling1d_10 (MaxPooling (None, 27, 128)           0
_________________________________________________________________
conv1d_42 (Conv1D)           (None, 27, 128)           82048
_________________________________________________________________
activation_52 (Activation)   (None, 27, 128)           0
_________________________________________________________________
conv1d_43 (Conv1D)           (None, 27, 128)           82048
_________________________________________________________________
activation_53 (Activation)   (None, 27, 128)           0
_________________________________________________________________
flatten_10 (Flatten)         (None, 3456)              0
_________________________________________________________________
dense_10 (Dense)             (None, 10)                34570
_________________________________________________________________
activation_54 (Activation)   (None, 10)                0
=================================================================
Total params: 364,170
Trainable params: 364,170
Non-trainable params: 0
```

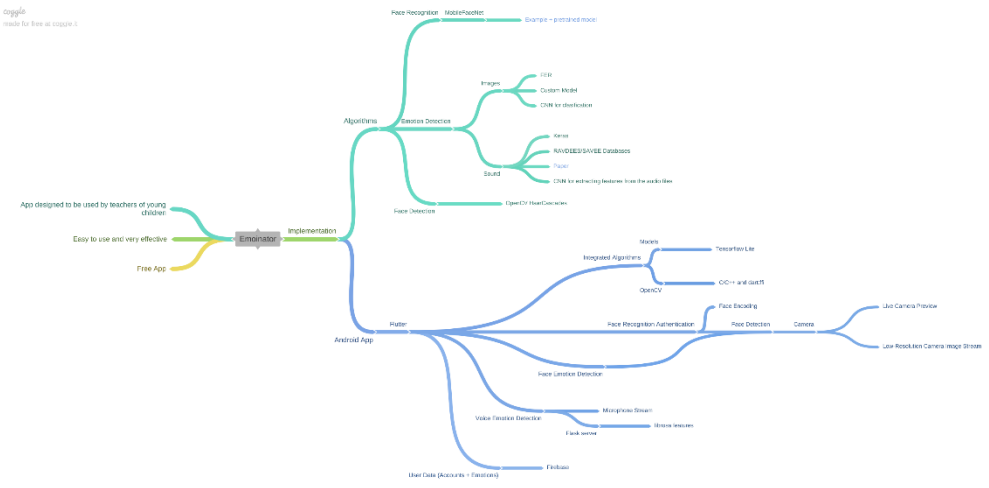
Rețeaua returneaza 10 numere reale, reprezentând probabilitatea pentru fiecare emoție (Calm, Fericit, Trist, Nervos, Infricosat) pentru 2 sexe (masculin/feminin).

Oprimizatorul folosit la compilarea modelului este RMSprop, cu un learning_rate de 0.00001, iar loss-ul este de tip categorical_crossentropy

Pentru analiza performanțelor, se folosește ca și metrică de evaluare acuratețea.

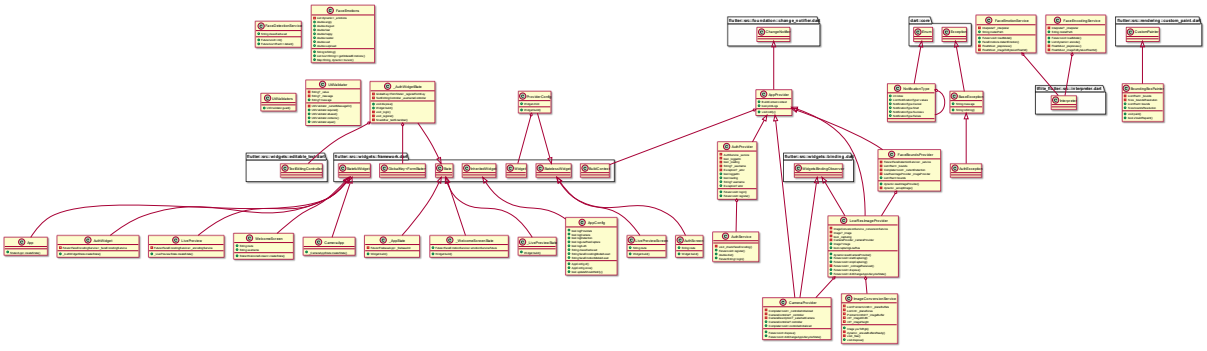
Integrarea algoritmilor inteligenți în aplicație

Ca și tehnologie de dezvoltare a aplicației mobile, Emoinator folosește Flutter/Dart si Firebase.



Mindmap-ul prezentat în imaginea de mai sus ilstrează funcționalitățile aplicației finale și în lucru.

De asemenea, o reprezentare vizuală foarte detaliată din punct de vedere arhitectural este prezentată în imaginea de mai jos:



Aplicația este compusă din 3 ecrane principale: Login, Register, HomePage.

În cadrul aplicației, este utilizat un stream de imagini de la camera frontală, deoarece detecția feței și a emoțiilor sunt procese continue, pentru a modifica interfața în funcție de modificări în poziția feței sau emoție și a trimite date către Firestore s-a folosit biblioteca provider.

Astfel, există:

- provider pentru controller-ul camerei (în cazul în care, în viitor, vom dori să schimbăm camera sursă/setarea de rezoluție);
- provider pentru un stream de imagini de rezoluție scăzută (valoarea exactă depinde de dispozitiv), care convertește, folosind cod C chemat din Dart, o imagine YUV_420 (formatul camerei în Android) în RGB
- provider pentru zonele de delimitare pentru fețele din imaginile de rezoluție scăzută, care folosește o funcție de detecție scrisă în C++/OpenCV, bazată pe clasificatori de tip Haar.

Codul de C/C++ este compilat sub formă biblioteci statice, iar apelurile din Dart sunt efectuate folosind dart:ffi.

Modelele tflite sunt încapsulate în servicii care expun câte o metodă pentru preprocesarea datelor de intrare, rularea propriu-zisă a modelului și formatarea datelor de ieșire.

Modelele de predicție au fost dezvoltate folosind keras, a fost nevoie de o metodă de a converti la un format pe care și dispozitivele mobile să îl suporte. Aici intervine folosirea bibliotecii tflite și folosirea modelelor convertite în aplicația mobilă (Deshmukh 2021).

Modelele tflite sunt încapsulate în servicii care expun câte o metodă pentru preprocesarea datelor de intrare, rularea propriu-zisă a modelului și formatarea datelor de ieșire.

Pentru predicții în timp real, calitatea imaginilor este scazută, renunțând astfel la calitatea input-urilor în favoarea performanțelor.

Deoarece modelul de detecție a emoțiilor din sunete este antrenat pe caracteristicile extrase folosind Librosa și neexistând o implementare echivalentă care să calculeze aceleași caracteristici în același format precum cel din Python, Emoinator externalizează acest task prin folosirea unui server Flask.

Rezultate experimentale obtinute

Recunoașterea emoțiilor din imagini

Procesul de antrenare și obținere a performanțelor actuale prezente pe Emoinator a început prin încercarea mai multor arhitecturi pe o bază de date mai mică.

Astfel, baza de date aleasă a fost https://www.kaggle.com/adityasinghal/facial-expression-dataset.

Acest dataset contine imagini separate în directoare cu numele celor 7 emoții pe care le considerăm (angry, disgust, fear, happy, neutral, sad, surprise), imagini in format Grayscale. Imaginile ilustrează fețele unor persoane adulte a căror expresii faciale exprimă emoțiile directoarelor în care se află.

Prima arhitectură este reprezentată de o rețea neuronală convolutivă care primește ca și date de intrare o imagine redimensionată la dimensiunea de 48x48 și care returnează 7 valori reale care reprezintă probabilitatea ca emoția exprimată de imaginea dată ca și input.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 48, 48, 64)	640
batch_normalization (BatchNo	(None, 48, 48, 64)	256
activation (Activation)	(None, 48, 48, 64)	0
max_pooling2d (MaxPooling2D)	(None, 24, 24, 64)	0
dropout (Dropout)	(None, 24, 24, 64)	0
conv2d_1 (Conv2D)	(None, 24, 24, 128)	204928
batch_normalization_1 (Batch	(None, 24, 24, 128)	512
activation_1 (Activation)	(None, 24, 24, 128)	0
max_pooling2d_1 (MaxPooling2	(None, 12, 12, 128)	0
dropout_1 (Dropout)	(None, 12, 12, 128)	0
conv2d_2 (Conv2D)	(None, 12, 12, 512)	590336
batch_normalization_2 (Batch	(None, 12, 12, 512)	2048
activation_2 (Activation)	(None, 12, 12, 512)	0
max_pooling2d_2 (MaxPooling2	(None, 6, 6, 512)	0
dropout_2 (Dropout)	(None, 6, 6, 512)	0
conv2d_3 (Conv2D)	(None, 6, 6, 512)	2359808
batch_normalization_3 (Batch	(None, 6, 6, 512)	2048
activation_3 (Activation)	(None, 6, 6, 512)	0
max_pooling2d_3 (MaxPooling2	(None, 3, 3, 512)	0
dropout_3 (Dropout)	(None, 3, 3, 512)	0
flatten (Flatten)	(None, 4608)	0
dense (Dense)	(None, 256)	1179904
batch_normalization_4 (Batch	(None, 256)	1024
activation_4 (Activation)	(None, 256)	0
dropout_4 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 512)	131584
batch_normalization_5 (Batch	(None, 512)	2048
activation_5 (Activation)	(None, 512)	0
dropout_5 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 7)	3591

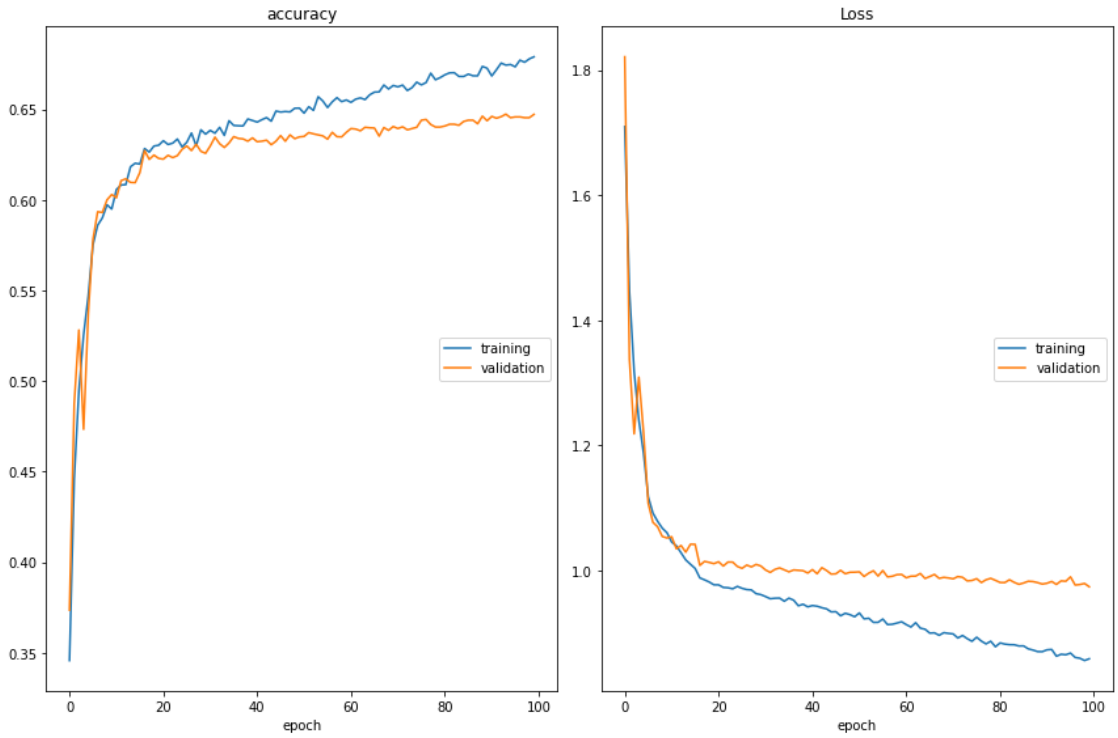
Total params: 4,478,727
Trainable params: 4,474,759
Non-trainable params: 3,968

Modelul a folosit ca și optimizator Adam cu un learning_rate constant de 0.0005. Loss-ul considerat a fost categorical_crossentropy, iar ca metrici de evaluare am folosit acuratețea.

Antrenarea a durat aproximativ 8 ore, pe mașină proprie, folosind CPU.

```
Epoch 00100: saving model to model_weights.h5
Wall time: 7h 56min 40s
```

Acuratețea obtinută dupa 100 de epoci a fost de 64.7. Procesul de învățare este detaliat în graficul de mai jos.




```
accuracy
        training                (min:    0.346, max:    0.679, cur:    0.679)
        validation              (min:    0.374, max:    0.647, cur:    0.647)
Loss
        training                (min:    0.856, max:    1.710, cur:    0.859)
        validation              (min:    0.974, max:    1.821, cur:    0.974)
```

O altă încercare folosind aceiași bază de date a fost cu o arhitectură care primește același tip de date de intrare și returnează același tip de date de iesire, însă diferențele apar între aceste doua straturi. Arhitectura detaliată se poate observa în imaginea de mai jos.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 48, 48, 64)	256
conv2d_1 (Conv2D)	(None, 48, 48, 64)	12352
batch_normalization (BatchNo	(None, 48, 48, 64)	256
activation (Activation)	(None, 48, 48, 64)	0
max_pooling2d (MaxPooling2D)	(None, 24, 24, 64)	0
dropout (Dropout)	(None, 24, 24, 64)	0
conv2d_2 (Conv2D)	(None, 24, 24, 128)	24704
conv2d_3 (Conv2D)	(None, 24, 24, 128)	49280
batch_normalization_1 (Batch	(None, 24, 24, 128)	512
activation_1 (Activation)	(None, 24, 24, 128)	0
max_pooling2d_1 (MaxPooling2	(None, 12, 12, 128)	0
dropout_1 (Dropout)	(None, 12, 12, 128)	0
conv2d_4 (Conv2D)	(None, 12, 12, 256)	98560
conv2d_5 (Conv2D)	(None, 12, 12, 256)	196864
batch_normalization_2 (Batch	(None, 12, 12, 256)	1024
activation_2 (Activation)	(None, 12, 12, 256)	0
max_pooling2d_2 (MaxPooling2	(None, 6, 6, 256)	0
dropout_2 (Dropout)	(None, 6, 6, 256)	0
conv2d_6 (Conv2D)	(None, 6, 6, 512)	393728
conv2d_7 (Conv2D)	(None, 6, 6, 512)	786944
batch_normalization_3 (Batch	(None, 6, 6, 512)	2048
activation_3 (Activation)	(None, 6, 6, 512)	0
max_pooling2d_3 (MaxPooling2	(None, 3, 3, 512)	0
dropout_3 (Dropout)	(None, 3, 3, 512)	0
flatten (Flatten)	(None, 4608)	0
dense (Dense)	(None, 512)	2359808
batch_normalization_4 (Batch	(None, 512)	2048
activation_4 (Activation)	(None, 512)	0
dropout_4 (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 256)	131328
batch_normalization_5 (Batch	(None, 256)	1024
activation_5 (Activation)	(None, 256)	0
dropout_5 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 7)	1799
activation_6 (Activation)	(None, 7)	0

Total params: 4,062,535
Trainable params: 4,059,079
Non-trainable params: 3,456

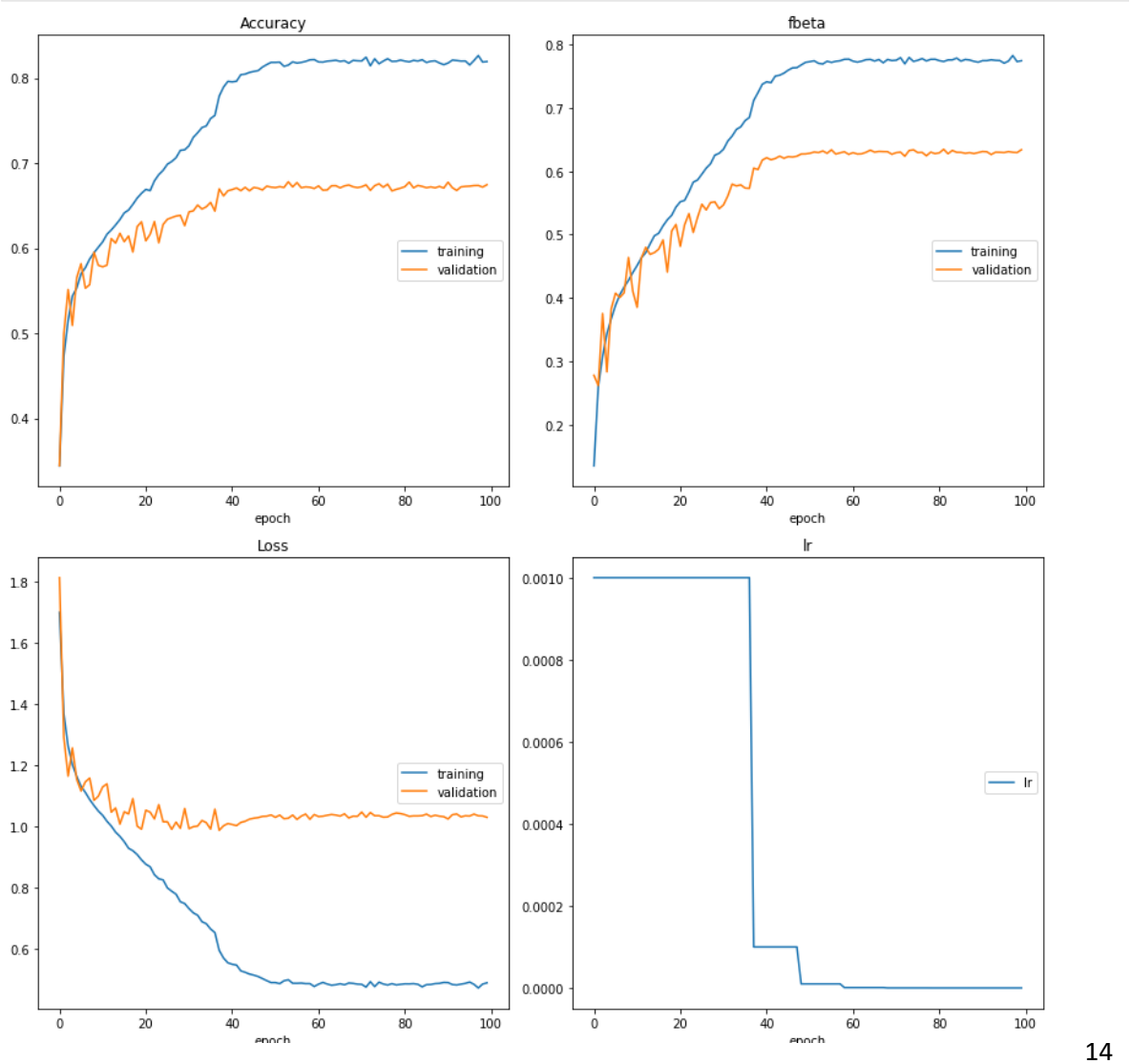
Modelul a folosit ca și optimizator Adam cu un learning_rate default. Loss-ul considerat a fost categorical_crossentropy, iar ca metrici de evaluare am folosit acuratețea si fbeta.

```python
from keras import backend as K

def fbeta(y_true, y_pred, threshold_shift=0):
    beta = 1

    # just in case of hipster activation at the final layer
    y_pred = K.clip(y_pred, 0, 1)

    # shifting the prediction threshold from .5 if needed
    y_pred_bin = K.round(y_pred + threshold_shift)

    tp = K.sum(K.round(y_true * y_pred_bin), axis=1) + K.epsilon()
    fp = K.sum(K.round(K.clip(y_pred_bin - y_true, 0, 1)), axis=1)
    fn = K.sum(K.round(K.clip(y_true - y_pred, 0, 1)), axis=1)

    precision = tp / (tp + fp)
    recall = tp / (tp + fn)

    beta_squared = beta ** 2
    return K.mean((beta_squared + 1) * (precision * recall) / (beta_squared * precision + recall + K.epsilon()))
```

Antrenarea a durat aproximativ 7 ore, pe mașină proprie, folosind CPU.

```
Wall time: 7h 18min 43s
```

Acuratețea obtinută dupa 100 de epoci a fost de 67.5. Procesul de învățare este detaliat în graficul de mai jos.



După analiza performanțelor celor două modele, am decis să continuăm cu arhirectura celuia care a generat acuratețea mai mare.

Pasul următor a fost să antrenăm și să evaluăm această arhitectură pe date reale, cu copii, asfel antrenând pe un subset al bazei de date CAFE, bază de date ce conține imagini cu fețe de copii.

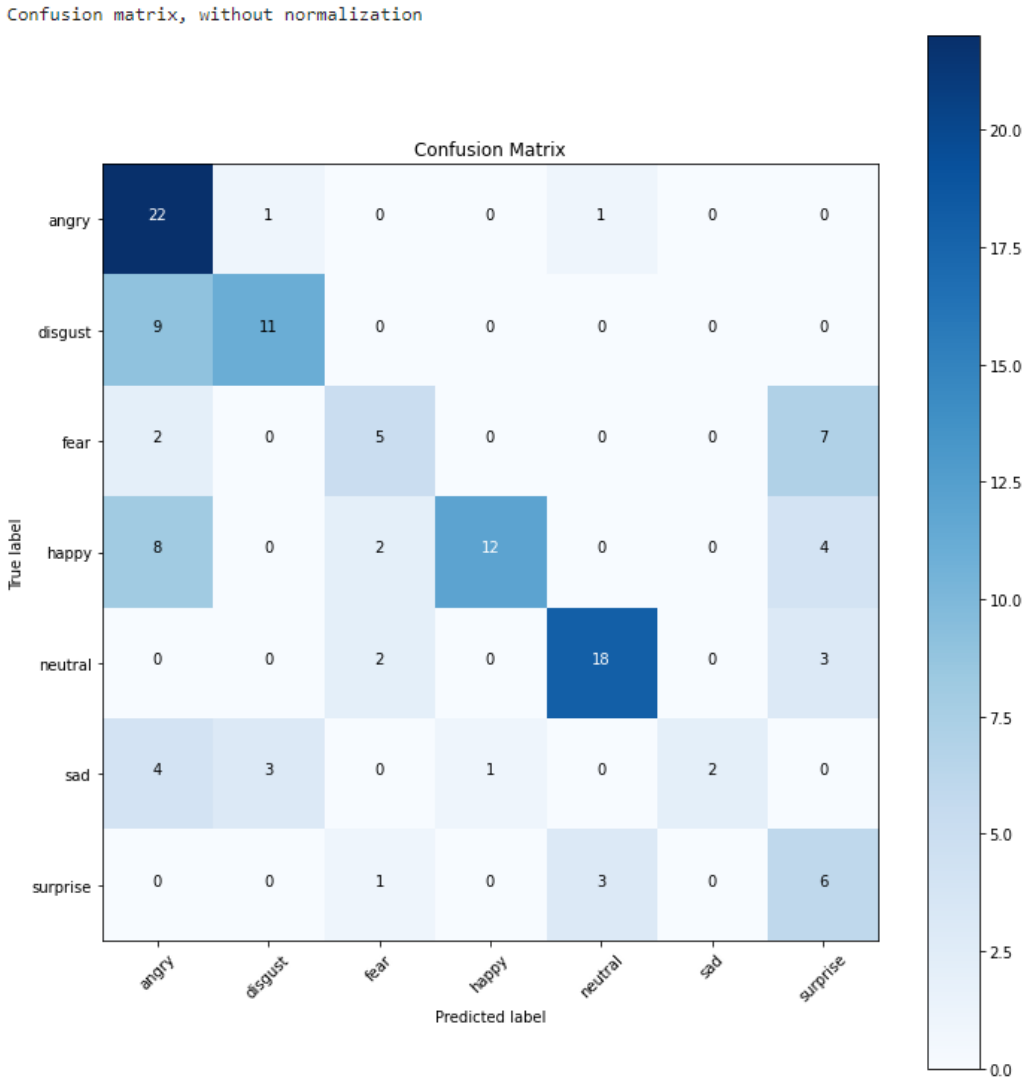
Antrenarea folosind același setup a generat o acuratețe de 65.6%.

Având această acuratețe, am încercat să o îmbunătățim prin completarea datelor de antrenament. Antrenarea pe întreg setul de date CAFE a generat o acuratețe de 67.19%.

Această îmbunătățire ne-a dat de înteles că numarul crecut de date duce la performanțe mai bune, așadar, urmatoarea încercare pentru a îmbunătății acuratețea a fost reprezentată de augmentarea datelor. Acest lucru a fost realizat prin folosirea bibliotecii ImageDataGenerator, imaginile fiind rotite la un numar aleator, si oglindite.

Acestă augmentare a datelor nu a generat performanțe mai bune, acuratețea scăzând drastic la 43.2%.

In final, matricea de confuzie pentru cel mai performant model este:



Procesul de antrenare și dezvoltare a unui model capabil să recunoască emoțiile din semnale audio a început prin încercarea diferitelor arhitecturi pe diferite baze de date.

Prima încercare a fost folosirea unei rețele neuronale convolutive pe concatenare a două baze de date, The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) și SAVEE.

Datele de intrare a rețelei neuronale sunt reprezentate de feature-uri extrase folosind biblioteca LibROSA.

```
df = pd.DataFrame(columns=['feature'])
bookmark=0
for index,y in enumerate(mylist):
    if mylist[index][6:-16]!='01' and mylist[index][6:-16]!='07' and mylist[index][6:-16]!='08'
    and mylist[index][:2]!='su' and mylist[index][:1]!='n' and mylist[index][:1]!='d':
        X, sample_rate = librosa.load('RawData/'+y, res_type='kaiser_fast',duration=2.5,sr=22050*2,offset=0.5)
        sample_rate = np.array(sample_rate)
        mfccs = np.mean(librosa.feature.mfcc(y=X,
                                             sr=sample_rate,
                                             n_mfcc=13),
                        axis=0)
        feature = mfccs
        #[float(i) for i in feature]
        #feature1=feature[:135]
        df.loc[bookmark] = [feature]
        bookmark=bookmark+1
```

Un exeplu a cum arată feturile extrase:

```
train[250:260]
```

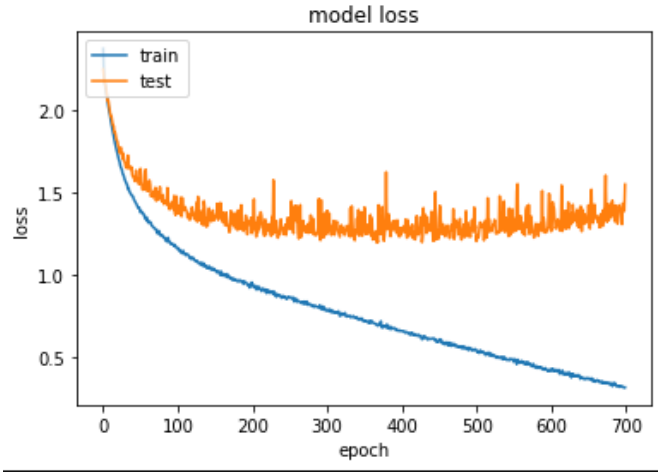
	0	1	2	3	4	5	6	7	8	9	...	207	208	209	210	211	212	213	214	215	
568	-48.660160	-47.174011	-46.640644	-47.137424	-46.684650	-45.625298	-46.475590	-47.645237	-47.000130	-46.469124	...	-21.299217	-21.517799	-18.872599	-17.978498	-19.847363	-21.641239	-22.246859	-18.944397	-14.809800	n
295	-51.315804	-51.868511	-52.083847	-49.493248	-49.743790	-50.106430	-49.594822	-50.763912	-53.579224	-52.005539	...	-21.769327	-21.617216	-22.111790	-24.167032	-25.368141	-26.380276	-25.586905	-22.206667	-17.184998	m
274	-52.899452	-51.317909	-49.772133	-49.043716	-47.140083	-47.156940	-48.022713	-48.754787	-48.948753	-51.214718	...	-25.263315	-24.946465	-25.680208	-26.737732	-26.778913	-27.259239	-27.853085	-18.539705	-13.406015	m
69	-59.151585	-59.151585	-59.151585	-59.151585	-59.151585	-59.151585	-59.151585	-59.151585	-59.149441	-58.931152	...	-32.370949	-31.170330	-28.492161	-26.996382	-26.160204	-26.592119	-27.137033	-21.495016	-16.727299	
247	-54.571712	-54.571712	-54.571712	-54.571712	-54.571712	-54.571712	-54.571712	-54.571712	-54.571712	-54.571712	...	-24.654432	-25.338594	-27.961061	-28.829678	-27.197336	-26.610741	-29.938820	-27.564766	-24.170235	m
64	-61.850487	-61.850487	-61.850487	-61.850487	-61.850487	-61.850487	-61.850487	-61.850487	-61.850487	-61.850487	...	-47.062740	-46.961891	-49.131477	-48.233181	-48.568222	-47.027996	-45.389091	-33.870029	-26.869381	fe
419	-67.574844	-67.574844	-67.574844	-67.574844	-67.574844	-67.574844	-67.574844	-67.574844	-67.574844	-67.574844	...	-49.959904	-51.298141	-52.195721	-51.200710	-47.806763	-48.544357	-47.354431	-34.162224	-26.604845	f
770	-25.481937	-27.502041	-23.210196	-24.669767	-30.271648	-30.442625	-29.336794	-29.348782	-31.109104	-31.967365	...	-44.471603	-45.171860	-45.745831	-46.115700	-47.349754	-46.768497	-45.697914	-45.936916	-46.896629	fem
599	-51.284157	-51.200310	-51.239902	-51.289810	-51.227596	-50.520382	-50.698696	-51.533646	-51.601990	-51.003292	...	-39.087189	-41.031597	-40.735863	-34.857109	-32.654385	-36.674881	-38.501194	-29.162163	-22.963715	fen
384	-46.942928	-47.441555	-47.186558	-46.874348	-46.440910	-45.991253	-46.546867	-47.537704	-48.876869	-49.237633	...	-23.970812	-24.316307	-25.188133	-24.795980	-24.326769	-24.257549	-23.837463	-21.591606	-18.135494	

Modelul folosit la antrenare are o ahirectură ce primește ca și date de intrare un array de 216 numere reale (feature-ile extrase) si returnează un array de 10 numere reale care reprezintă probabilitățile emoțiilor.

```
Model: "sequential_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_40 (Conv1D)           (None, 216, 256)          1536
_________________________________________________________________
activation_50 (Activation)   (None, 216, 256)          0
_________________________________________________________________
conv1d_41 (Conv1D)           (None, 216, 128)          163968
_________________________________________________________________
activation_51 (Activation)   (None, 216, 128)          0
_________________________________________________________________
dropout_10 (Dropout)         (None, 216, 128)          0
_________________________________________________________________
max_pooling1d_10 (MaxPooling (None, 27, 128)           0
_________________________________________________________________
conv1d_42 (Conv1D)           (None, 27, 128)           82048
_________________________________________________________________
activation_52 (Activation)   (None, 27, 128)           0
_________________________________________________________________
conv1d_43 (Conv1D)           (None, 27, 128)           82048
_________________________________________________________________
activation_53 (Activation)   (None, 27, 128)           0
_________________________________________________________________
flatten_10 (Flatten)         (None, 3456)              0
_________________________________________________________________
dense_10 (Dense)             (None, 10)                34570
_________________________________________________________________
activation_54 (Activation)   (None, 10)                0
=================================================================
Total params: 364,170
Trainable params: 364,170
Non-trainable params: 0
```

Optimizatorul folosit a fost RMSprop cu un leaarning_rate de 0.00001, iar loss-ul ales a fost categorical_crossentropy. Ca și metrică de evaluare, s-a folosit acuratețea.

Rețeaua a fost antrenată pentru 100 de epoci si a generat o acuratețe de 46%.



O altă încercare a fost folosind alte 2 modele, pe altă bază de date.

Baza de date folosită este https://www.kaggle.com/uwrfkaggler/ravdess-emotional-speech-audio.

Primul model se foloseste de 3 categorii de feature-uri pentru clasificare:

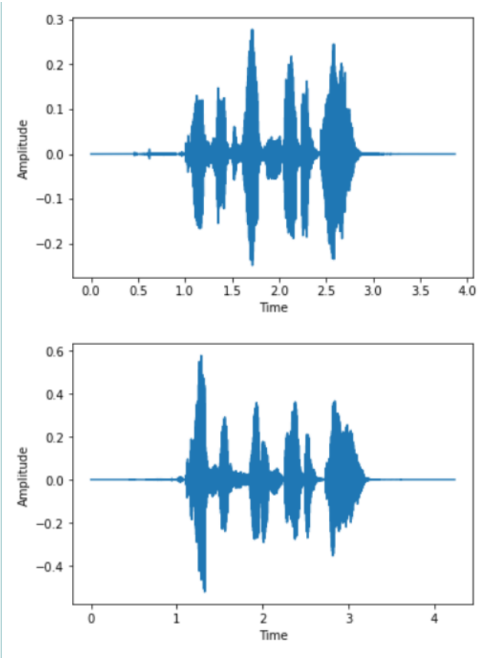
- MFCC: Mel Frequency Cepstral Coefficient, represents the short-term power spectrum of a sound.
- Chroma: Represents 12 different pitch classes.
- Mel: Spectrogram Frequency

Pentru antrenare și testare folosim un model sequential din keras, mai jos se poate observa în detaliu arhitectura acestui model

```
Model: "sequential_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_35 (Dense)             (None, 180, 256)          512
_________________________________________________________________
dense_36 (Dense)             (None, 180, 512)          131584
_________________________________________________________________
dropout_10 (Dropout)         (None, 180, 512)          0
_________________________________________________________________
dense_37 (Dense)             (None, 180, 512)          262656
_________________________________________________________________
dense_38 (Dense)             (None, 180, 256)          131328
_________________________________________________________________
flatten_8 (Flatten)          (None, 46080)             0
_________________________________________________________________
dense_39 (Dense)             (None, 4)                 184324
=================================================================
Total params: 710,404
Trainable params: 710,404
Non-trainable params: 0
_________________________________________________________________
```

Modelul a fost compilat folosind optimizatorul Adam, loss-ul ales este categorical_crossentropy, iar ca metrică de evaluare se folosește acuratețea.

Semnalele audio pot fi vizualizate ca valuri și prin utilizarea valorilor lor putem construi un sistem de clasificare. Mai jos se pot observa imaginile care descriu amplitudinea unor semnale audio.



Acest model generează după 100 de epoci o acuratețe de 44%.

Al doilea model se folosește de aceeași bază de date și de aceleași feature-uri dar pentru antrenare se folosește de un model diferit : multi-layer perceptron model definit in sklearn.

Aceste 3 experimente au dus la alegerea primului model testat pe o bază de date audio pentru copii.

Singura bază de date care să conțină semnale audio si sa fie etichetată este cea oferită de EmoReact, însă etichetele nu sunt puse doar pe baza semnalelor audio, ci si pe semnalele vizuale. După extragerea semnalelor audio din fișierele de tip video pe care le pune la dispoziție EmoReact, primul model a fost reantrenat, însă acuratețea a scăzut la 20%, astfel am rămas la ponderile neuronilor antrenate pe baza de date cu adulți.

Predicted / Actual	angry	calm	disgust	fearful	happy	sad	surprise
angry	49	1	0	2	1	1	0
calm	0	33	0	1	1	5	0
disgust	0	0	3	0	0	0	0
fearful	2	0	0	42	0	3	0
happy	1	1	0	3	44	0	0
sad	1	2	0	1	0	49	0
surprise	0	0	0	0	0	0	3

Integrarea algoritmilor inteligenți în aplicație

Procesul de integrare a algoritmilor în aplicație a fost îmbunătățit din punct de vedere al perfomanțelor prin alegerea calității imaginilor pe care le procesăm. Într-o primă versiune, aplicația folosea imagini captate cu o rezoluție foarte mare, pe care le trimiteam modelului pentru predicție.

Datorită faptului că vrem un flux continuu de imagini ce trebuie procesate, am redus calitatea imaginilor captate pentru ca aplicația să nu se blockeze.

De asemenea, dupa integrarea modelelor și testarea funcționalităților, codul aplicației a fost refactorizat, lucru care a îmbunătățit performanțele aplicației.

Complexitate

Pentru detecția facială, Emoinator folosește biblioteca OpenCV local, ceea ce produce rezultate satisfăcătoare, în timp real. Pentru recunoașterea facială, există de asemenea o implementare locală, folosind un model preantrenat în Python și serializat folosind TFLite.

Asemănător abordării folosită la recunoașterea facială, pentru detecția emoțiilor folosind date de intrare vizuale, se folosește un model preantrenat în Python, de asemenea serializat folosind TFLite.

Aceste modele serializate rulează pe GPU, ceea ce generează performanțe ridicate.

Detecția emoțiilor din sunete, se extrag înregistrări audio continue cu o lungime de 4 secunde, care sunt trimise prin request-uri multipart la serverul de Flask, unde se prezice emoția folosind un model preantrenat in Python. După ce se termină predicția, serverul trimite clientului rezultatul. Comunicarea dintre aplicația Flutter și serverul Flask se face cu ajutorul bibliotecii Dio.

Scalabilitate

Folosirea unei soluții serverless (Firebase) pentru stocarea datelor, impune anumite costuri atunci când numărul de utilizatori simultani și cantitatea datelor cresc. (https://firebase.google.com/pricing)

Serverul Flask, pentru performanțe ridicate, ar trebui încărcat pe un server performant ce poate fi accesat online, iar din punct de vedere al sustenabilității, acesta crează câte un nou fir de execuție pentru fiecare cerere, lucru care permite interacțiunea cu mai mulți utilizatori concomitent.

Concluzii și posibile îmbunătățiri

Aplicația Emoinator este într-o versiune care își propune să demonstreze un concept, acela că este posibilă o analiză automată a emoțiilor copiilor, acest lucru fiind un factor critical în decizia modului în care ar trebui să se interacționeze cu ei.

Pentru a duce această aplicație la un nivel scalabil si funcțional, este loc de îmbunătățiri.

În primul rând, un aspect foarte relevant este reprezentat de absența unei interfețe pentru profesor, datele coelctate nu sunt procesate și prezentate vizual astfel încât profesorul să tragă decizii cu ușurință în ceea ce privește modul în care ar trebui să își adapteze stilul de predare/comunicare pe baza emoțiilor copiilor. Pe această latură vedem o posibilă dezvoltare în ceea ce priveste integrarea unor metode inteligente care să îi sugereze profesorului tipuri de activități asemănatoare celor care au avut rezultate prozitive și sfaturi cu ce categorie de activități să fie evitate pe baza rezultatelor negative.

În final, îmbunatățiri mai pot fi aduse prin creșterea acurateții modelelor îmbogățind datele de antrenament și rafinarea hyperparametrilor.

Referințe

Deshmukh, Yashwardhan. 2021. *On-Device Machine Learning: Train And Run TensorFlow Lite Models In Your Flutter Apps.* 23 01. https://medium.com/google-cloud/on-device-machine-learning-train-and-run-tensorflow-lite-models-in-your-flutter-apps-15ea796e5ad4.

Florian , Schroff, Kalenichenko Dmitry , and Philbin James . n.d. *FaceNet: A Unified Embedding for Face Recognition and Clustering.* https://arxiv.org/pdf/1503.03832.pdf.

Kaipeng, Zhang, Zhang Zhanpeng , Li Zhifeng , and Qiao Yu . n.d. "Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks." https://kpzhang93.github.io/MTCNN_face_detection_alignment/.

Lam, Viola. n.d. *4 Little Trees.* https://www.4littletrees.com/.

Paul Viola, Michael Jones. 2001. "Rapid Object Detection using a Boosted Cascade of Simple." *Carnegie Mellon University .* https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf.

Wikipedia. 2021. 24 11. https://en.wikipedia.org/wiki/WAV.

—. 2021. *Emotion_recognition.* 20 11. Accessed 12 10, 2021. https://en.wikipedia.org/wiki/Emotion_recognition.

wikipedia. fără an. *Facial recognition system.*

Wikipedia. 2021. *Wikipedia.* 9 12. Accessed 12 10, 2021. https://en.wikipedia.org/wiki/Facial_recognition_system.