UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAŞI
FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

API pentru detecţia emoţiilor faciale din imagini

propusă de

Andrei Herdeş

Sesiunea: iulie, 2018

Coordonator ştiinţific
Conf. dr. Miheala Breabăn

UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAŞI
FACULTATEA DE INFORMATICĂ

API pentru detecţia emoţiilor faciale din imagini

Andrei Herdeş

Sesiunea: iulie, 2018

Coordonator ştiinţific
Conf. dr. Miheala Breabăn

Avizat,

Îndrumător Lucrare de Licență

Titlul, Numele şi prenumele _____

Data _____ Semnătura _____

DECLARAȚIE privind originalitatea conținutului lucrării de licență

Subsemnatul(a)……

domiciliul în ……..

născut(ă) la data de ……..…., identificat prin CNP ……..……..……..., absolvent(a) al(a) Universității „Alexandru Ioan Cuza" din Iaşi, Facultatea de ……. specializarea ……, promoția ……..,

declar pe propria răspundere, cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal şi dispozițiile Legii Educației Naționale nr. 1/2011 art.143 al. 4 si 5 referitoare la plagiat, că lucrarea de licență cu titlul:

_____

_____

_____elaborată

sub îndrumarea dl. / d-na _____, pe care urmează să o susțină în fața comisiei este originală, îmi aparține şi îmi asum conținutul său în întregime.

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

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

Dată azi, …… Semnătură student ……

DECLARAŢIE DE CONSIMŢĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licenţă cu titlul “API pentru detecţia emoţiilor faciale din imagini”, codul sursă al programelor şi celelalte conţinuturi (grafice, multimedia, date de test etc.) care însoţesc această lucrare să fie utilizate în cadrul Facultăţii de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” din Iaşi, să utilizeze, modifice, reproducă şi să distribuie în scopuri necomerciale programele-calculator, format executabil şi sursă, realizate de mine în cadrul prezentei lucrări de licenţă.

Iaşi,

Absolvent Andrei Herdeş

_____

Cuprins

Introducere

Inteligenţa emoţională, cunoscută şi ca coeficientul emoţional, este capabilitatea individului de a recunoaşte propriile emoţii dar şi a celor din jur, de a discerne între diferite sentimente şi a le identifica corect. Informaţiile oferite de emoţii ghidează gândirea şi comportamentul individului pentru a se adapta la mediul înconjurător sau pentru a atinge un anumit scop. [1]

Diferite studii precum *Annual Psychology* [2] au arătat efectele pozitive ale unui coeficient emoţional mare în următoarele contexte:

- Interacţiuni sociale mai bune pentru copii - În rândul copiilor şi adolescenţilor, inteligenţa emoţională are un efect pozitiv asupra interacţiunilor sociale şi a relaţiilor, iar lipsa ei duce la devierea de la normele sociale, comportamentul anti-social manifestat atât în cadrul şcolii cat şi în afara ei fiind raportat de copii, membri ai familiei, precum şi profesorii lor.
- Interacţiuni sociale mai bune pentru adulţi - Un grad ridicat de inteligenta emoţională al adulţilor semnifică o mai bună percepţie asupra abilităţilor sociale şi duce la relaţii interpersonale de succes, în timp ce lipsa ei poate avea efecte negative asupra vieţii sociale.
- Persoanele cu un grad ridicat de inteligenţă emoţională sunt percepuţi pozitiv de ceilalţi.
- Relaţii mai bune în cadrul familiei/cuplului.
- Realizări în mediul academic/ Performantă sporită la locul de muncă.
- Bunăstare psihologică – Inteligenţa emoţională poate contribui la creşterea satisfacţiei în viata şi poate înlătura anumite insecurităţi sau cauze ale depresiei.

Nu exista o relaţie directă de tip cauză-efect între aceste lucruri, o parte din ele pot fi puse pe seama inteligenţei generale sau a trăsăturilor personale ale fiecărui individ în parte. Totuşi, a intui emoţiile indivizilor cu o acurateţe ridicată poate fi un avantaj în foarte multe contexte, precum monitorizarea reacţiilor şi emoţiilor unui public expus unei prezentări. Cel care realizează prezentarea poate urmări felul în care ideile sale au fost percepute de către persoanele din public, momentele în care majoritatea şi-a pierdut interesul asupra topicilor abordate. În acest fel se pot face anumite ajustări

prezentării astfel încât să devină mai interactivă. În domeniul marketing-ului pot fi analizate reacţiile oamenilor la reclamele publicate şi nivelul de interes manifestat. Acesta influenţează decizia finală de a cumpăra un produs sau nu. Un alt domeniu este terapia destinată persoanelor ce suferă de autism şi întâmpină dificultăţi în recunoaşterea emoţiilor.

Aspectele prezentate duc la concluzia că un astfel de serviciu este util societăţii şi poate avea un impact pozitiv asupra mediului în care trăim. Soluţii exista deja pe piaţă, voi aminti Kairos API, Emotient, Affectiva sau Emotion API oferit de Microsoft Azure.

Fiind influenţat de evoluţia puternică a învăţării automate din ultimii ani [3], îmi propun prin intermediul acestei lucrări realizarea unui API de detecţie a emoţiilor din imagini, video-uri/livestream-uri ce va fi hostat pe platforma Google Cloud, folosind Google App Engine. Aplicaţia va putea fi integrată în orice alt produs ce doreşte o astfel de funcţionalitate, comunicarea realizându-se prin protocolul HTTP.

Contribuţii personale

În realizarea acestei lucrări am folosit şi integrat numeroase servicii şi framework-uri. Structura proiectului cuprinde doua componente principale: componenta web (aplicaţie web realizată în Java folosind Spring Framework şi un API creat în Python folosind Flask), componenta de învăţare automata (realizarea unei reţele neuronale folosind Python şi Keras). Contribuţiile personale includ realizarea aplicaţiei web şi integrarea cu serviciile oferite de Google Cloud (Google App Engine, Google Cloud SQL), găsirea unui dataset public folosit pentru antrenarea reţelei neuronale cu scopul de a identifica emoţiile din setul furie, fericire, neutru, tristeţe, surprindere. Antrenarea a cuprins mai multe încercări de parametri suportaţi de framework-ul Keras. Pentru API-ul hostat am alcătuit o suita de teste de stres pentru a observa comportamentul platformei în cazul unui volum foarte mare de date de procesat, dar şi felul în care server-ul implementat de mine reacţionează. În funcţiile de rezultatele observate am ajustat configuraţia aleasă pentru deploy, dar am optimizat şi felul în care API-ul procesează request-urile. Consider ca am reuşit sa realizez o aplicaţie ce oferă o experienta plăcută iar modul de utilizare al acesteia este intuitiv.

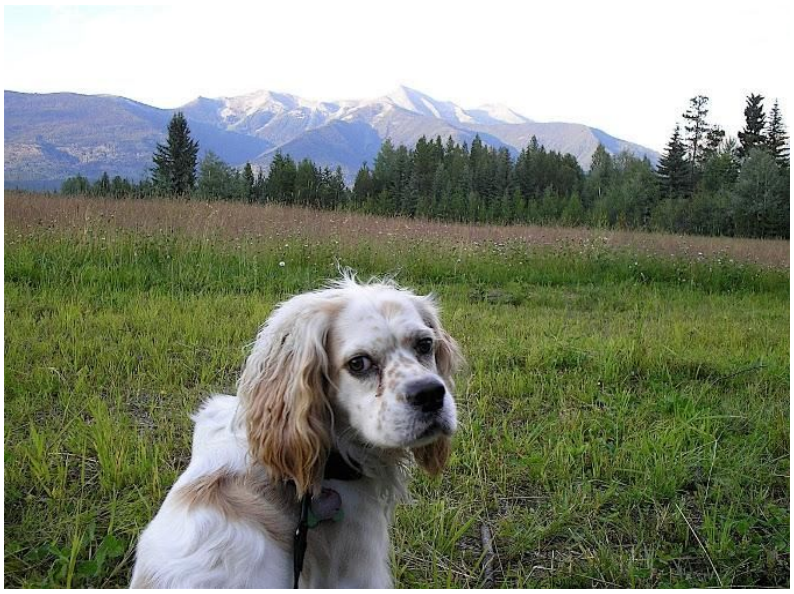
Capitolul 1 Procesarea imaginilor

În prezent există o cerere foarte mare de servicii pentru procesarea imaginilor, aplicaţii ce folosesc tehnici din învăţarea automată pentru analiza, clasificarea sau alterarea acestora în scopuri educative, de divertisment, ştiinţifice, sau ce ţin de siguranţa oamenilor. Google, Microsoft, IBM, Cloudy Vision, Clarifay sau Cloud Sight se află în topul furnizorilor de acest tip [4].

Google Vision API

Google propune Vision, un API ce oferă programatorilor un set de servicii REST, alături de librarii client destinate facilitării comunicării dintre aplicaţia client, ce poate fi scrisa în oricare din limbajele Java, C#, GO, node.js, PHP sau Ruby, şi serverele Google. Astfel de servicii includ detecţia de feţe, logo-uri, label-uri, trăsături caracteristice dintr-o imagine (trăsăturile includ culoarea predominantă regăsită), identificarea text-ului din imagini, sau detectarea conţinutului pentru adulţi.

Detecţia de label-uri se referă la un set larg de categorii ce pot fi regăsite într-o imagine. Variază de la metode de transport pana la identificarea animalelor [5]. Codul din Imaginea 2 reprezintă primele cinci elemente identificate în imaginea trimisă:



Imagine 1: Imagine trimisă serviciului Google Vision API pentru clasificare

```
{
  "responses": [
    {
      "labelAnnotations": [
        {
          "mid": "/m/0bt9lr",
          "description": "dog",
          "score": 0.97346616
        },
        {
          "mid": "/m/09686",
          "description": "vertebrate",
          "score": 0.85700572
        },
        {
          "mid": "/m/01pm38",
          "description": "clumber spaniel",
          "score": 0.84881884
        },
        {
          "mid": "/m/04rky",
          "description": "mammal",
          "score": 0.847575
        },
        {
          "mid": "/m/02wbgd",
          "description": "english cocker spaniel",
          "score": 0.75829375
        }
      ]
    }
  ]
}
```


Imagine 2: Răspunsul primit de la Google Vision API pe baza Imaginii 1

Primele 1000 de request-uri sunt gratuite în fiecare lună pentru fiecare serviciu în parte. Costul următoarelor mii de cereri lunare este de $1.50 per mie, pana la 5,000,000, iar de la 5,000,000 costul fiecărei mii este de $0.60. Există şi varianta „pachetelor promoţionale”, precum detecţia conţinutului pentru adulţi şi detecţia label-urilor din imagini. Primul este gratuit dacă utilizatorul îl foloseşte deja pe cel de-al doilea. [6]

Limitările existente ţin de dimensiunea maximă a imaginilor (20 MB), a obiectului JSON transmis (10 MB), sau numărul maxim de imagini din request (16). Cota de cereri ce pot fi transmise este de maxim 600 pe minut şi 20,000,000 de imagini pe serviciu, pe lună.

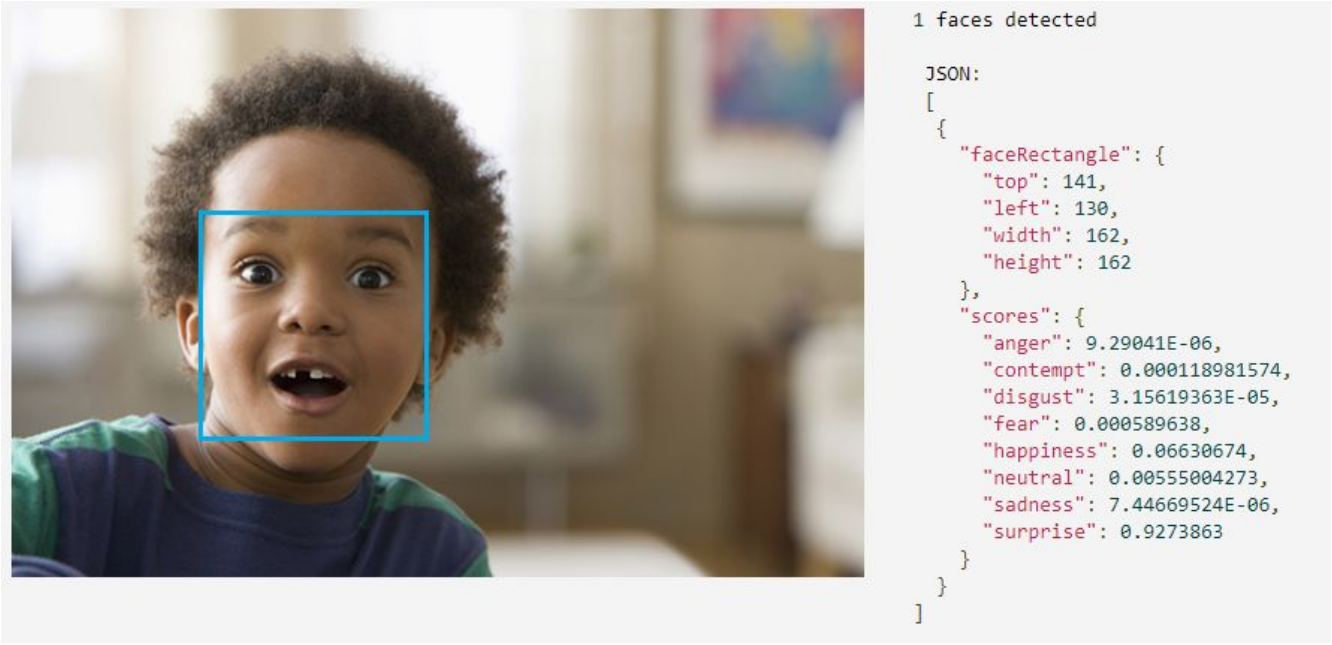
Microsoft Azure

Soluţia celor de la Microsoft este asemănătoare din foarte multe perspective. Procesarea imaginilor în text, identificarea conţinutului pentru adulţi şi atributele dominante ale unei imagini sunt întâlnite şi în cadrul acestui API.

FEATURE NAME:	VALUE
Description	{ "tags": ["outdoor", "grass", "dog", "field", "animal", "mammal", "white", "green", "sitting", "brown", "small", "standing", "laying", "looking", "mountain", "large", "yellow", "grassy"], "captions": [{ "text": "a brown and white dog sitting on top of a grass covered field", "confidence": 0.904050946 }] }
Tags	[{ "name": "outdoor", "confidence": 0.9997894 }, { "name": "grass", "confidence": 0.9995813 }, { "name": "sky", "confidence": 0.9971105 }, { "name": "dog", "confidence": 0.9713887 }, { "name": "field", "confidence": 0.927947044 }, { "name": "animal", "confidence": 0.8070608 }, { "name": "mammal", "confidence": 0.8029416 }, { "name": "white", "confidence": 0.786545038 }, { "name": "green", "confidence": 0.697269142 }, { "name": "lush", "confidence": 0.11259526 }]

Imagine 3: Rezultatele API-ului de la Microsoft [7]

Folosind aceeaşi imagine ca şi parametru observăm similaritatea serviciilor. Librăriile client sunt scrise în limbajele de programare amintite şi în prezentarea API-ului oferit de Google. Totuşi, Microsoft vine în plus cu o funcţionalitate dedicată recunoaşterii de emoţii în imagini. A existat până acum sub denumirea de Emotion, dar începând cu 15 februarie, 2019 va fi asimilat de Face API, ce furnizează capabilităţile generale legate de procesarea feţelor din imagini. Emoţiile pe baza cărora se face clasificarea sunt furie, contemplare, dezgust, frică, fericire, starea neutră, tristeţe şi surprindere. API-ul primeşte ca parametru o imagine şi returnează un JSON care conţine coordonatele feţelor găsite în imagine şi scorul emoţiilor în urma analizei. Un exemplu de output poate fi observat în Imaginea 4.



Imagine 4

Obiectiv

În prezenta lucrare îmi propun sa realizez un API pentru detecţia emoţiilor, hostat de un provider de cloud ce oferă scalabilitatea aplicaţiilor, astfel încât integrările cu aplicaţii existente care doresc o astfel de funcţionalitate să se facă uşor, iar serviciile oferite să rămână constante indiferent de numărul utilizatorilor. Antrenarea reţelei neuronale se va face folosind un set de date disponibil online, gratuit, în încercarea de a vedea ce performanţe pot fi atinse atunci când resursele avute la dispoziţie nu se pot compara cu datele sau infrastructura giganţilor Google sau Microsoft**.**

Capitolul 2 Recunoaşterea feţelor în imagini

Bazele recunoaşterii faciale

Pionierii recunoaşterii faciale automate sunt Woody Bledsoe, Helen Chan Wolf si Charles Bisson. Între anii 1964 şi 1965 cei trei au lucrat împreună pentru a realiza identificarea feţelor din poze. Folosind o bază de date cu foarte multe imagini reprezentând feţe umane, aceştia au încercat să restrângă numărul de imagini existente la un set cu trăsături asemănătoare pozei date ca parametru de intrare. Succesul acestei metode putea fi exprimat prin raportul dintre numărul de imagini selectate şi numărul total de înregistrări din baza de date. Bledsoe a precizat că dificultăţile întâmpinate au fost datorate variaţiei mari întâlnite la gradul de rotaţie al capului, intensitatea luminii, expresia feţei sau vârstei subiecţilor din poze. Aceste variabile făceau ca şi imaginile cu aceeaşi persoană să fie greu de corelat într-un şablon de similarităţi.

Proiectul consta în extragerea coordonatelor trăsăturilor faciale dintr-o imagine de un om, iar apoi oferite programului ca date de intrare. Trăsăturile folosite au fost poziţia pupilelor, laţimea gurii, lăţimea ochilor, sau distanţa dintre pupile. Un operator putea procesa aproximativ 40 de poze pe ora, iar când baza de date a fost construită numele persoanei din poză a fost asociat pozei şi trăsăturilor identificate în aceasta. În faza de recunoaştere, setul de distanţe era comparat cu distanţele identificate în poza folosita ca parametru de intrare, cea mai apropiată înregistrare fiind returnată.

Întrucât este improbabil ca în poze gradul de rotaţie şi înclinare al capetelor să fie acelaşi, toate distanţele prelucrate trebuiau normalizate pentru a reprezenta orientarea frontală a capului. Pentru a realiza această normalizare programul încearcă sa identifice poziţia capului, unghiurile identificate fiind folosite pentru a ajusta imaginea astfel încât capul persoanei din poză să fie poziţionat frontal în cadru. Pentru a calcula aceste unghiuri programul trebuie să ştie geometria tri-dimensională a capului. Având în vedere ca dimensiunile privind forma capetelor subiecţilor fotografiaţi nu erau disponibile, Bledsoe a folosit nişte dimensiuni standard derivate din măsurătorile a şapte capete. [8]

Tehnicile folosite de Bledsoe stau la baza identificării de obiecte în imagini şi derivări ale acestora sunt folosite chiar şi în prezent. Alte modalităţi includ analiza tri-dimensionala a entităţilor ce

trebuie identificate, pentru a capta cu o precizie mai mare informaţiile referitoare la trăsăturile acestora, sau camerele cu viziune termică.

OpenCV

OpenCV este o bibliotecă open source destinată procesării şi prelucrării de imagini, stând la baza multor framework-uri folosite în deep learning, precum TensorFlow, Torch sau CAFFE. Aplicaţii ale acestei biblioteci includ: recunoaştere facială, recunoaştere ale gesturilor, interacţiunea om-calculator, identificarea de obiecte, realitate augumentată. Succesul proiectului se datorează numărului mare de profesionişti implicaţi, fiind demarat iniţial de o filială Intel din Rusia, portabilitatea acestuia, dar şi motivaţiei din spatele lui: de a crea o infrastructură şi un standard pentru procesarea de imagini, oferind persoanelor din domeniu o platformă peste care să poată construi, în loc să fie nevoite să reinventeze roata prin implementarea unor algoritmi deja existenţi. Acest lucru face ca OpenCV, la 18 ani de la momentul lansării, să fie una din cele mai populare librării de acest gen. În acest proiect OpenCV rezolva problema recunoașterii faciale, decisivă în a identifica emoțiile din imagini.

Cascade Haar

În acest proiect a fost folosită soluţia oferită de OpenCV, identificarea caracteristicilor în cascade Haar („Haar-like features”). Secvenţa Haar a fost propusă în 1909 de Alfréd Haar şi reprezintă funcţiile cu o oscilaţie ce poate fi asemănată unui val, de aici şi denumirea de „wavelet”. Folosirea acestui tipar în clasificarea şi detecţia de obiecte a fost propusă de Paul Viola şi Michael Jones în lucrarea lor *„Rapid Object Detection using a Boosted Cascade of Simple Features”*, publicată în 2001. În continuare voi prezenta abordarea acestora.[9]

Acest algoritm utilizează învăţarea automată, modelul fiind antrenat pe foarte multe exemple pozitive şi negative, ca mai apoi sa detecteze în alte imagini tipurile de obiecte pentru care a fost antrenat. În cazul feţelor umane clasificatorul a avut nevoie de foarte multe exemple de poze ce reprezentau feţe si exemple de poze ce nu reprezentau feţe. Procedura de detecţie s-a făcut pe baza valorilor unor proprietăţi simple. Principalul motiv a fost faptul ca un sistem bazat pe proprietăți va fi mai eficient faţă de unul bazat pe pixeli.

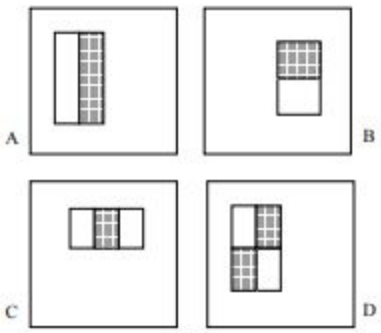


Figura 1

Proprietăţile folosite au la baza funcţia Haar şi sunt reprezentate în Figura 1. Valoarea a doua dreptunghiuri este diferenţa dintre sumele de pixeli ce alcătuiesc cele doua regiuni. Regiunile au aceeaşi formă şi sunt adiacente vertical sau orizontal. Valoarea a trei dreptunghiuri este calculată suma dintre zonele exterioare scăzută din suma pixelilor ce alcătuiesc zona din mijloc. O proprietate de tip patru triunghiuri este cuantificată ca şi diferenţa dintre sumele perechilor diagonale. Având în vedere că rezoluţia de bază a detectorului este 24x24, setul exhaustiv de proprietăţi depăşeşte 180,000.

Pentru a evita un număr atât de mare de computaţii, Paul Viola şi Michael Jones au folosit o reprezentare intermediară a imaginii denumită şi imaginea integrală. Imaginea integrală din punctele x şi y conţine suma de pixeli de deasupra şi de la stânga punctelor x şi y.

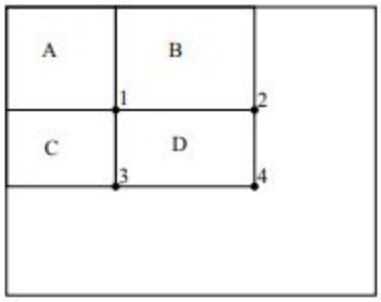


Figura 2

În Figura 2 suma pixelilor dintr-un dreptunghi poate fi calculată folosind cele patru referinţe. Suma imaginii integrale de la locaţia 1 este suma pixelilor din dreptunghiul A. Valoarea de la locaţia 2 este reprezentată de A + B. La locaţia 3 este reprezentată de A + C şi locaţia 4 însumează numărul

pixelilor tuturor dreptunghiurilor. Suma pixelilor din dreptunghiul D poate fi calculată folosind formula 4 + 1 – (2 + 3).

Fiind dat un set de caracteristici, un set de date pozitive şi unul negative, orice metodă de învăţare automată poate fi aplicată. În cazul de faţă Paul Viola şi Michael Jones s-au folosit de o variantă a algoritmului **AdaBoost** atât pentru selectarea unui set de caracteristici cât şi pentru antrenarea clasificatorului. În forma sa orignală, algoritmul de învăţare AdaBoost este folosit pentru a oferi un impuls performanţei de clasificare al unui alt algoritm. Pornind de la ideea ca sunt peste 180,000 de caracteristici dreptunghiulare asociate unei imagini, un număr mult mai mare faţă de numărul pixelilor şi mult prea mare pentru a calcula întreg setul în mod eficient, scopul celor doi a fost de a selecta doar acele caracteristici esenţiale clasificării imaginilor. În acest scop, algoritmul de învăţare a fost proiectat pentru a selecta singura caracteristică dreptunghiulară ce poate separa cel mai uşor un rezultat pozitiv faţă de unul negativ. Experimentele iniţiale au demonstrat că un clasificator construit din 200 de caracteristici are un randament de 95%, cu un raport de fals pozitiv de 1 la 14084. Rezultatele însă nu erau suficiente pentru foarte multe sarcini din contexte reale. Pentru detecţia de feţe în imagini s-a constatat ca algoritmul AdaBoost selectează iniţial proprietăţi uşor de interpretat şi pline de semnificaţie:

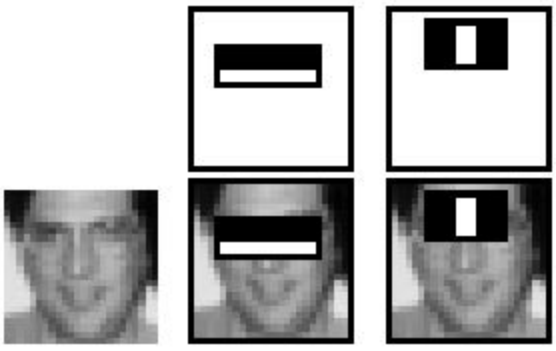


Figura 3

Prima proprietate selectată se concentrează asupra faptului ca zona ochilor este adesea mai închisă la culoare faţă de cea a nasului şi obrajilor, după cum putem observa în Figura 3. Cea de a doua

proprietate compară intensitatea culorii din zona ochilor cu cea de la baza nasului, zona ochilor fiind mai închisă la culoare.

Astfel s-a ajuns la ideea unor clasificatori în cascadă. Fiecare clasificator a fost antrenat pentru a identifica anumite proprietăţi esenţiale iar apelarea fiecăruia dintre ei se făcea pe baza rezultatului obţinut de clasificatorul anterior.

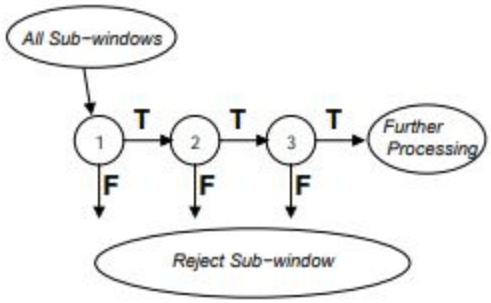


Figura 4

Rezultatul final a fost un clasificator în cascadă format din 38 de straturi, antrenat pentru a identifica faţa umană poziţionată frontal în imagini. A fost folosit un set de exemple pozitive şi unul de exemple negative. Setul pozitiv de antrenament a fost format din 4916 de poze scalate la o rezoluţie de bază de 24x24 pixeli, alături de oglindirea fiecărei imagini, însumând în total un set de 9832 de date. Setul negativ a fost format din 10,000 de imagini ce nu conţineau feţe. Numărul de caracteristici din primele cinci straturi ale detectorului sunt 1, 10, 25, 25 şi 50. Următoarele straturi au o creştere majoră în caracteristici, numărul total al acestora fiind de 6061. Viteza detecţiei este direct proporţională cu numărul de caracteristici evaluate de clasificator pentru fiecare porţiune din imagine. Evaluat pe setul de date de antrenament MIT+CMU, o medie de 10 caracteristici, din totalul de 6061, erau evaluate per porţiune din imagine, pană ca ca detectorul sa ofere un rezultat. În Figura 5 se poate observa cum funcţionează algoritmul în practică şi încearcă să aplice caracteristicile peste imaginea primită ca input. [10]

Figura 5

OpenCV dispune de astfel de clasificatoare deja antrenate pentru detecţia feţei, ochilor sau altor repere faciale, dar oferă şi posibilitatea de antrenare de la zero. În acest proiect am folosit un clasificator pre-antrenat de identificare a feţelor ce privesc frontal camera. După încărcarea modelului, funcţia de clasificare primeşte ca date de intrare imagini încărcate în formatul alb-negru şi returnează două coordonate ce reprezintă latura dreaptă a feţei identificate împreună cu înălţimea şi lăţimea acesteia. Mai departe poza este decupată, procesată şi transmisă reţelei neuronale ce va face clasificarea emoţiei.

Capitolul 3 Reţele neuronale convolutive

Pentru realizarea reţelei neuronale convolutive am folosit Keras, o bibliotecă open-source scrisă în Python ce reprezintă o interfaţă pentru alte framework-uri de învăţare automată, precum TensorFlow sau Theano. Pentru antrenare am recurs la un serviciu cloud de tip PaaS, FloydHub, ce oferă putere de computaţie pentru antrenarea modelelor. Setul de date cu imagini reprezentând emoţii şi adnotat corespunzător a fost disponibil pe un repository public de pe GitHub [11], dar a avut nevoie de o triere a imaginilor înainte de a fi folosit.

Am ales sa folosesc Keras întrucât conţine numeroase implementări ale componentelor unei reţele neuronale, precum straturi, funcţii de cost, optimizări şi un set de elemente ajutătoare ce fac ca lucrul cu imagini sau text să fie mai facil. Oferind un mediu de lucru high-level, modular şi extensibil, proiectarea unei reţele neuronale a fost un proces intuitiv. [12]

De la bun început am observat ca setul de date nu este unul balansat, existând diferenţe mari între numărul de elemente ale claselor sale. Emoţiile cuprinse în dataset sunt: furie, contemplaţie, dezgust, frică, fericire, stare neutră, tristeţe şi surprindere. Datele de antrenament au fost în număr de 9085 de imagini, iar cele de validare 3642, însă predominau imaginile ce desemnau starea neutră şi cea de fericire.

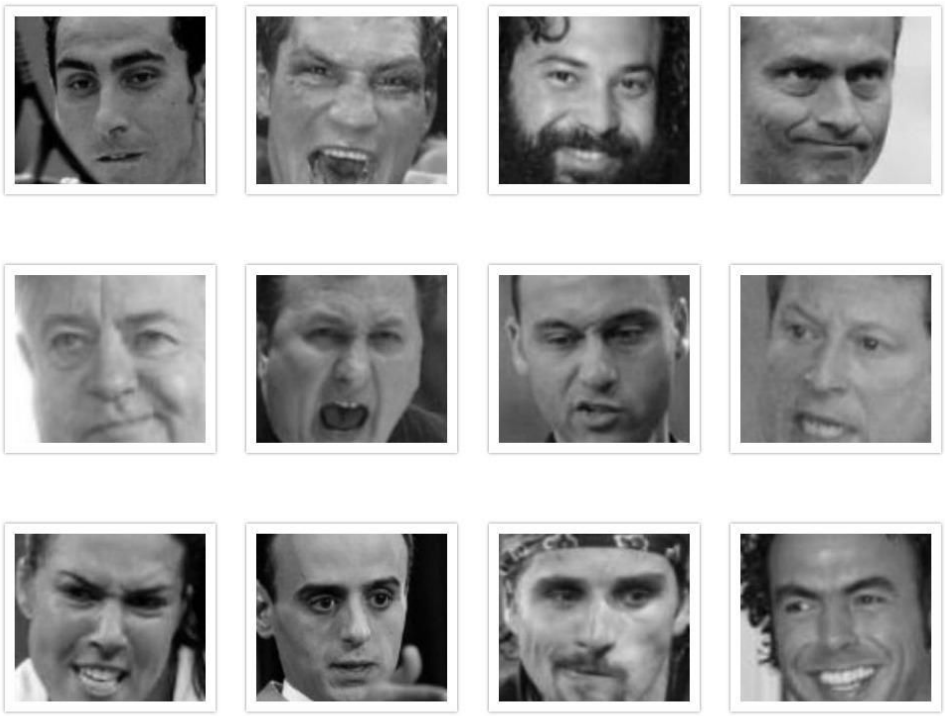


Figura 6: Exemple de imagini din setul de date folosit

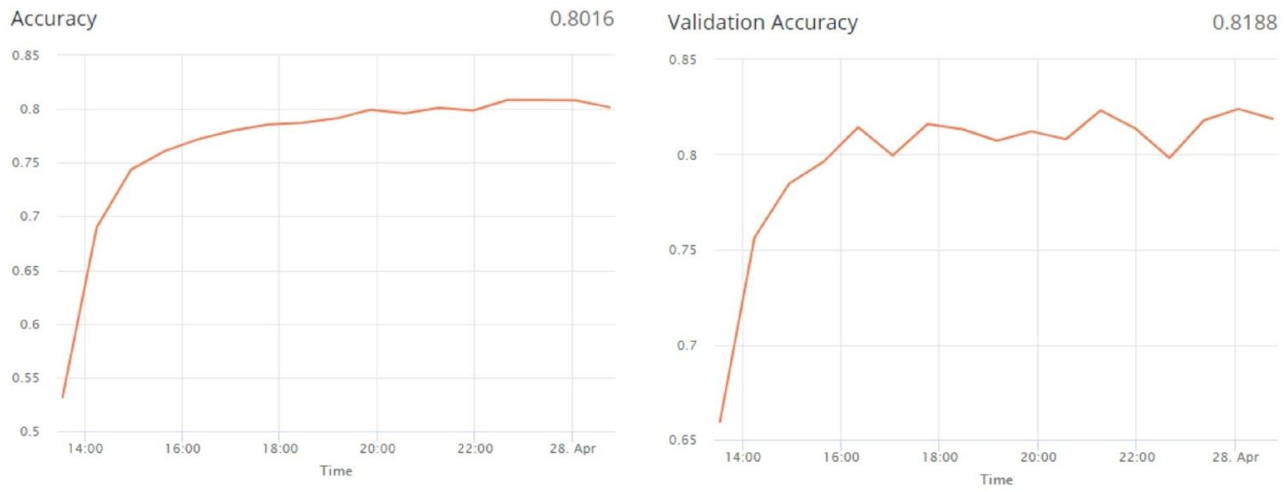


Figura 7: Rezultatele obţinute în urma primei antrenări, folosind FloydHub

Antrenarea modelului a mers pe principiul încercării mai multor parametri şi metode de antrenare, compararea rezultatelor la validare, cat şi pentru imagini noi. Rezultatele primului test nu au fost satisfăcătoare, predicţiile modelului nu erau în concordanţă cu realitatea.

Augumentarea datelor de intrare a îmbunătăţit rezultatele antrenamentului, însă în acelaşi timp acest lucru a dus la un grad de randomizare mai mare a datelor şi astfel la fluctuaţii mai mari în acurateţea la validare, după cum se poate vedea în Figura 8. Augumentările aplicate au fost forfecările aleatoare ale imaginilor şi decuparea anumitor elemente din imagini, fiind aplicat un zoom aleator. acestora.

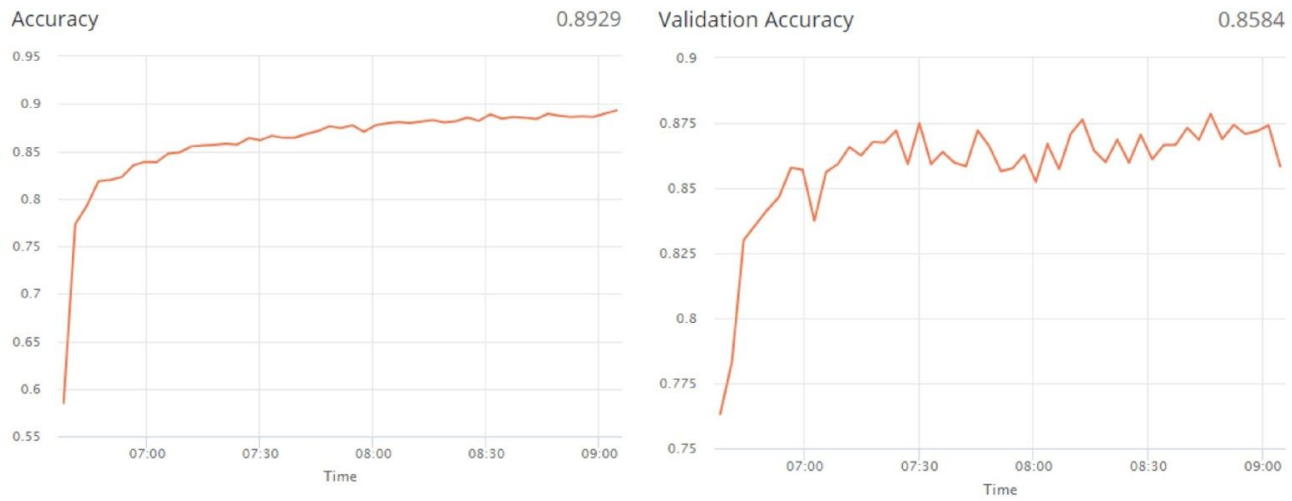


Figura 8: Rezultatele în urma celei de a doua antrenări folosind FloydHub

Deşi precizia clasificatorului a crescut, am realizat ca anumite clase din setul de date reprezinta doar zgomot pentru clasificator din cauza numărului extrem de mic de instanţe. Astfel, am decis sa elimin din setul iniţial de emoţii starea de frică, contemplare şi dezgust. Clasele rămase fiind fericirea, furia, tristeţea, surprinderea şi starea neutră. Fiind în continuare o discrepanţă mare între clasele ce desemnau starea de fericire şi cea neutră, instanţele acestora de antrenament însumând 8558, în timp ce clasele rămase alcătuind împreună 571, am hotărât să adaug parametrul *class_weight* la antrenare. În acest fel o instanţă a uneia din clasele în inferioritate va avea impactul mai multor instanţe ale claselor cu un număr mai mare de imagini.

Figura 9: Rezultatele obţinute în urma celei de a treia antrenări, folosind FloydHub

Conform figurii 9, acurateţea clasificatorului a crescut, însă se observă o inconsistenţă a acurateţei la validare şi creşterea funcţiei de cost la validare pe parcursul acelor inconsistenţe; o posibila cauza fiind over-fitting-ul. În acelaşi timp, în timpul testării cu imagini noi pentru reţea am observat o frecvenţă mare a etichetării pozelor cu una din clasele fericire sau stare neutra, chiar şi în cazul în care stările respective nu se regăseau.

Deşi am încercat sa balansez setul de date, consider ca numărul de imagini în sine este insuficient pentru a încerca clasificarea celor 5 clase cu un randament mare de acurateţe. În acelaşi timp starea neutră poate fi o cauză a over-fitting-ului, întrucât aceasta poate semăna cu oricare alta stare, în funcţie de fizionomia fiecărei persoane în parte.

Capitolul 4 Cloud Computing

Google App Engine

Google App Engine, cunoscut şi sub denumirea de App Engine sau GAE, este un framework web şi o platforma de computaţie în cloud pentru dezvoltarea şi hostarea aplicaţiilor web în centre de date gestionate de Google. Aplicaţiile rulează pe multiple servere, platforma oferind scalare automată în mediul flexibil. Pe măsură ce numărul de request-uri creşte, App Engine alocă în mod automat mai multe resurse pentru ca aplicaţia hostată sa poată face faţă cererii.

Avantajele utilizării acestui serviciu sunt: diversitatea de framework-uri şi medii de rulare de care un programator dispune, fiabilitate, scalabilitate. Toate aplicaţiile App Engine pentru care hostarea este plătită au asigurat uptime-ul 99.95% din timp, iar design-ul acestei platforme este realizat astfel încât, în eventualitatea întreruperii activităţii a mai multor centre de date, serviciile oferite să fie disponibile în continuare, fără întreruperi. [13]

În această lucrare am folosit Google App Engine pentru a hosta 2 aplicaţii web în mediul flexibil, ce comunică între ele. Am ales aceasta platforma datorită documentaţiei generoase de care dispune, comunităţii mari de utilizatori ce pot asista la rezolvarea unor probleme în timpul folosirii platformei, precum şi datorită tool-urilor oferite în consola de administrare a serviciilor cloud. Consider ca această platformă este user-friendly iar cu serviciile oferite se poate lucra într-o manieră uşor de înţeles. API-ul de recunoaştere a emoţiilor din imagini, o aplicaţie scrisă în Python ce foloseşte framework-ul Flask, dispune de un endpoint unde se pot trimite request-uri cu imagini şi oferă în format JSON coordonatele feţelor găsite împreună cu emoţia asociată ei, coordonatele ochilor şi numărul total de feţe găsite. Cea de a doua aplicaţie este layer-ul de prezentare al proiectului. Implementată în Java, folosind framework-ul Spring, aceasta face legătura între utilizator şi clasificatorul de emoţii. Am ales sa creez două aplicaţii deoarece doream ca proiectul sa fie modular, iar API-ul să fie de sine stătător.

Folosind Google App Engine aplicaţiile utilizează mecanismul de load balancing furnizat. Atunci când resursele devin insuficiente, mai multe instanţe vor fi create pentru a face faţă request-urilor. Am supus API-ul unui test de performanţă pentru a mă asigura ca aplicaţia va putea

servi un număr mare de utilizatori fără a întâmpina probleme. Am creat 10,000 de thread-uri ce trimit consecutiv 50 de imagini spre a fi procesate API-ului, testând scalabilitatea acestuia. La primul test, cu o configuraţie modestă, două maşini virtuale cu o memorie de stocare de 10 GB, cu un singur procesor şi 0,5 GB de memorie RAM, pentru a evita eventualele costuri, rezultatele au lăsat de dorit. În punctul în care numărul de request-uri a ajuns la 15 pe secunda, jumătate din acestea sfârşeau prin a semnala o eroare interna a server-ului. Într-un context real, aplicaţia ar fi creat frustrare în rândul programatorilor ce ar fi folosit-o.

Figura 10: Rezultatele obţinute în urma celui de al doilea test

La cel de-al doilea test configuraţia a fost schimbată spre a folosi funcţionalitatea de scalare automată. Configuraţia maşinilor a rămas totuşi aceeaşi. Putem observa conform figurii 10 cum pe măsură ce numărul de request-uri a crescut şi platforma GAE a creat mai multe instanţe care sa servească request-urile. Linia de culoare roz din primul grafic semnifică request-urile finalizate cu eroare din partea server-ului, iar cea portocalie numărul total de request-uri. Gradul erorilor este vizibil mai mic, deşi numărul request-urilor a crescut semnificativ faţă de testul precedent.

Figura 11:Numărul de instanţe este direct proporţional cu numărul de cereri primite

SQL Cloud

Arhitectura aplicaţiilor conţine şi un nivel de persistenţă a datelor oferit tot de Google, SQL Cloud. La fel ca şi în cazul platformei GAE, şi acest serviciu dispune de proprietăţi configurabile de către utilizator, o consola pentru admin în vederea supravegherii acţiunilor desfăşurate folosind baza de date. În plus, serviciile specifice lucrului cu baze de date sunt: mentenanţa (crearea de back-up-uri la intervale stabilite de utilizator) şi replicare, adăugarea de flag-uri, selectarea versiunii (momentan versiunile oferite sunt MySQL 5.7 şi PostgreSQL 9.6). O funcţionalitate deosebită este creşterea automată a spaţiului de stocare odată ce un anumit prag de ocupare este atins. Dacă această setare este activată, spaţiul disponibil este verificat la intervale de 30 de secunde. Când scade sub nivelul configurat de utilizator instanţei îi va fi alocată automat mai multă capacitate de stocare. Un dezavantaj al acestei opţiuni ar fi faptul că memoria de care o instanţă dispune poate fi doar mărită, nu şi micşorată, modificările fiind permanente pe întreaga durată de viaţă a instanţei. O cerere mai mare de spaţiu de stocare doar pentru o perioadă scurtă de timp poate duce la creşteri ale costurilor ce se vor dovedi redundante pe termen lung.

Depăşirea capacităţii maxime admise poate face ca instanţa să fie inactivă, motivul lipsei serviciului nefiind responsabilitatea companiei Google. Se consideră ca fiind vina persoanei ce foloseşte serviciul. Totuşi, este posibilă setarea unui nivel maxim de stocare propriu fiecărei instanţe, astfel încât chiar dacă opţiunea de alocare de resurse este folosită, să existe o limită superioară care să nu poată fi depăşită. [14]

Pentru a utiliza această bază de date este nevoie de o librărie de tip connector. Lucrând în Java, am folosit JDBC (Java Database Connectivity), API-ul ce stă la baza unor framework-uri precum Hibernate sau JPA. Acestea nu sunt încă suportate pe platforma GAE. Din acest motiv interogările şi mapările dintre obiecte şi entităţile din baza de date au fost făcute manual. Legătura dintre baza de date şi aplicaţia web se făcea printr-un singur punct, respectând principiile unicii responsabilităţi. Conexiunile sunt oferite dintr-un pool gestionat de JDBC pentru fiecare request către baza de date.

Aplicaţia a fost structurată pe layere într-o maniera clasică, urmărind good practice-urile existente în industrie. Layer-ul de accesare a obiectelor din baza de date a fost format din interfeţe şi implementările lor. Fiecare interfaţă este construită pentru unul din modelele aplicaţiei, urmărind operaţiile CRUD. Următorul layer este cel de service, şi acesta urmând acelaşi principiu al interfeţelor şi implementărilor adiacente. Aici s-a făcut upload-ul de imagini şi trimiterea request-ului mai departe către API-ul de identificare a emoţiilor, după realizarea procesărilor necesare. Una din transformările esenţiale a fost trecerea de la implementarea fişierelor multipart din Java, particulară framework-ului Spring, la formatul standard al fişierelor în limbajul menţionat. Următorul strat al aplicaţiei face legătura cu layer-ul de front-end. În controllere populez form-urile existente în pagini la iniţializarea acestora, preiau datele introduse de utilizator, acestea fiind validate înainte de procesare. Fiecare controller are mapate endpoint-urile ce le gestionează. Tot din controllere aplicaţia serveşte fişierele statice HTML de pe server.

Capitolul 5 Modul de folosire al aplicaţiei

Produsul final este uşor de folosit. Orice utilizator ce doreşte testarea API-ului va trebui să se înregistreze în aplicaţie. Procesarea tranzacţiilor între aplicaţia web şi baza de date este foarte rapidă datorită faptului ca am ales ca toate modulele să fie hostate pe servere din aceeaşi locaţie, europe-west3.

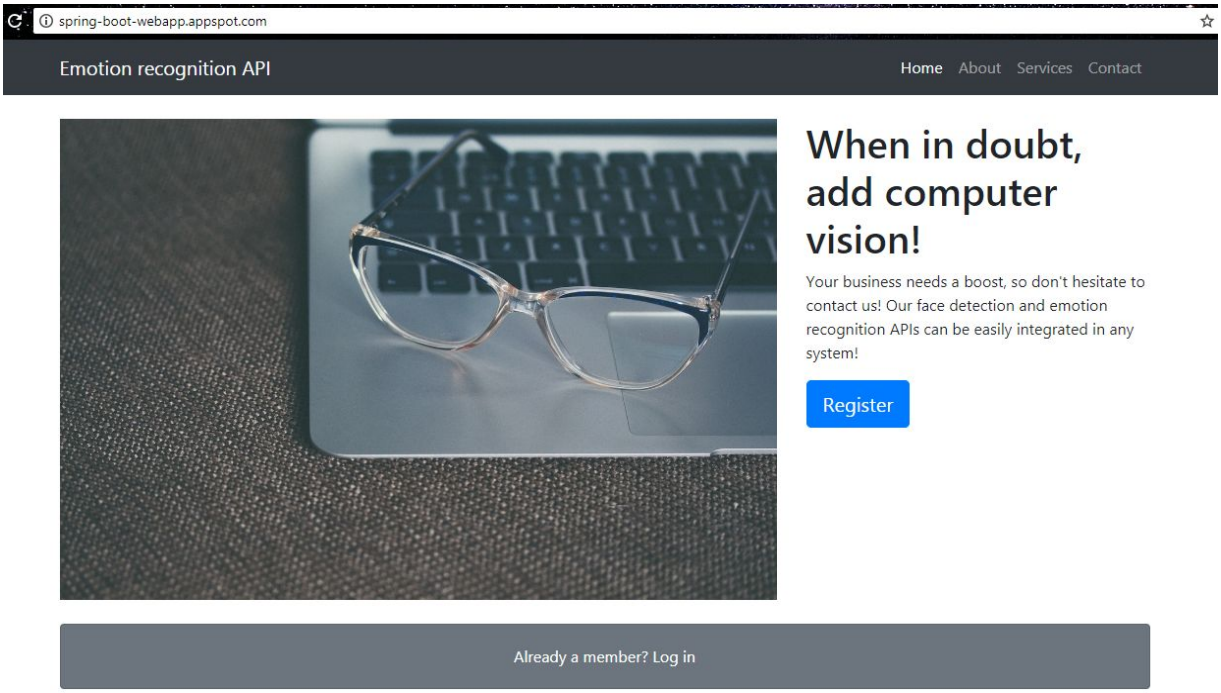


Figura 12: Prima pagină a aplicației

Odată înregistrat, user-ul se poate loga şi testa gratuit API-ul folosind orice dată ca input. Imaginile procesate nu sunt salvate pe server. Răspunsul va fi în format JSON şi va avea o forma de acest gen:

```
response 2: {"face_1":
                  {"coordinates":
                        {"y": 512.0,
                         "width": 219.0,
                         "x": 1122.0,
                         "height": 219.0},
                   "emotion": "happiness"},
             "face_3":
                  {"coordinates":
                        {"y": 726.0,
                         "width": 182.0,
                         "x": 976.0,
                         "height": 182.0},
                   "emotion": "happiness"},
             "face_2":
                  {"coordinates":
                        {"y": 522.0,
                         "width": 285.0,
                         "x": 1526.0,
                         "height": 285.0},
                   "emotion": "happiness"},
             "number_of_faces": 3}
```


Mai departe utilizatorul îşi poate alege planul dorit şi va intra în posesia API-key-ului pentru a putea accesa funcţionalitatea oferită de oriunde. API-key-ul trebuie sa fie un identificator unic ce autorizează utilizatorul sa poată folosi clasificatorul de imagini. Generarea unei astfel de chei se face folosind *UUID.randomUUID()* din platforma Java. Fiecare utilizator care va opta pentru acest serviciu va primi API-key-ul pe adresa de mail cu care s-a înregistrat. Un API-key poate expira sau poate fi dezactivat. Pentru a evita un bottle-neck, vom presupune ca ce depăşeşte 100 de request-uri de la acelaşi utilizator într-un interval de un minut face parte din analiza unui video/live-stream, aşadar nu vom valida de fiecare data API-key-ul. Tot din dorinţa de a optimiza execuţia codului, la fiecare

request primit prin aplicația realizată în Java, un fir de execuție va procesa request-ul pentru clasificatorul de emoții în timp ce altul va loga activitatea în baza de date.

Alt serviciu oferit de acest proiect este transmiterea mail-urilor de atenționare utilizatorului atunci când limita maximă de request-uri este aproape sa fie atinsă.

Capitolul 6 Concluzii finale

Lucrarea „API pentru detecţia emoţiilor din imagini” surprinde tendinţele existente pe piaţa furnizorilor de servicii pentru învăţare automată, încercând să concureze cu cele oferite de corporaţiile mari precum Google sau Microsoft. Având resurse limitate, comparaţia este greu de realizat. Totuşi, folosind funcţionalităţile de hostare ale aplicaţiilor web oferite de Google, proiectul realizat poate face faţă unui număr mare de request-uri şi oferă informaţiile necesare procesărilor ulterioare ale imaginilor. Pe baza datelor furnizate de API se pot crea diverse statistici, în funcţie de necesitatea clientului. Posibile căi de a folosi aceste informaţii sunt analiza reacţiilor unor subiecţi la anumite spot-uri publicitare, sau campanii de marketing, gradul de interes manifestat de elevi sau studenţi pe parcursul prezentării cursului de către profesor, sau în domeniul bancar, prin blocarea tranzacţiilor ce au loc la bancomat pe baza imaginilor înregistrate de camera video din dotarea fiecărui astfel de dispozitiv.

Direcţii de viitor

Cu un set de date mai consistent, posibile direcţii de dezvoltare ale aplicaţiei includ: realizarea unui sistem de generare de statistici şi rapoarte bazat pe fluxul de date primit şi clasificările realizate.

Voi reitera exemplul studenţilor şi reacţiile acestora asupra materialului de curs. O varianta personalizată a sistemului ce analizează momentele în care există o pierdere generală a interesului şi care ar fi motivul acestora, făcut pentru fiecare elev în parte şi aplicat nu doar în condiţiile prezentării cursurilor ar putea duce la construirea unui raport emoţional specific fiecărui elev/student. Acest lucru ar putea fi de folos în domeniile medicale sau sociale.

Altă utilizare a acestui API poate fi identificarea problemelor de natură psihologica în societate. O astfel de soluţie ar face ca mai multe persoane sa beneficieze de ajutor specializat.

Bibliografie

[1], Colman, Andrew (2008).*A Dictionary of Psychology* (3 ed.). Oxford University Press.

[2], Mayer, John D (2008). „*Human Abilities: Emotional Intelligence"*. Annual Review of Psychology.59: 507–536.

[3] Roundup Of Machine Learning Forecasts And Market Estimates, 2018, Forbes

[4], Guarav Oberoi, Comparing the Top Five Computer Vision APIs

[5], Detecting labels with Google Vision

[6], An overview of App Engine

[7], Azure Emotion API

[8], Smriti Tikoo, Nitin Malik, Detection, segmentation and recognition of Face and its features using neural network

[9], Paul Viola, Michael Jones (2001), „Rapid Object Detection using a Boosted Cascade of Simple Feautures"

[10], Adam Harvey Explains Viola-Jones Face Detection

[11], Set de date pentru recunoașterea emoțiilor în imagin

[12], Keras blog

[13], App Engine Service Level Agreement - Google App Engine - Google Cod

[14], Google Cloud SQL instance settings