

UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAȘI  
FACULTATEA DE INFORMATICĂ



LUCRARE DE LICENȚĂ

**AGEN**

*Recunoașterea vârstei și 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 Licență  
Lector dr. Anca Ignat  
8 Februarie 2019

### **DECLARAȚIE privind originalitatea conținutului lucrării de licență**

Subsemnatul Dorneanu Cristian, domiciliul în Iasi, născut la data de 12 Martie 1995, identificat prin CNP 1950312226711, absolvent al Universității „Alexandru Ioan Cuza” din Iași, Facultatea de Informatica, promoția 2014-2017, declar pe propria răspundere, cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art.143 al. 4 si 5 referitoare la plagiat, că lucrarea de licență cu titlul: „Agen, recunoașterea vârstei si a genului cu rețele neuronale convoluționale”, elaborată sub îndrumarea dnei Lector dr. Anca Ignat, pe care urmează să o susțin în fața comisiei este originală, îmi aparține și îmi asum conținutul său în întregime.

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

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

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

<b>Introducere .....</b>	<b>5</b>
1    Motivație.....	5
2    Obiective .....	5
<b>Descrierea unei rețele neuronale convolutive.....</b>	<b>6</b>
1    Stratul convolutional.....	6
2    Stratul de pooling.....	6
3    Stratul complet conectat .....	7
<b>Descrierea rețelei aplicației „Agen” .....</b>	<b>8</b>
1    Obținerea și prelucrarea datelor .....	8
2    Proiectarea rețelei pentru recunoașterea vârstei .....	9
3    Proiectarea rețelei pentru recunoașterea genului .....	9
4    Antrenarea rețelei.....	11
5    Salvarea modelului.....	11
6    Analiza rezultatelor .....	11
7    Evaluarea.....	12
<b>Aplicația „Agen” .....</b>	<b>13</b>
1    Arhitectura și componentele aplicație.....	13
1.1        Componenta „config” .....	13
1.2        Componenta „calculation-api-service” .....	14
1.3        Componenta „calculation-core-service” .....	14
1.4        Componenta „ageandgender-app” .....	15
2    Comunicarea între componente .....	16
3    Descrierea procesului .....	17
4    Scalabilitate și extensibilitate .....	18
5    Direcții viitoare .....	18
5.1        Migrarea Bazei de date .....	18

5.2	Securizarea aplicației .....	18
5.3	Creșterea acurateței .....	18
5.4	Trecerea spre video .....	19
<b>Contribuții personale.....</b>		<b>20</b>
<b>Concluzii.....</b>		<b>21</b>

# Introducere

## 1 Motivație

Domeniul științei datelor a evoluat considerabil în ultimele decenii, tehnologia devenind din ce în ce mai folosită în societatea modernă. Descoperirile din acest domeniu au dus la automatizarea proceselor existente din diverse domenii, cum ar fi diversele tehnologii furnizate în industria de autoturisme (autopilot adaptiv) cu scopul de a crește confortul și siguranța. Un alt exemplu este reprezentat de un sistem de recunoaștere facială folosit pentru a urmări prezența în cadrul orelor de curs. Descoperirile din acest domeniu și inovațiile care au rezultat datorită acestora, mi-au stârnit curiozitatea, reprezentând motivul alegerii temei pentru această lucrare.

## 2 Obiective

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

Aplicația „Agen” este o încercare de rezolvare a acestei probleme folosind rețele neuronale convoluționale pentru clasificarea într-o categorie de vârstă și pentru recunoașterea genului a unei persoane folosind fața acesteia, extrase din imaginea furnizată ca date de intrare. Rețeaua neuronală folosită în aplicația „Agen” a fost dezvoltată folosind librăria Keras iar extragerea feței unei persoane dintr-o imagine este realizată cu ajutorul librăriei OpenCv [1, 2]. Aplicația „Agen” implementează o arhitectură de microservicii, limbajele folosite în serviciile sale fiind Java și Python. Comunicarea cu aplicația „Agen” se realizează prin intermediul unui API REST-ful. Pentru ca rețeaua neuronală să fie capabilă să clasifice o față într-o categorie de vârstă și gen, aceasta trebuie antrenată, folosind date reale.

În următoarele capitole este descrisă problema în detaliu, implementare propusă de aplicația „Agen” și arhitectura acesteia.

## **Descrierea unei rețele neuronale convolutive**

O rețea neuronală este o paradigma de programare, inspirată din natură, care are ca scop învățarea din date observaționale. Diferența între o rețea neuronală și o rețea neuronală convolventă constă în arhitectura acestora, cea din urmă fiind specializată în lucrul cu imagini. Datele de intrare pentru o rețea neuronală convolventă sunt, în cele mai multe cazuri, imagini, volume cu o lățime, înălțime și o adâncime care reprezintă numărul canalelor de culoare. Arhitectura unei rețele neuronale convolutive este alcătuită din mai multe straturi, dintre care cele mai importante sunt: stratul convolvent, stratul de pooling și stratul complet conectat [3].

### **1 Stratul convolvent**

Stratul convolvent este blocul central al unei rețele neuronale convolutive. Fiecare neuron din acest strat este conectat la o regiune din volumul de intrare, dimensiunea acestei regiuni fiind determinată de valoarea hiperparametrului cu numele de câmp receptiv. Numărul de neuroni din volumul rezultat în urma stratului convolvent este determinat de numărul de filtre, dimensiunea filtrului (câmp receptiv) și pasul cu care filtrul este glisat peste volumul de intrare. În cazul în care numărul de parametri (ponderi), dintr-o rețea neuronală convolventă, este prea mare, se poate aplica o tehnică numită împărțirea parametrilor care presupune folosirea a aceleași ponderi pentru fiecare strat de adâncime din volumul primit ca date de intrare [4].

### **2 Stratul de pooling**

O arhitectură de rețele neuronale convolutive este alcătuită din mai multe straturi convolutive, conectate în serie, între care pot apărea straturi de pooling. Acest strat are ca scop reducerea spațială a volumului primit ca date de intrare, a numărului de parametri și a complexității computaționale. În acest strat este necesară valoarea a doi hiperparametri, dimensiunea filtrului și 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 constă în 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 neuronală, în stratul complet conectat fiecare neuron este conectat la fiecare neuron din volumul primit ca date de intrare.

## Descrierea rețelei aplicației „Agen”

Problema clasificării vârstei și a genului unei persoane a fost împărțită în două, clasificarea vârstei, respectiv clasificarea genului. Din acest motiv au fost proiectate două arhitecturi de rețele neuronale convoluționale diferite. Implementarea acestor rețele a fost realizată prin intermediul librăriei Keras. În proiectarea acestor rețele au fost luate în considerare următoarele aspecte:

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

În continuare se va descrie fiecare aspect în detaliu ce face parte din componenta centrală a aplicației „Agen”.

### 1 Obținerea și prelucrarea datelor

Setul de date pentru antrenarea rețelei este format din imagini extrase din baza de date IMDB și Wikipedia [5]. Antrenarea rețelei se face în mod supervizat, ceea ce implică faptul că fiecare imagine va avea atașată vârsta și genul persoanei respective. Setul de date constă în aproximativ 90 000 imagini. Înainte de a antrena rețeaua, se parcurge setul de date și se extrage fața din fiecare imagine, dacă există, și se suprascrivește imaginea originală cu fața detectată. Detectarea feței se realizează folosind librăria OpenCv. În cazul apariției de fețe multiple într-o imagine, se alege doar fața cu dimensiunea cea mai mare, cea mai apropiată fața. Vârsta și genul asociate feței din fiecare imagine sunt stocate într-un fișier de tip „.mat”.

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

## 2 Proiectarea rețelei pentru recunoașterea vârstei

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

Pentru stratul de ieșire, au fost selectate 10 categorii de vârstă 1-10, 10-15, 15-21, 21-28, 28-35, 35-42, 42-50, 50-65, 65-80, 80-100. Această configurație a fost aleasă pentru o acuratețe mai mare, scopul aplicației nefiind recunoașterea exactă a vârstei ci clasificarea într-o categorie. Selecția a 10 categorii de vârstă ar fi rezultat într-o acuratețe mai scăzută și o complexitate de spațiu și timp mai mare, lucru care nu ar fi fost suportat de infrastructura unde a fost dezvoltată aplicația „Agen” [6].

Modelul este alcătuit din patru straturi convoluționale cu numărul de filtre egal cu 64, 64, 128, 128, în această ordine, cu o lățime și lungime a filtrului egală cu 3, un pas egal cu 1 și o umplutură a volumului de input cu valoarea 0 astfel încât dimensiunea în lungime și lățime a stratului convoluțional să fie egală cu cea a volumului de intrare. Funcția de activare folosită pentru aceste straturi este softmax. Între aceste straturi convoluționale se află câte un strat de pooling cu o lungime și lățime a filtrului egală cu 2 și un pas egal cu 2, cu scopul de a reduce dimensiunea volumului după fiecare convoluție cu 75%.

În continuarea acestor straturi se află un strat de normalizare de tip batch și un strat complet conectat având 11 noduri reprezentând clasele rezultate. Structura rețelei poate fi schițată în următorul mod:

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

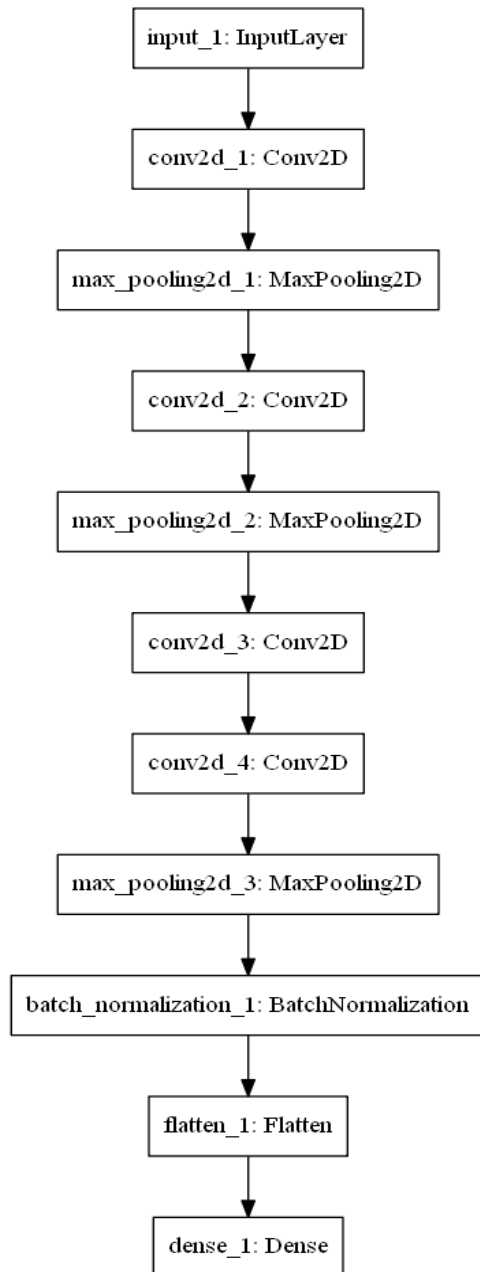
## 3 Proiectarea rețelei pentru recunoașterea genului

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

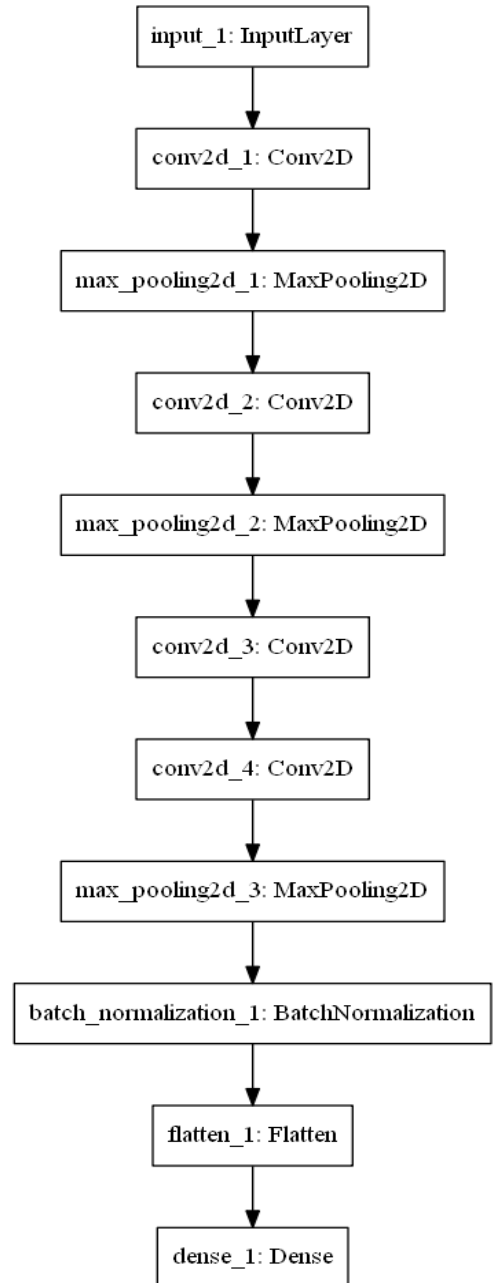
Pentru stratul de ieșire, au fost selectate 2 categorii de gen cu valoare 1 pentru masculin respectiv 0 pentru feminin. Modelul este alcătuit din patru straturi convoluționale cu numărul de filtre egal cu 64, 64, 128, 128, în această ordine, cu o lățime și lungime a filtrului egală cu 3, un pas egal cu 1 și o umplutură a volumului de input cu valoarea 0. Funcția de activare folosită pentru aceste straturi este softmax. Între aceste straturi convoluționale se află câte un strat de pooling cu o lungime și lățime a filtrului egală cu 2 și un pas egal cu 2.

În continuarea acestor straturi se afla un strat de normalizare de tip batch și un strat complet conectat având 2 noduri reprezentând clasele rezultate. Structura rețelei poate fi schițată în următorul mod:

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



Structura rețelei neuronale convoluționale pentru recunoașterea vârstei



Structura rețelei neuronale convoluționale pentru recunoașterea genului

## 4 Antrenarea rețelei

Aplicația „Agen” folosește o rețea neuronală convolutională antrenată într-o manieră supervizată ceea ce presupune ca fiecare față din setul de date să conțină vârsta, în acest caz categoria de vârstă, și genul real transformate sub forma unor vectori de o dimensiune astfel încât să poată fi comparat cu rezultatul produs de rețea.

Ca și funcție de optimizare este folosită SGD (Stochastic Gradient Descent) cu o rată de învățare inițială egală cu 0,1 un momentum egal cu 0,9 și o descompunere de 0,005. La fiecare 4 epoci, rata de învățare va fi înmulțită cu un factor de 0,1. Ca și funcție de pierdere este folosită Cross Entropy. Dimensiunea lotului aleasă este de 32 din motivul insuficienței de resurse computaționale. Rețeaua este antrenată pe parcursul a 20 de epoci iar ponderile rețelei sunt salvate la terminarea unei epoci în cazul în care valoarea de pierdere a scăzut [7].

Folosind TensorFlow ca implementare a bibliotecii Keras, rețelele au putut fi antrenate folosind placa video Nvidia cu ajutorul tehnologiei CUDA, sporind astfel eficiența în timp, fiind nevoie de aproximativ 50 minute pentru a antrena o rețea pe parcursul a 20 de epoci.

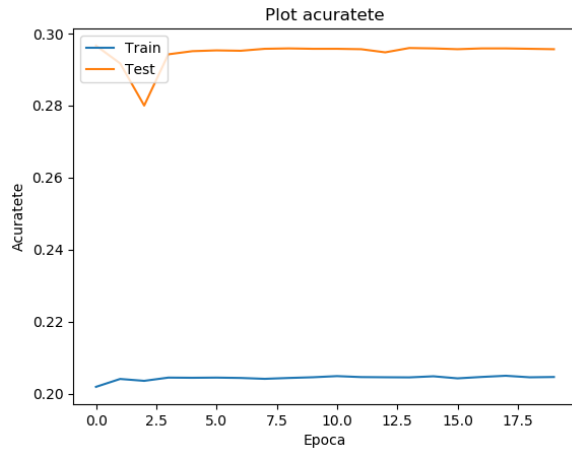
## 5 Salvarea modelului

Pentru a nu antrena rețeaua de fiecare dată când aplicația „Agen” este pornită (proces costisitor în timp), rețeaua este antrenată doar în cazul în care nu există nici un fișier cu ponderile rezultate dintr-o antrenare anterioară a rețelei. În cazul în care există acest fișier, rețeaua va încărca ponderile salvate, fără a mai fi nevoie de a o antrena din nou.

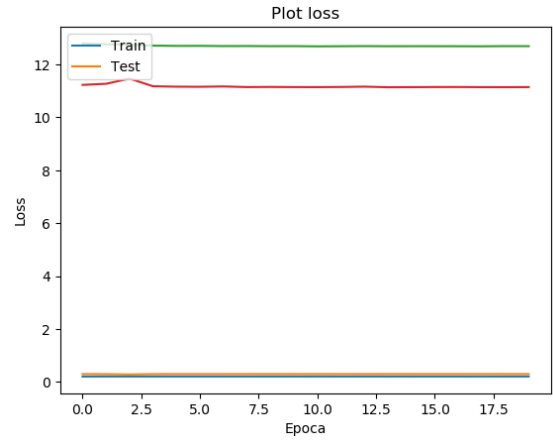
## 6 Analiza rezultatelor

După antrenarea rețelei pentru clasificarea vârstei s-a obținut, pentru datele de antrenament o acuratețe de 22,6% și o valoare a pierderii egală cu 11,14, iar pentru datele de validare, o acuratețe de 29,08% și o valoare a pierderii egală cu 12,6.

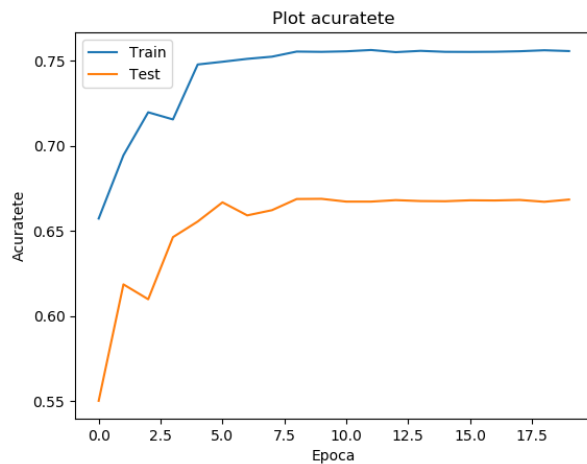
După antrenarea rețelei pentru clasificarea genului s-a obținut, pentru datele de antrenament o acuratețe de 76,8% și o valoare a pierderii egală cu 0,98, iar pentru datele de validare, o acuratețe de 66,5% și o valoare a pierderii egală cu 0,68.



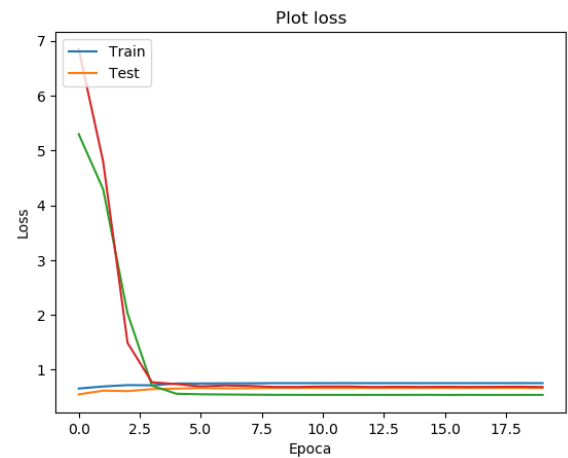
*Plot pentru acuratatea rețelei neuronale în recunoașterea vârstei*



*Plot pentru pierderea rețelei neuronale în recunoașterea vârstei*



*Plot pentru acuratatea rețelei neuronale în recunoașterea genului*



*Plot pentru pierderea rețelei neuronale în recunoașterea genului*

## 7 Evaluarea

Pentru evaluarea unei noi imagini, mai întâi se extrage fata din imagine, apoi fata extrasa este transmisă ca dată de intrare către rețea iar rezultatul este decodificat la clasa de vârstă și gen la care aparține persoana din imagine.

## Aplicația „Agen”

Aplicația implementează o arhitectura de microservicii. Deși simplitatea aplicației și numărul redus de servicii nu justifică adoptarea acestei arhitecturi, în acest stadiu, aceasta arhitectura a fost aleasă pentru posibilitatea de scalabilitate pe care o oferă și nivelul scăzut de cuplare ce facilitează adăugarea de noi servicii și funcționalități [8].

Se identifică două servicii principale ale acestei aplicații. Un serviciu implementat folosind Java și Spring Boot ce are drept rol comunicarea cu clientul prin expunerea unui API REST-ful și salvarea estimări (sub numele de calculare în aplicație) în baza de date. Serviciul central aplicației este implementat folosind limbajul Python și are ca rol clasificarea într-o clasă de vârstă și gen a unei persoane pe baza unei imagini primite ca date de intrare. Tot aici are loc procurarea și prelucrarea datelor de intrare și antrenarea rețelei [9].

În construirea aplicației următoarele aspecte au fost luate în calcul:

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

## 1 Arhitectura și componentele aplicație

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

Aplicația „Agen” este structurată după arhitectura de microservicii având următoarele servicii/componente: „config”, „eureka-server”, „calculation-api-service”, „calculation-core-service”, „ageandgender-app”.

### 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 această componentă, din acest motiv cerința pentru această componentă este disponibilitate mare și toleranță la erori. Această componentă este implementată folosind limbajul Java și 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 aplicatia „Agen”. Se foloseste o baza de date H2 pentru stocarea de cereri/estimări (calculări). Aplicatia neavând un serviciu de autentificare sau functionalitate de corelare a cererilor cu utilizatorul, nu este nevoie folosirea unor baze de date specializate cum ar fi MySQL. Sunt expuse doua endpoint-uri, ce suporta metodele POST pentru initierea procesului de estimare a vârstei si genului pe baza unei imagini primite ca data de intrare si salvarea acestuia 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 si nu furnizarea 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 status-ul calculării va fi in eroare sau terminat, caz in care va fi furnizat un obiect JSON cu estimările de vârsta si gen.

## 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 fata 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 partea in care se copie local arhivele.

Comunicare cu serviciul „calculation-api-service” este realizata prin intermediul unui broker de mesaje numit RabbitMQ [10]. După salvarea sau încărcarea ponderilor, serviciul va fi pus in stare de așteptare, pana când va primi un mesaj (cerere de calculare). După terminarea calculării, rezultatul va fi publicat, iar in serviciul anterior se va executa stocarea/modificarea in baza de date.



### 1.3 Componenta „ageandgender-app”

Aplicația „Agen” conține și un client ce va consuma REST API-ul produs de serviciul „calculation-api-service” [11]. În urma calculării, rezultatul va fi afișat împreună cu imaginea de intrare și fața detectată.

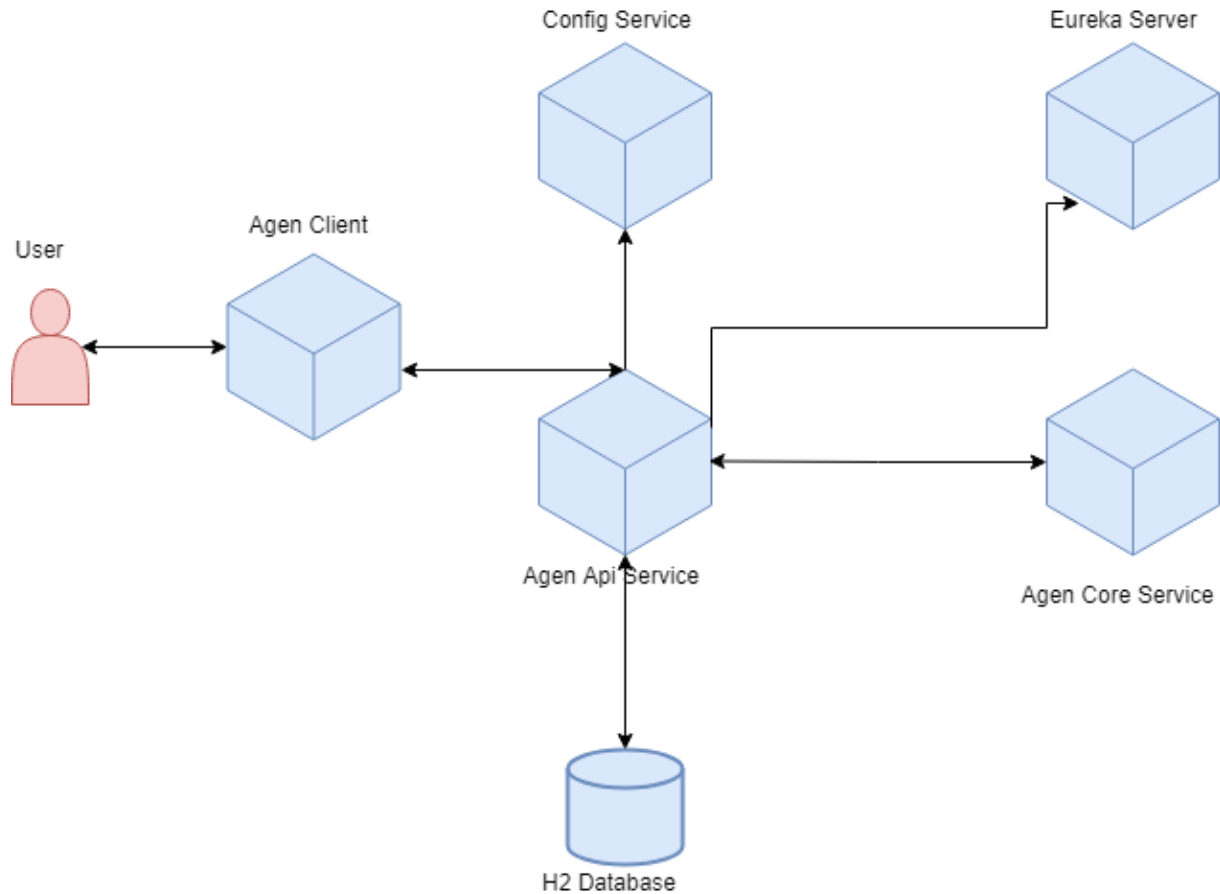
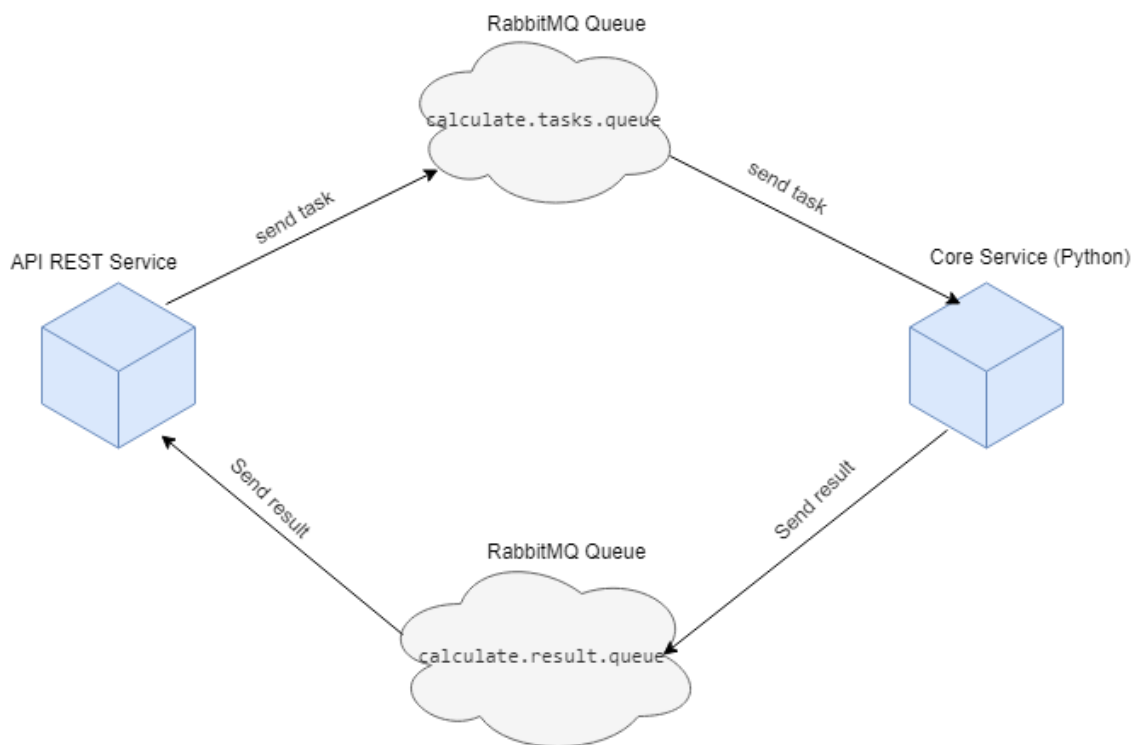


Diagrama a arhitecturii aplicației Agen

## 2 Comunicarea intre componente

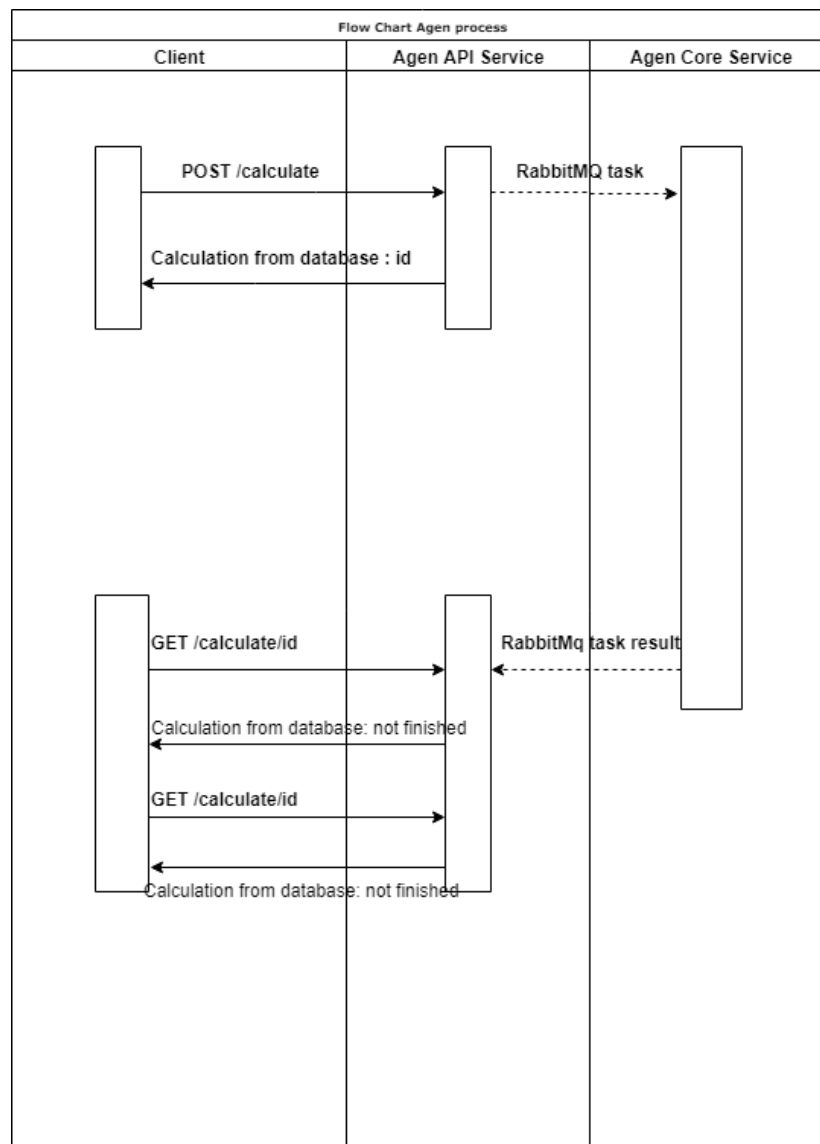
Comunicare intre serviciile „calculation-api-service” si „calculation-core-service” este realizata prin intermediul unui broker de mesaje numit RabbitMQ. Sunt definite doua cozi pentru schimbul de mesaje, „calculate.tasks.queue” si „calculate.result.queue”. In prima coada, procesul de calculare este 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 și apelarea metodei POST (în client apăsare butonului după ce imaginea a fost încărcată). Un mesaj va fi trimis, prin intermediul cozilor, serviciului central pentru a primi o estimare iar cererea va fi salvată în baza de date și se va furniza imediat clientului un obiect JSON unde câmpul „id” are valoarea calculării salvate în baza de date. În momentul în care estimarea s-a terminat, rezultatul este trimis, prin intermediul cozilor, serviciului apelant, care îl va salva în baza de date. În acest timp clientul poate interoga baza de date pentru a verifica stadiul cererii prin apelarea metodei GET împreună cu valoarea câmpului „id”. În momentul în 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 în funcție de trafic si solicitare, nemaifiind nevoie de a adaugă instanțe ale întregii aplicații, cum ar fi in cazul aplicațiilor monolitice. In momentul de fata aplicația „Agen” nu suporta aceasta funcționalitate. 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 aceasta secțiune 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 vârstă 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 și genului unei persoane dintr-o înregistrare video, în timp real. Pentru a implementa această funcționalitate este necesară creșterea acurateții și scăderea timpului necesar unei calculări, asta însemnând eliminarea stocării în baza de date și a timpului necesar comunicării între servicii. Acest lucru ar însemna ca funcționalitatea de recunoaștere a vârstei și genului să fie mutată în partea de client a aplicației.

## Contribuții personale

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

1. Aplicația „Agen” expune un API REST-ful definit prin două endpoint-uri implementând metodele POST și GET prin care utilizatorul va putea testa funcționalitatea de recunoaștere a vârstei și a genului unei persoane dintr-o imagine primită ca date de input ce conține fața acesteia.
2. Proiectarea unor rețele neuronale convoluționale pentru recunoașterea vârstei și 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 să consume API REST-ful expus de aplicația „Agen”.
4. Implementarea unei arhitecturi de microservicii pentru a crește gradul de scalabilitate și extensibilitate al aplicației.

## Concluzii

Domeniul științei datelor a evoluat considerabil în ultimele decenii, tehnologia devenind din ce în ce mai folosită în societatea modernă. 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 vârstă și pentru recunoașterea genului a unei persoane folosind fața acesteia, extrase din imaginea furnizată ca date de intrare. Rețeaua neuronală folosită în aplicația „Agen” a fost dezvoltată folosind librăria Keras iar extragerea feței unei persoane dintr-o imagine este realizată cu ajutorul librăriei OpenCv.

Aplicația „Agen” implementează o arhitectură de microservicii cu scopul de a crește gradul de scalabilitate și extensibilitate. Pentru fiecare serviciu se pot adăuga instanțe noi în funcție de trafic și solicitare, nemaifiind nevoie de a adăuga instanțe ale întregii aplicații, cum ar fi în cazul aplicațiilor monolitice. În momentul de față aplicația „Agen” nu suportă această funcționalitate.

Acuratețea cu care aplicația „Agen” clasifică o persoană într-o categorie de vârstă și gen poate, deși este relativ scăzută, aceasta poate fi îmbunătățită 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 și graficele din această lucrare au fost generate prin intermediul librăriei Keras și al librăriei matplotlib în 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] <https://bernhardwenzel.com/articles/tutorial-build-a-message-driven-microservice-application/>
- [9] <https://dzone.com/articles/buiding-microservice-using-springboot-and-docker>
- [10] <https://www.rabbitmq.com/getstarted.html>
- [11] <https://spring.io/guides/gs/rest-service/>
- [12] Chris Solomon, Toby Breckon – „*Fundamentals of Digital Image Processing A Practical Approach with Examples in Matlab*”
- [13] *Diagramele si flow chart-ul au fost facute pe:*  
<https://www.draw.io/>