UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAȘI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

AGEN

Recunoașterea vârstei si a genului utilizând rețele neuronale convoluționale

propusă de

Anca Ignat

Sesiunea: *Februarie, 2019*

Coordonator științific

Lector dr. Anca Ignat

UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

AGEN

Cristian Dorneanu

Sesiunea: *Februarie, 2019*

Coordonator științific

Lector dr. Anca Ignat

Avizat,

Îndrumător Lucrare de Licenţă

Lector dr. Anca Ignat

8 Februarie 2019

DECLARAȚIE privind originalitatea conținutului lucrării de licență

Subsemntatul Dorneanu Cristian, domiciliul în Iasi, născut la data de 12 Martie 1995, identificat prin CNP 1950312226711, absolvent al Universităţii „Alexandru Ioan Cuza" din Iaşi, Facultatea de Informatica, promoţia 2014-2017, declar pe propria răspundere, cunoscând consecinţele falsului în declaraţii în sensul art. 326 din Noul Cod Penal şi dispoziţiile Legii Educaţiei Naţionale nr. 1/2011 art.143 al. 4 si 5 referitoare la plagiat, că lucrarea de licenţă cu titlul: „Agen, recunoaşterea vârstei si a genului cu reţele neuronale convoluționale", elaborată sub îndrumarea dnei Lector dr. Anca Ignat, pe care urmează să o susţin în faţa comisiei este originală, îmi aparţine şi îmi asum conţinutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licenţă să fie verificată prin orice modalitate legală pentru confirmarea originalităţii, consimţind inclusiv la introducerea conţinutului său într-o bază de date în acest scop.

Am luat la cunoştinţă despre faptul că este interzisă comercializarea de lucrări ştiinţifice in vederea facilitării fasificării de către cumpărător a calităţii de autor al unei lucrări de licenţă, de diploma sau de disertaţie şi în acest sens, declar pe proprie răspundere că lucrarea de faţă nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

8 Februarie 2019 Cristian Dorneanu

DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „*Agen*”, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test etc.) care însoțesc 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.

Iași, *8 Februarie 2019*

Cristian Dorneanu

Cuprins

Introducere

1 Motivație

Domeniul științei datelor a evoluat considerabil in ultimele decenii, tehnologia devenind din ce in ce mai folosita in societatea moderna. Descoperirile din acest domeniu au dus la automatizarea proceselor existente din diverse domenii, cum ar fi diversele tehnologii furnizate in industria de autoturisme (autopilot adaptiv) cu scopul de a creste confortul si siguranța. Un alt exemplu este reprezentat de un sistem de recunoaștere faciala folosit pentru a urmări prezenta in cadrul orelor de curs. Descoperirile din acest domeniu si inovațiile care au rezultat datorita acestora, mi-au stârnit curiozitatea, reprezentând motivul alegerii temei pentru aceasta lucrare.

2 Obiective

Obiectivul acestei lucrări este rezolvarea unei probleme din domeniul științei datelor si anume recunoașterea vârstei si a genului unei persoane dintr-o imagine ce conține fața persoanei respective. Relevanța acestei probleme a crescut odată cu apariția unor tehnologii si aplicații cum ar fi, platformele de socializare si sistemele de publicitate orientata.

Aplicația „Agen” este o încercare de rezolvare a acestei probleme folosind rețele neuronale convoluționale pentru clasificarea într-o categorie de vârsta si pentru recunoașterea genului a unei persoane folosind fața acesteia, extrase din imaginea furnizata ca date de intrare. Rețeaua neuronala folosita in aplicația „Agen” a fost dezvoltata folosind librăria Keras iar extragerea feței unei persoane dintr-o imagine este realizata cu ajutorul librăriei OpenCv [1, 2]. Aplicația „Agen” implementează o arhitectura de Microservicii, limbajele folosite in serviciile sale fiind Java si Python. Comunicarea cu aplicația „Agen” se realizează prin intermediul unui API REST-ful. Pentru ca rețeaua neuronala sa fie capabila sa clasifice o fața într-o categorie de vârsta si gen, aceasta trebuie antrenata, folosind date reale.

In următoarele capitole este descrisa problema in detaliu, implementarea propusa de aplicația „Agen” si arhitectura acesteia.

Descrierea unei rețele neuronale convoluționale

O rețea neuronala este o paradigma de programare, inspirata din natura, ce are ca spor învățarea in date observaționale. Diferența intre o rețea neuronala si o rețea neuronala convoluționala consta in arhitectura acestora, cea din urma fiind specializata in lucrul cu imagini. Datele de intrare pentru o rețea neuronala convoluționala sunt, in cele mai multe cazuri, imagini, volume cu o lățime, înălțime si o adâncime ce reprezintă numărul canalelor de culoare. O rețea neuronala convoluționala este alcătuita din mai multe tipuri de straturi, dintre care cele mai importante sunt: stratul convoluțional, stratul de pooling si stratul complet conectat [3].

1 Stratul convoluțional

Stratul convoluțional este blocul central al unei rețele neuronale convoluționale. Fiecare neuron din acest strat este conectat la o regiune din volumul de intrare, dimensiunea acestei regiuni fiind determinata de valoarea hyperparametru-lui cu numele de câmp receptiv. Numărul de neuroni din volumul rezultat in urma stratului convoluțional este determinat de numărul de filtre, dimensiunea filtrului (câmp receptiv) si pasul cu care filtrul este glisat peste volumul de intrare. In cazul in care numărul de parametri (ponderi), dintr-o rețea neuronala convolutionala, este prea mare, se poate aplica o tehnica numita împărțirea parametrilor ce presupune folosirea a aceleași ponderi pentru fiecare strat de adâncime din volumul primit ca date de intrare [4].

2 Stratul de pooling

O arhitectura de rețele neuronale convoluționale este alcătuita din mai multe straturi convoluționale, conectate in serie, intre ca care pot apărea straturi de pooling. Acest strat are ca scop reducerea spațiala a volumului primit ca date de intrare, a numărului de parametri si a complexității computaționale. In acest strat este necesara valoarea a doi hyperparametri, dimensiunea filtrului si pasul cu care filtrul este glisat peste volumul de intrare. Unul din cele mai folosite straturi de pooling este cel de Max pooling unde procesul consta in selectarea celei mai mari valori dintr-o regiune din volumul de intrare, la un anumit pas.

3 Stratul complet conectat

Asemănător unui strat dintr-o rețea neuronala, in stratul complet conectat fiecare neuron este conectat la fiecare neuron din volumul primit ca date de intrare.

Descrierea rețelei aplicației „Agen”

Problema clasificării vârstei si a genului unei persoane a fost împărțita in doua, clasificarea vârstei, respectiv clasificarea genului. Din acest motiv au fost proiectate doua arhitecturi de rețele neuronale convoluționale diferite. Implementarea acestor rețele a fost realizata prin intermediul librăriei Keras. In proiectarea acestor rețele au fost luate in considerare următoarele aspecte:

1. Obținerea si prelucrarea datelor pentru antrenarea rețelei
2. Proiectarea unui model de rețea neuronala pentru recunoașterea vârstei
3. Proiectarea unui model de rețea neuronala pentru recunoașterea genului
4. Antrenarea rețelelor si alegerea de hyperparametri
5. Salvarea modelului si a parametrilor in urma antrenării rețelei
6. Analiza rezultatelor
7. Evaluarea noilor date

In continuare se va descrie fiecare aspect in detaliu din componenta centrala a aplicației „Agen”.

1 Obținerea si prelucrarea datelor

Setul de date pentru antrenare rețelei este format din imagini extrase din baza de date IMDB si Wikipedia [5]. Antrenarea rețelei se face in mod superviza, ceea ce implica faptul ca fiecare imagine va avea atașata vârsta si genul persoanei respective. Setul de date consista in aproximativ 60 000 imagini. Înainte de a antrena rețeaua, se parcurge setul de date si se extrage fața din fiecare imagine, daca exista, si se suprascrie imaginea originala cu fața detectata. Se vor alege doar imaginile ce conțin exact o fața deoarece vârsta si genul ce vor fi folosite pentru antrenare rețelei, pentru a verifica daca aceasta e estimat corect, este calculata la nivelul întregii imagini, astfel existând o singura vârsta si un singur gen la nivelul întregii imagini. Informațiile privind vârsta si genul asociate feței din fiecare imagine sunt stocate într-un fișier de tip „.mat” (Matlab) [5].

Detecția feței se realizează folosind librăria OpenCv. Imaginea este încărcata in modul BGR al canalelor de culoare, este transformata in imagine alb-negru, după care se alege dimensiunea minima pentru care o fața poate fi considerata. In continuare este identificată fața din imagine prin intermediul librăriei OpenCv, folosind Haar Cascades, obținând-se, prin apelarea metodei din librăria OpenCv, coordonatele unui dreptunghi

unde se afla fața recunoscuta. Se trimite in continuare ca rezultat imaginea originala având dreptunghiul feței conturat si dreptunghiul extras din imaginea inițiala.

Din setul de date total colectat, 10% au fost folosite pentru validare si 90% pentru antrenare rețelei.

Distribuția vârsta/gen pe număr de imagini.

1-10	11-15	16-21	22-28	29-35	36-42	43-50	51-65	66-80	81-100
247	655	4424	12572	12365	9981	9981	7296	2142	493

Bărbat	Femeie
35822	22318

2 Proiectarea rețelei pentru recunoașterea vârstei

Pentru proiectarea rețelei au fost considerate trei aspecte: dimensiunea stratului de intrare, numărul claselor rezultate in stratul de ieșire si proiectarea straturilor centrale. Pentru stratul de intrare, dimensiunea unei imagini a fost definita având o lungime de 128, o lățime de 128 si o adâncime de 3 (R, G, B), motivul alegerii acestor dimensiuni fiind reducerea complexității de timp si spațiu si divizibilitatea numărului 128 cu numărul 2 (in cazul straturilor de pooling, când dimensiunea va fi redusa).

Pentru stratul de ieșire, au fost selectate 10 categorii de vârsta 1-10, 11-15, 16-21, 22-28, 29-35, 36-42, 43-50, 51-65, 66-80, 81-100. Aceasta configurație a fost aleasa pentru o acuratețe mai mare, scopul aplicației nefiind recunoașterea exacta a vârstei ci clasificarea într-o categorie. Selecția a 100 categorii de vârsta ar fi rezultat într-o acuratețe mai scăzută si o complexitate de spațiu si timp mai mare, lucru care nu ar fi fost suportat de infrastructura unde a fost dezvoltata aplicația „Agen” [6].

Modelul este alcătuit din trei straturi convoluționale cu numărul de filtre egal cu 128, 256, 256, in aceasta ordine, cu o lățime si lungime a filtrului egala cu 3, un pas egal cu 1 si o umplutura a volumului de input cu valoarea 0 astfel încât dimensiunea in lungime si lățime a stratului convoluțional sa fie egala cu cea a volumului de intrare. Funcția de activare folosita pentru aceste straturi este softmax. Intre aceste straturi convoluționale se afla cate un strat de pooling cu o lungime si lățime a filtrului egala cu 2 si un pas egal cu 2, cu scopul de a reduce dimensiunea volumului după fiecare convoluție cu 75%.

In continuarea acestor straturi se afla un strat de normalizare de tip batch si un strat complet conectat având 11 noduri reprezentând clasele rezultate. Structura rețelei poate fi schițata in următorul mod:

INPUT -> [CONV -> RELU -> POOL]*2 ->CONV -> RELU -> NORM -> POOL -> FC

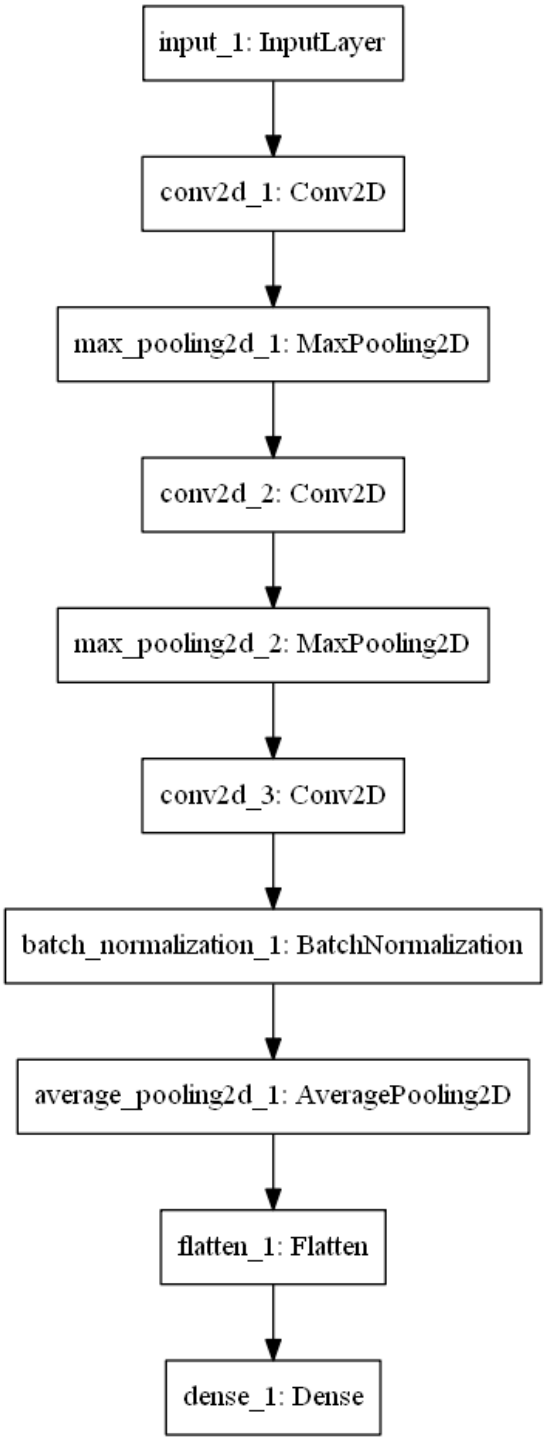
3 Proiectarea rețelei pentru recunoașterea genului

Similar proiectării rețelei pentru recunoașterea vârstei, au fost considerate aceleași aspecte.

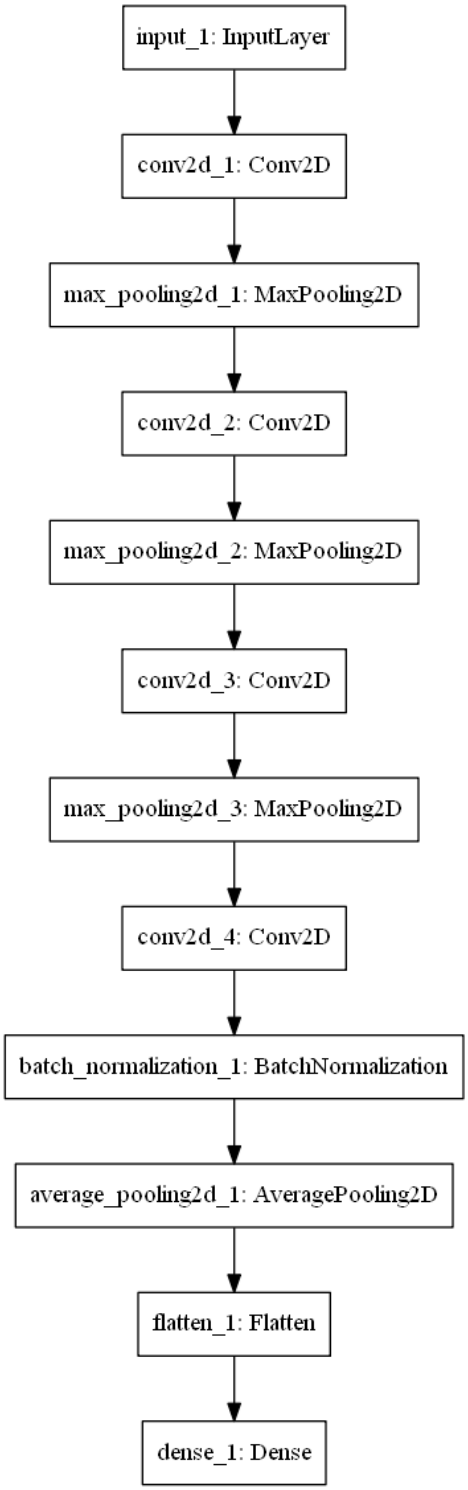
Pentru stratul de ieșire, au fost selectate 2 categorii de gen cu valoare 1 pentru masculin respectiv 0 pentru feminin. Modelul este alcătuit din patru straturi convoluționale cu numărul de filtre egal cu 128, 128, 256, 256, in aceasta ordine, cu o lățime si lungime a filtrului egala cu 3, un pas egal cu 1 si o umplutura a volumului de input cu valoarea 0. Funcția de activare folosita pentru aceste straturi este softmax. Intre aceste straturi convoluționale se afla cate un strat de pooling cu o lungime si lățime a filtrului egala cu 2 si un pas egal cu 2.

In continuarea acestor straturi se afla un strat de normalizare de tip batch si un strat complet conectat având 2 noduri reprezentând clasele rezultate. Structura rețelei poate fi schițata in următorul mod:

INPUT -> [CONV -> RELU -> POOL]*3 ->CONV -> RELU -> NORM -> POOL -> FC



Structura retelei neuronale convolutionale pentru recunoasterea varstei



Structura retelei neuronale convolutionale pentru recunoasterea genului

4 Antrenarea rețelei

Aplicația „Agen” folosește o rețea neuronala convoluționala antrenata într-o maniera supervizata ceea ce presupune ca fiecare fața din setul de date va conține vârsta, in acest caz categoria de vârsta, si genul real transformate sub forma unor vectori de o dimensiune astfel încât sa poată fi comparat cu rezultatul produs de rețea. Antrenarea rețelei presupune ca fiecare imagine sa fie sa fie trimisa ca data de intrare a rețelei iar rezultatul obținut (clasificare) sa fie comparat cu rezultatul real atașat imaginii, iar in cazul in care sunt diferite, ponderile rețelei sa fie modificate corespunzător.

Ca si funcție de optimizare este folosita SGD (Stochastic Gradient Descent) cu o rata de învățare inițiala egala cu 0,01 un momentum egal cu 0,9 si o descompunere de 0,005. La fiecare 4 epoci, rata de învățare va fi înmulțita cu un factor de 0,1. Ca si funcție de pierdere este folosita Cross Entropy. Dimensiunea lotului aleasa este de 32 din motivul insuficientei de resurse computaționale. Rețeaua este antrenata pe parcursul a 20 de epoci iar ponderile rețelei sunt salvate la terminarea unei epoci in cazul in care valoare de pierdere a scăzut [7].

Prin intermediul implementarii TensorFlow aleasa ca backend al librăriei Keras, rețelele au putut fi antrenate folosind placa video a sistemului unde a fost dezvoltata aplicatia „Agen”, Nvidia, cu ajutorul tehnologiei CUDA, sporind astfel eficienta in timp, fiind nevoie de aproximativ 50 minute pentru a antrena o rețea pe parcursul a 20 de epoci.

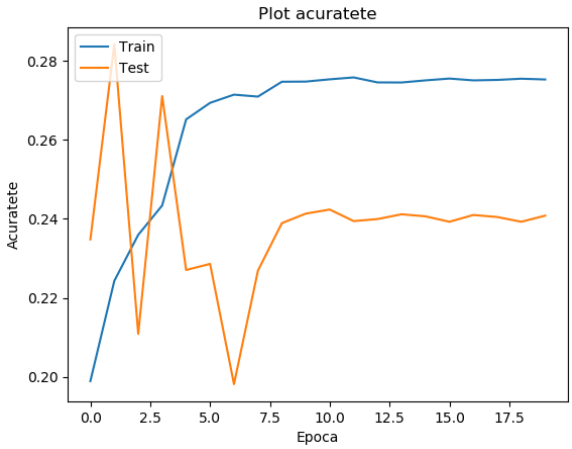
5 Salvarea modelului

Pentru a nu antrena rețeaua de fiecare data când aplicația „Agen” este pornita (proces costisitor in timp), rețeaua este antrenata doar in cazul in care nu exista nici un fișier cu ponderile rezultate dintr-o antrenare anterioara a rețelei. In cazul in care exista acest fișier, rețeaua va încarcă ponderile salvate, fără a mai fi nevoie de a o antrena din nou.

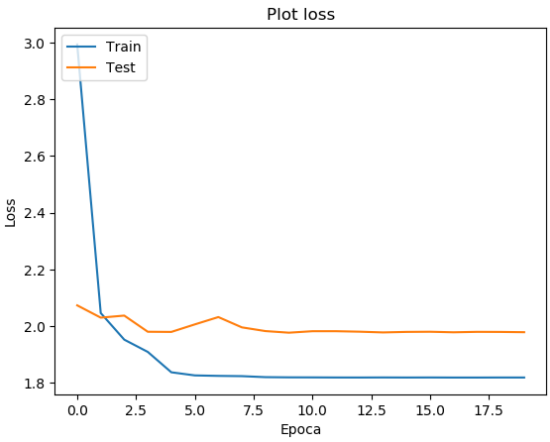
6 Analiza rezultatelor

După antrenare rețelei pentru clasificare vârstei s-a obținut, pentru datele de antrenament o acuratețe de 27,6% si o valoare a pierderii egala cu 1,82, iar pentru datele de validare, o acuratețe de 24,2% si o valoare a pierderii egala cu 2,04.

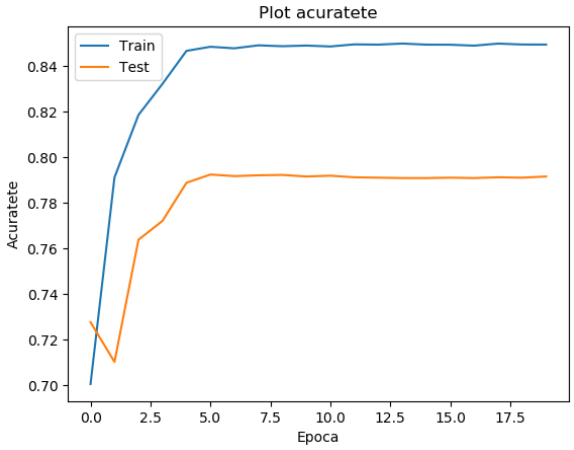
După antrenare rețelei pentru clasificare genului s-a obținut, pentru datele de antrenament o acuratețe de 85,12% si o valoare a pierderii egala cu 0,34, iar pentru datele de validate, o acuratețe de 79,1% si o valoare a pierderii egala cu 0,49.



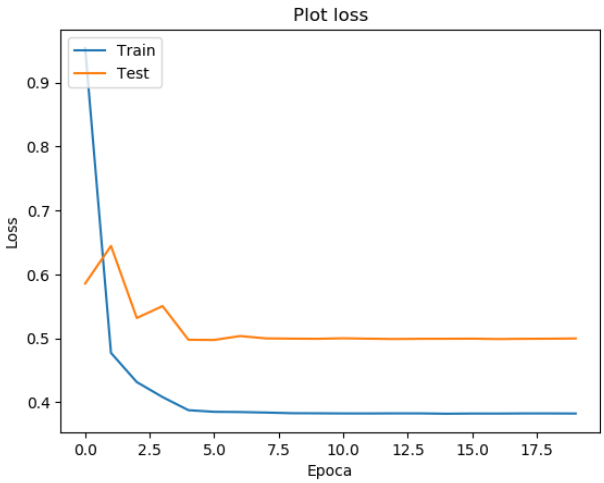
Plot pentru acuraterea retelei neuronale in recunoasterea varstei



Plot pentru pierderea retelei neuronale in recunoasterea varstei



Plot pentru acuraterea retelei neuronale in recunoasterea genului



Plot pentru pierderea retelei neuronale in recunoasterea genului

Pentru a testa acuratețea rețelelor obținuta in urma antrenamentului, am folosit Adience Benchmark[8]. Setul de date Adience consista in 26 mii imagini, încărcate de pe smart-phone, având astfel multiple variații ale poziției capului.

	Acuratețe vârsta	Acuratețe gen
Agen	24.2	79.15
Benchmark Adience	**45.1**	**79.5**

7 Evaluarea

Pentru evaluarea unei noi imagini, mai întâi se extrage fața din imagine, apoi fața extrasa este transmisa ca data de intrare către rețea iar rezultatul este decodificat la clasa de vârsta si gen la care aparține persoana din imagine.

Aplicația „Agen”

Aplicația implementează o arhitectura de microservicii. Deși simplitatea aplicației si numărul redus de servicii nu justifica adoptarea acestei arhitecturi, in acest stadiu, aceasta arhitectura a fost aleasa pentru posibilitatea de scalabilitate pe care o oferă si nivelul scăzut de cuplare ce facilitează adăugarea de noi servicii si funcționalități [9].

Se identifica doua servicii principale ale acestei aplicații. Un serviciu implementat folosind Java si Spring Boot ce are drept rol comunicarea cu clientul prin expunerea unui API REST-ful si salvarea estimări (sub numele de calculare in aplicație) in baza de date. Serviciul central aplicației este implementat folosind limbajul Python si are ca rol clasificarea într-o clasa de vârsta si gen a unei persoane pe baza unei imagini primite ca date de intrare. Tot aici are loc procurarea si prelucrarea datelor de intrare si antrenarea rețelei [10].

In construirea aplicației următoarele aspecte au fost luate in calcula:

1. Arhitectura si componentele aplicației
2. Comunicarea intre componente
3. Procesul de recunoaștere a vârstei si genului pe baza unei imagini
4. Scalabilitate si extensibilitate
5. Direcții viitoare

1 Arhitectura si componentele aplicație

Vom numi calculare, procesul de recunoaștere a vârstei si genului unei persoane.

Aplicația „Agen” este structurata după arhitectura de microservicii având următoarele servicii/componente: „gateway”, „config”, „eureka-server”, „calculation-api-service”, „calculation-core-service”, „ageandgender-app”. Pentru procesul de deployment este folosit Docker, fiecare serviciu având un fișier „Dockerfile” unde sunt specificate operațiile pentru inițializarea unei imagini ce va conține toate dependentele necesare rulării serviciului respectiv.

1.1 Componenta „config”

Componenta „config” acționează ca un loc central de stocare a configurațiilor tuturor serviciilor ce fac parte din aplicația „Agen”. Fiecare serviciu este dependent de aceasta componenta, din acest motiv cerința pentru aceasta componenta este disponibilitate mare si toleranta la erori. Aceasta componenta este implementata folosind limbajul Java si framework-ul Spring boot.

1.2 Componenta „calculation-api-service”

Componenta „calculation-api-service” este implementata folosind limbajul Java si framework-ul Spring Boot si are rolul de a expune un API REST-ful pentru aplicația „Agen”. Se folosește o baza de date H2 pentru stocarea de cereri/estimări (calculări). Aplicația neavând un serviciu de autentificare sau funcționalitate de corelare a cererilor cu utilizatorul, nu este nevoie folosirea unor baze de date specializate cum ar fi MySql. Este expus un singur endpoint, „/calculate”, ce suporta metodele POST pentru inițierea procesului de estimare a vârstei si genului pe baza unei imagini primite ca data de intrare si salvarea acesteia in baza de data si metoda GET pentru extragerea estimării din baza de date.

Funcționalitatea metodei POST consta doar in inițierea procesului de calculare si salvarea in baza de date a cererii si nu cea de furnizare a rezultatului, aceasta metoda furnizând un obiect JSON cu valoarea câmpului „id”, pentru a putea fi extras din baza de date, si imaginea primita ca date de intrare sub forma de sir de caractere. Astfel clientul va trebui sa apeleze metoda GET folosind valoarea câmpului „id” furnizata la inițierea procesului pana când va primi un obiect JSON ce va conține câmpul „status” cu valoarea „FINISHED” sau „ERROR”, caz in care va fi furnizat un obiect JSON cu estimările de vârsta si gen sau mesajul de eroare in cazul unei erori.

1.3 Componenta „calculation-core-service”

Funcționalitatea de recunoașterea vârstei si genului unei persoane este implementata in componenta „calculation-core-service” folosind limbajul Python versiunea 3.6. La pornirea serviciului se verifica daca exista ponderi ale rețelelor salvate in sistemul de fișiere asociat serviciului. Daca nu exista atunci se executa metoda de configurare unde se verifica mai întâi existenta arhivelor ce conțin setul de imagini, iar in cazul in care nu exista se vor copia local. In continuare imaginile sunt extrase si salvate pe disc, urmând sa fie suprascrise fiecare cu fața extrasa, redimensionata la o lățime si lungime de 128 si o adâncime de 3 (R, G, B). Fetele sunt încărcate in memorie unde sunt asociate vârsta si genul pentru fiecare, după care începe procesul de antrenare al rețelei si salvare al ponderilor. Acest întreg proces este foarte costisitor in timp, in principal procesul de stocare al arhivele.

Comunicarea cu serviciul „calculation-api-service” este realizata prin intermediul unui broker de mesaje numit RabbitMQ [11]. După salvarea sau încărcarea ponderilor, serviciul va fi pus in stare blocanta, așteptând primirea unui mesaj/task (cerere de calculare). După primirea unui task, se va estima genul si vârsta din imaginea primita, in cazul in care exista, iar rezultatul va fi publicat in coada, urmând sa fie primit de serviciul „calculation-api-service” unde va fi stocat in baza de date.

1.4 Componenta „ageandgender-app”

Aplicația „Agen” conține un client ce va consuma REST API-ul produs de serviciul „calculation-api-service” [12]. Aceste serviciu a fost dezvoltat folosind framework-ul Anglular, iar pentru partea grafica, Material Design Bootstrap. Prin intermediul acestuia se va declanșa procesul de calculare, prin apelarea metodei POST a endpoint-ului „agen/api/calculate” expus de serviciul „gateway”. Acest serviciu are rolul de a direcționa cererea către serviciul corespunzător, in cazul de fața „calculation-api-service”, unde cererea va fi salvata in baza de date si un nou task va fi creat si publicat pentru serviciul „calculation-core-service”. In urma calculării, rezultatul va fi afișat împreuna cu imaginea de intrare si fața detectata.

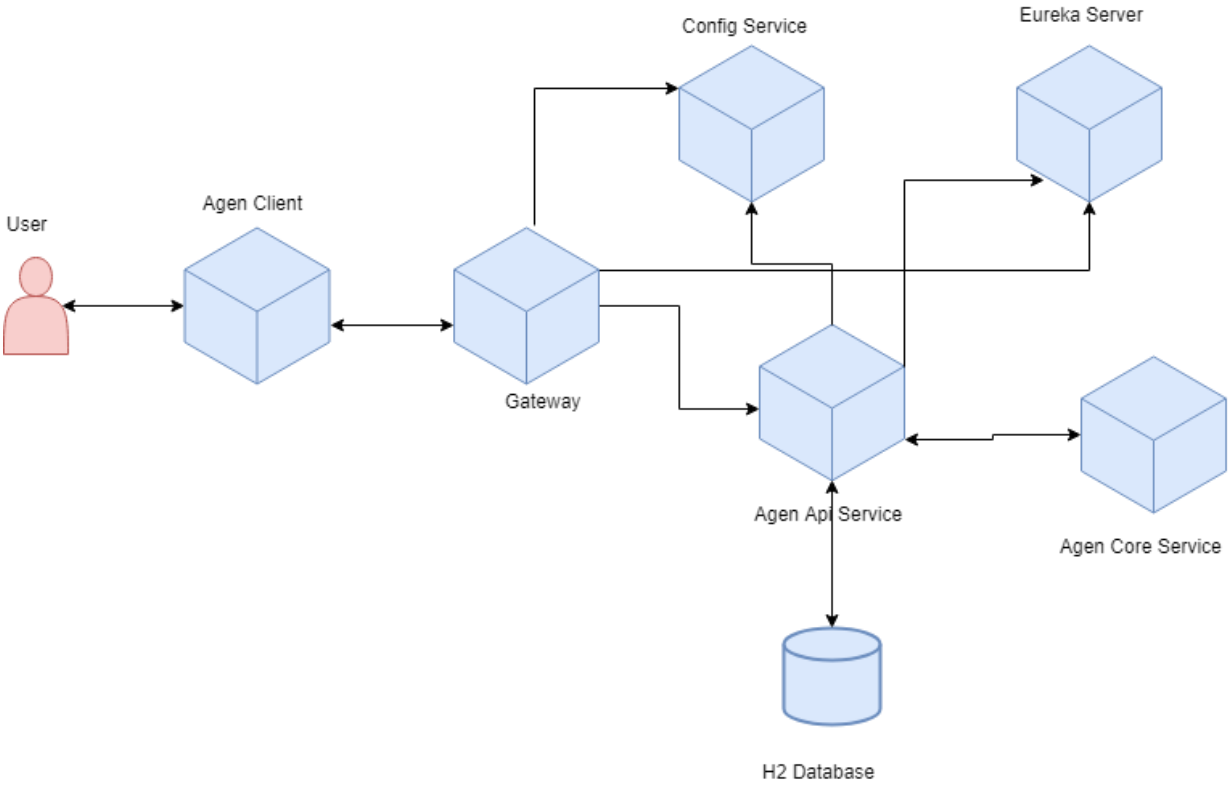


Diagrama a arhitecturii aplicației Agen

2 Comunicarea intre componente

Comunicare intre serviciile „calculation-api-service” si „calculation-core-service” este realizata prin intermediul unui broker de mesaje numit RabbitMQ. Sunt definite doua cozi pentru schimbul de mesaje, „calculate.tasks.queue” si „calculate.result.queue”. In prima coada, procesul de calculare este inițiat, serviciul „calculation-api-service” publicând o cerere de calculare. Rezultatul este așteptat in cea de-a doua coada, unde se va realiza si salvarea in baza de date.

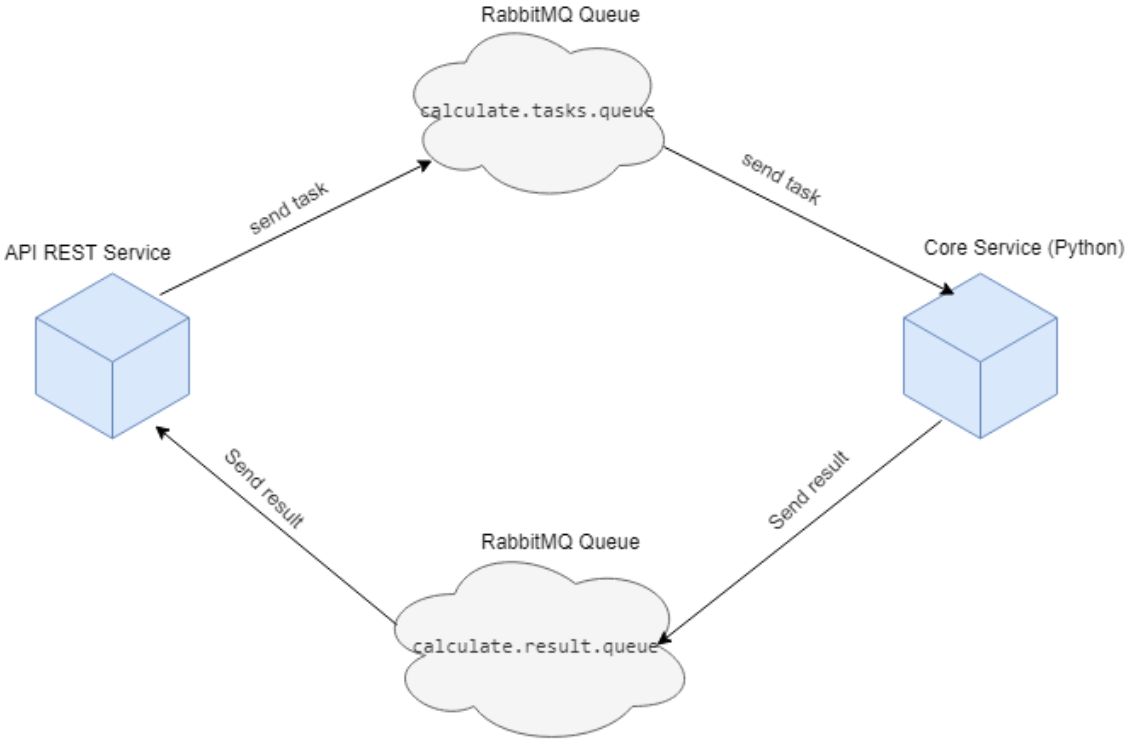
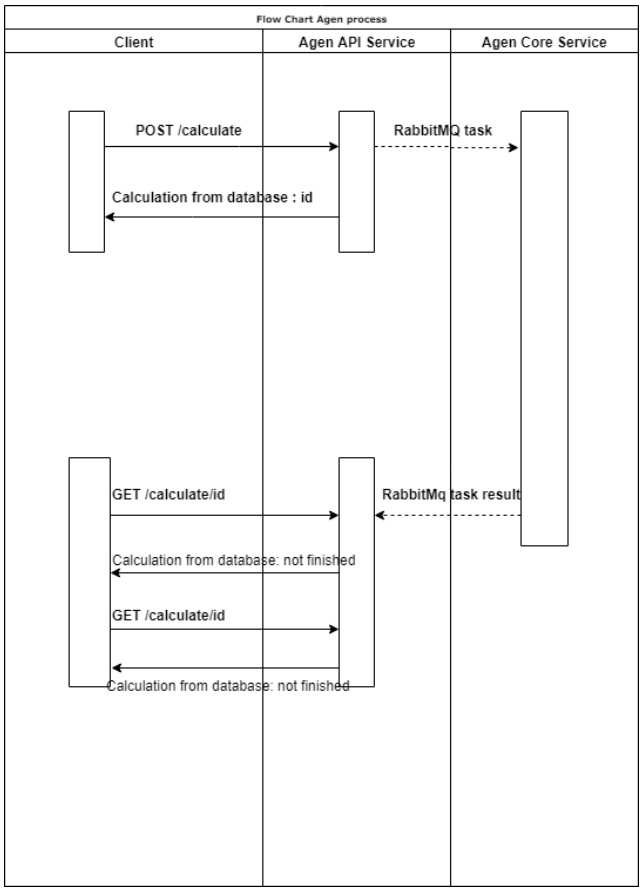


Diagrama pentru Comunicare intre „calculation-api-service” si „calculation-core-service”

3 Descrierea procesului

Procesul începe prin accesarea endpoint-ului aplicației si apelarea metodei POST (in client apăsare butonului după ce imaginea a fost încărcata). Un mesaj va fi trimis, prin intermediul cozilor, serviciului central pentru a primi o estimare iar cererea va fi salvata in baza de date si se va furniza imediat clientului un obiect JSON unde câmpul „id” va avea valoarea unui id generat de baza de date, in urma salvării cererii. In momentul in care procesul de estimarea a fost finalizat, rezultatul va fi trimis, prin intermediul cozilor, serviciului apelant, care îl va salva in baza de date. In acest timp clientul poate interoga baza de date pentru a verifica stadiul cererii prin apelarea metodei GET împreuna cu valoarea câmpului „id”. In momentul in care cererea va avea stadiul „FINISHED” sau „ERROR” procesul a luat sfârșit.



Process Flow chart

4 Scalabilitate si extensibilitate

Folosind o arhitectura de microservicii, serviciile aplicației „Agen” pot fi lansate separat in containere/mașini individuale. Pentru fiecare serviciu se pot adaugă instanțe noi in funcție de trafic si solicitare, nemaifiind nevoie de a adaugă instanțe ale întregii aplicații, cum ar fi in cazul aplicațiilor monolitice. In momentul de fața aplicația „Agen” folosește Docker pentru sistemul de deploy, facilitând astfel crearea de containere/replici la cerere. Gradul de extensibilitate este ridicat datorita implementării arhitecturii de microservicii unde noile funcționalități pot fi implementate prin adăugarea de noi servicii.

5 Direcții viitoare

In acest capitol sunt prezentate unele idei de îmbunătățire a aplicației „Agen”.

5.1 Migrarea Bazei de date

Aplicația „Agen” folosește baza de date H2 pentru stocarea datelor. Pentru a îmbunătăți scalabilitatea aplicației, este necesara migrarea la o baza de date care sa suporte elemente operaționale cheie, cum ar fi „data sharding”. In acest context este necesara si implementarea unei politici de load balancing a serviciilor aplicației.

5.2 Securizarea aplicației

Aplicația „Agen” nu folosește nici o forma de verificare a identității utilizatorului. Este necesara astfel securizarea aplicației prin crearea unui serviciu de autentificare folosind protocolul OAuth 2.0. De asemenea, comunicarea cu interfața furnizata de aplicație trebuie securizata prin adăugarea unui api key la fiecare request si validat de backend.

5.3 Creșterea acurateței

Acuratețea cu care aplicația „Agen” clasifica o persoana într-o categorie de vârsta si gen poate fi îmbunătățita prin implementarea a diverse metode de optimizare a rețelelor neuronale, cum ar fi îmbunătățirea setului de date, modificarea arhitecturii rețelei neuronale sau experimentarea cu diferite valori ale parametrilor.

5.4 Trecerea spre video

O alta direct spre care ar putea tinde aplicația „Agen” ar fi recunoașterea vârstei si genului unei persoane dintr-o înregistrare video, in timp real. Pentru a implementa aceasta

funcționalitate este necesara creșterea acurateței si scăderea timpului necesar unei calculări, asta însemnând eliminarea stocării in baza de date si a timpului necesar comunicării intre servicii. Acest lucru ar însemna ca funcționalitatea de recunoaștere a vârstei si genului sa fie mutata in partea de client a aplicației.

5.5 Configurare dinamica

O alta funcționalitate esențiala ar fi configurarea dinamica a retelelor folosite pentru recunoașterea genului si a vârstei sau posibilitatea de a alege si a afișa fișierul de ponderi in funcție de scorul obținut.

Contribuții personale

Principalele caracteristici ale aplicației „Agen” in ceea ce privește dezvoltarea sunt:

1. Aplicația „Agen” expune un API REST-ful definit prin doua endpoint-uri implementând metodele POST si GET prin care utilizatorul va putea testa funcționalitatea de recunoaștere a vârstei si a genului unei persoane dintr-o imagine primita ca data de input ce conține fața acesteia.
2. Proiectarea unor rețele neuronale convoluționale pentru recunoașterea vârstei si a genului prin intermediul librăriei Keras ce folosește ca implementare TensorFlow, permițând astfel antrenarea rețelelor folosind placa video cu ajutorul tehnologiei CUDA pentru a reduce considerabil timpul rulării.
3. Dezvoltarea unui client care sa consume API REST-ful expus de aplicația „Agen”.
4. Implementarea unei arhitecturi de microservicii pentru a creste gradul de scalabilitate si extensibilitate al aplicatiei.
5. Implementarea procesuli de deploy folosind Docker pentru a creste gradul de scalabilitatea al aplicatiei.

Concluzii

Domeniul științei datelor a evoluat considerabil in ultimele decenii, tehnologia devenind din ce in ce mai folosita in societatea moderna. Descoperirile din acest domeniu au dus la automatizarea proceselor existente din diverse domenii.

Aplicația „Agen” este o încercare de rezolvare a acestei probleme folosind rețele neuronale convoluționale pentru clasificarea într-o categorie de vârsta si pentru recunoașterea genului a unei persoane folosind fața acesteia, extrase din imaginea furnizata ca date de intrare. Rețeaua neuronala folosita in aplicația „Agen” a fost dezvoltata folosind librăria Keras iar extragerea feței unei persoane dintr-o imagine este realizata cu ajutorul librăriei OpenCv.

Aplicația „Agen” implementează o arhitectura de microservicii cu scopul de a creste gradul de scalabilitate si extensibilitate. Pentru fiecare serviciu se pot adaugă instanțe noi in funcție de trafic si solicitare, nemaifiind nevoie de a adaugă instanțe ale întregii aplicații, cum ar fi in cazul aplicațiilor monolitice. In momentul de fața aplicația „Agen” nu suporta aceasta funcționalitate.

Acuratețea cu care aplicația „Agen” clasifica o persoana într-o categorie de vârsta si gen poate, deși este relativ scăzută, aceasta poate fi îmbunătățita prin implementarea a diverse metode de optimizare a rețelelor neuronale, cum ar fi îmbunătățirea setului de date, modificarea arhitecturii rețelei neuronale sau experimentarea cu diferite valori ale parametrilor.

Histogramele si graficele din aceasta lucrare au fost generate prin intermediu librăriei Keras si al librăriei matplotlib in limbajul de programare Python.

Bibliografie

[1] *https://keras.io/*

[2] Joseph Howse (April 2013) - *„OpenCV Computer Vision with Python”*

[3] Michael Nielsen – *„Neural Networks and Deep Learning”*

http://neuralnetworksanddeeplearning.com/index.html

[4] *http://cs231n.github.io/*

[5] *https://data.vision.ee.ethz.ch/cvl/rrothe/imdb-wiki/*

[6] Rasmus Rothe, Radu Timofte, Luc Van Gool - *„DEX: Deep EXpectation of apparent age from a single image”*

[7] *https://towardsdatascience.com/predict-age-and-gender-using-convolutional-neural-network-and-opencv-fd90390e3ce6*

*[8] Age and Gender Estimation of Unfiltered Faces Eran Eidinger, Roee Enbar, Tal Hassner**

https://www.openu.ac.il/home/hassner/Adience/EidingerEnbarHassner_tifs.pdf

[9] *https://bernhardwenzel.com/articles/tutorial-build-a-message-driven-microservice-application/*

[10] *https://dzone.com/articles/buiding-microservice-using-springboot-and-docker*

[11] *https://www.rabbitmq.com/getstarted.html*

[12] *https://spring.io/guides/gs/rest-service/*

[13] Chris Solomon, Toby Breckon – *„Fundamentals of Digital Image Processing A Practical Approach with Examples in Matlab”*

[14] *Diagramele si flow chart-urile au fost facute pe:*

https://www.draw.io/

[15] *https://docs.docker.com/get-started/*

[16] *https://angular.io/docs*

[17] *https://mdbootstrap.com/docs/angular/*

[18] *https://bernhardwenzel.com/articles/tutorial-build-a-message-driven-microservice-application/*

[19] *https://github.com/shamangary/Keras-real-time-age-gender-estimation-demo*