

UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI
FACULTATEA DE INFORMATICĂ



LUCRARE DE LICENȚĂ

Animal Image Recognition

propusă de

Șerban-Mihai Botez

Sesiunea: *Iulie, 2019*

Coordonator științific

Lect. dr. Anca Ignat

UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAȘI
FACULTATEA DE INFORMATICĂ

Animal Image recognition

Șerban-Mihai Botez

Sesiunea: *Iulie, 2019*

Coordonator științific

Lect. dr. Anca Ignat

Avizat,

Îndrumător Lucrare de Licență

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

_____elaborată sub îndrumarea dl. / d-na
_____, pe care urmează să o susțină 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.

Data azi,

Semnătură student

DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „*Animal Image Recognition*”, 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, 27.06.2019

Absolvent Șerban-Mihai Botez

(semnătura în original)

ACORD PRIVIND PROPRIETATEA DREPTULUI DE AUTOR

Facultatea de Informatică este de acord ca drepturile de autor asupra programelor-calculator, în format executabil și sursă, să aparțină autorului prezentei lucrări,
Șerban-Mihai Botez.

Încheierea acestui acord este necesară din următoarele motive:

[Se explică de ce este necesar un acord, se descriu originile resurselor utilizate în realizarea produsului-program (personal, tehnologii, fonduri) și aportul adus de fiecare resursă.]

Iași, 27.06.2019

Decan *Adrian Iftene*

(semnătura în original)

Absolvent *Șerban-Mihai Botez*

(semnătura în original)

Cuprins

Cuprins	1
Motivație	3
Introducere	4
Contribuții	5
1. Descrierea problemei	6
1.1 Abordări Anterioare.....	7
2. Procesarea Imaginilor	8
2.1 Tipuri de imagini.....	8
2.2 Adâncimea culorilor și formatul imaginilor.....	9
2.3 Zgomotul în imagini.....	10
3. Clasificatori în cascadă	11
3.1 Algoritmul Viola&Jones	11
3.2 Antrenarea clasificatorilor și detecția pe imagini .Error! Bookmark not defined.	
4. Rețele Neuronale	17
4.1 Rețele neuronale convoluționale.....	17
4.2 Arhitectura modelului pentru recunoașterea animalelor ...Error! Bookmark	

not defined.

5. Descrierea soluției.....	23
5.1 Arhitectura proiectului	19
5.2 Prezentare interfață, antrenare, statistici.....	20
6. Concluzii	23
Bibliografie.....	24

Motivație

Vederea Artificială (Computer Vision) este o ramura a inteligenței artificiale care se ocupă de procesarea imaginilor din lumea reală. Este compusă din tehnici low level de procesare a imaginilor, cum ar fi eliminarea zgomotului sau redimensionarea imaginilor și tehnici high level (recunoașterea modelelor și clasificarea lor). Recunoașterea facială este cel mai folosit domeniu al vederii artificiale, fiind necesară în multe sisteme de securitate, controlul accesului și marketing.

Recunoașterea facială pe animale este în mare parte folosită pentru aplicații de găsimă a animalelor pierdute, în clasificarea automatizată a speciilor dar și în preluarea de imagini sau videoclipuri ale unor rase specifice din mediul lor natural, cu interacțiune umană cat mai limitată.

Chiar dacă exista deja multe tehnici si implementari de recunoaștere facială, nu s-a ajuns la un sistem cu o acuratețe de 100% deoarece majoritatea imaginilor sunt preluate din lumea reală. Astfel, există posibilitatea continuă de a îmbunătăți abordările folosite, lucru care atrage tot mai mulți cercetatori si pasionați spre această ramură a inteligenței artificiale.

Introducere

Această lucrare are ca scop construirea și prezentarea unei aplicații prin intermediul căreia se pot detecta și recunoaște fețe și corpuri de animale în imaginile furnizate ca input, folosind clasificatori și modele deja antrenate. De asemenea, aplicația va permite utilizatorilor să își antreneze proprii clasificatori sau modele, în funcție de animalul ce se dorește a fi găsit.

În continuare se vor prezenta capitolele lucrării, cât și o descriere în mare pentru fiecare capitol:

1. **Descrierea problemei.** În acest capitol se prezintă descrierea problemei abordate de către această lucrare, dar și ideile principale ale problemei. Unele abordări și idei ale problemei inițiale nu au putut face parte din soluția finală. Ele vor fi prezentate în subcapitolul „Abordări anterioare”, împreună cu motivele pentru care s-au decis să se urmeze alte cai în rezolvarea problemei.
2. **Procesarea imaginilor.** În acest capitol se prezintă fundamentele procesării imaginilor și importanța acestora în obținerea unui model bine antrenat.
3. **Clasificatori în cascadă.** Acest capitol are rolul de a prezenta algoritmi folosiți pentru metoda cu clasificatori în cascadă, cât și cum funcționează de fapt această abordare.
4. **Rețele neuronale.** Acest capitol conține o scurtă introducere în conceptul de rețele neuronale, ce tip de rețele neuronale vor fi folosite pentru problema prezentată în lucrare cât și detalii despre procesul de crearea a unui model.
5. **Descrierea soluției.** În acest capitol se descrie arhitectura proiectului, împreună cu beneficiile fiecărei abordări alese cât și parametrii folosiți pentru antrenare, însoțiți de către date statistice. În acest capitol sunt prezentate și câteva tehnologii folosite.
6. **Concluzii.** Acest ultim capitol aduce rezultatele tuturor abordărilor folosite, împreună cu beneficiile fiecărei abordări și opiniile personale legate de rezultatele lucrării. De asemenea, sunt prezentate și posibile direcții de viitor a proiectului și a problemei.

Contribuții

În această lucrare se dorește să se implementeze, antreneze și evalueze un model care este capabil să prezică animalele dintr-o imagine primită ca input. Există deja tehnici de depistare a anumitor obiecte dintr-o imagine, dar acestea funcționează cu o acuratețe bună doar pe obiecte care nu se mișcă foarte mult în diferite imagini (de exemplu steagul unei țări sau logo-ul unei companii), fie necesită o bază de cunoștințe în procesarea de imagini foarte amănunțită pentru a putea fi folosite de un public mai larg.

Elementul principal al acestei lucrări este reprezentat de crearea unei aplicații cu o interfață prietenoasă, care să poată fi folosită de persoane fără cunoștințe avansate în domeniu, dar și de persoane implicate în inteligența artificială și învățare automată. De asemenea, se încearcă ca modelul final să aibă o acuratețe cât mai ridicată împreună cu un timp rezonabil de antrenare pe imagini cât mai diversificate, pentru a nu se ignora partea practică a proiectului.

Deoarece există enorm de multe cazuri în care un singur animal poate fi găsit într-o imagine, combinat cu multitudinea speciilor de animale ce prezintă trăsături diferite, modelul final poate fi mereu îmbunătățit prin folosirea de tehnologii diferite, combinarea diferitelor moduri de extragere a trăsăturilor sau procesarea imaginilor de antrenare.

Capitolul 1

Descrierea problemei

Problema care se dorește a se rezolva este recunoașterea animalelor dintr-o imagine furnizată de către utilizator, folosind un model deja antrenat pe animale specifice. Pentru sistemul vizual uman, această problemă nu prezintă dificultăți atât timp cât se cunosc anumite trăsături ale speciei prezente în imagine. Pentru un computer se încearca o abordare asemanătoare, bazându-se pe trăsături deja cunoscute luate din alte imagini unde se știe deja animalul prezent în imagine. Deoarece imaginile sunt stocate ca valori binare pentru fiecare pixel în parte, un computer are nevoie de o serie complexă de instrucțiuni pentru a recunoaște trăsăturile dintr-o imagine și a putea spune ce animal se găsește în acea imagine.

Metodele folosite sunt clasificatoare în cascadă și rețele neuronale pentru antrenare și detecție, folosind un set de imagini deja clasificate pentru obținerea trăsăturilor.

1.1 Abordări anterioare

În rezolvarea problemei s-au încercat mai multe tehnici pentru crearea modelului final, acestea având diferite motive pentru care nu au putut fi folosite în soluția finală. Cateva dintre aceste tehnici sunt: corner detectors, local invariant descriptors.

SIFT (Scale-Invariant Feature Transform) este un algoritm folosit pentru detecția trăsăturilor în imagini. Aceste trăsături sunt extrase dintr-un set de imagini de referință și mai apoi sunt stocate într-o bază de date. Un plus al algoritmului este rezultatul pe care îl oferă chiar dacă imaginea a fost redimensionată, scalată sau i s-a schimbat luminozitatea. Motivul pentru care nu s-a optat ca să se urmeze această abordare este timpul mare pe care îl necesită algoritmul pentru a găsi trăsăturile. Un clasificator în cascadă are nevoie de mult timp pentru antrenare, dar după terminarea antrenării, procesul de detecție este foarte rapid. [1]

Detectoarele de colțuri sunt operații matematice care încearcă să găsească trăsături într-o imagine, indiferent de luminozitate sau alte transformări ale imaginii. Problema la detectoarele de colțuri este lipsa aplicării pe imaginile din mediul înconjurător, deoarece acestea nu sunt robuste și pot detecta același colț de mai multe ori sau chiar lăsa anumite erori la detecție să scape nerezolvate. [2]

Capitolul 2

Procesarea imaginilor

Înainte de a putea începe crearea unui model și antrenare acestuia, trebuie să ne alegem un set de imagini care vor consta din setul de date de antrenament. În general, acest set de date trebuie să conțină cât mai multe imagini, pentru a ca modelul nostru să obțină cât mai multe trăsături pentru specia care se dorește a se găsi. De asemenea, imaginile dintr-un anumit set vor conține doar o specie de animale, pentru ca modelul să poată fi antrenat doar pentru acea specie și pentru a se reduce procentul detecțiilor fals pozitive (de exemplu, într-o imagine se detectează un câine ca fiind o pisică, pentru că aceste două specii au multe trăsături în comun).

Setul de date de antrenament conține două seturi de imagini: setul de imagini pozitive și setul de imagini negative. Imaginile pozitive sunt imagini care conțin animalul care se dorește a fi găsit din diferite unghiuri, pentru a se obține o gamă cât mai largă de trăsături. Imaginile negative pot fi imagini de orice tip, dar trebuie obligatoriu să nu conțină animalul cerut. Este de preferat ca setul de imagini negative să conțină imagini din mediul înconjurător al animalului care trebuie găsit, pentru a se putea face cât mai bine diferența dintre trăsăturile animalului și fundalul imaginii.

Pentru ca antrenarea să se realizeze într-un timp rezonabil și pentru a se obține o acuratețe cât mai bună, se vor aplica diferite procesări pe imagini (redimensionare, decupare, transformare în alb-negru, eliminarea zgomotului). Următoarele subcapitole vor prezenta aceste concepte, pentru a se putea înțelege mai bine termenii și asocierile folosite în următoarele capitole.

2.1. Tipuri de imagini

Imaginile pot fi împărțite în două categorii, în funcție de metoda de stocare folosită: imagini vectoriale și imagini digitale (acestea mai sunt numite și imagii raster sau bitmap). Imaginile vectoriale sunt combinații de puncte care urmează mai multe funcții matematice. Deoarece majoritatea imaginilor vectoriale sunt de fapt linii trase între puncte, aceste imagini sunt independente de rezoluție și pot fi mărite sau micșorate fără pierderi de calitate.

Imaginile digitale (raster sau bitmap) sunt formate din pixeli așezați pe o grilă. Pixelul reprezintă cel mai mic element dintr-o imagine, fiecare pixel având cate o valoare într-un anumit interval, în funcție de tipul de imagine digitală care este reprezentată. Rezoluția unei imagini este determinată de numărul de pixeli pe lățime și înălțime. Atunci când o imagine este digitală este redimensionată, calitatea pixelilor scade deoarece rezoluția se schimbă și numărul de pixeli trebuie marit sau micșorat pentru a se ajunge la rezoluția dorită. În continuare se va lucra și discuta doar despre imagini digitale, deoarece este cel mai popular tip de reprezentare a imaginilor.

2.2. Adâncimea culorilor și formatul imaginilor

Numărul de culori diferite dintr-o imagine diferă în funcție de adâncimea culorilor sau numărul de biți pe un pixel. Mai jos se găsește un tabel ce conține câțiva biți și culorile asociate lor.

Bits per pixel	Number of colors
1 bpp	2 colors
2 bpp	4 colors
3 bpp	8 colors
4 bpp	16 colors
5 bpp	32 colors
6 bpp	64 colors
7 bpp	128 colors
8 bpp	256 colors
10 bpp	1024 colors
16 bpp	65536 colors
24 bpp	16777216 colors (16.7 million colors)
32 bpp	4294967296 colors (4294 million colors)

Sursă tabel : https://www.tutorialspoint.com/dip/concept_of_bits_per_pixel.htm

De obicei, imaginile alb-negru (grayscale) conțin 8 biți pe un pixel, ceea ce înseamnă 256 culori diferite. Imaginile colore conțin de obicei 16 sau 24 biți pe un pixel. Pentru o imagine cu 8 biți, valoarea unui pixel pentru culoarea neagră este 0, în timp ce valoarea pentru culoarea albă este 255. În general la antrenare, imaginile sunt convertite în imagini grayscale pentru a reduce timpul de antrenare.

În funcție de modul în care sunt folosite imaginile, ele pot fi salvate în diferite formate. Cele mai populare formate sunt JPG, PNG, TIFF și GIF. Pentru imaginile din setul de antrenare se recomandă să se folosească formatul JPG, deoarece acest format comprimă imaginea și minimizează spațiul de pe disc. Chiar dacă imaginea își pierde câteva detalii în timpul compresiei, aspectul final al imaginii nu este cu mult diferit față de o imagine fără compresie.

2.3.Zgomotul în imagini

Zgomotul în imagini reprezintă diferite variații ale culorilor, luminozității sau mediului înconjurător. Deoarece zgomotul are un impact foarte puternic asupra procesării imaginilor, este esențial ca acesta să fie eliminat pentru a se putea crea un model final cât mai bun.

Zgomotul creat de mediul înconjurător se referă la animalul principal din imagine și fundalul imaginii care nu conține acel animal. Pentru setul de imagini pozitive este esențială eliminarea zgomotului creat de mediul înconjurător pentru a se putea extrage trăsăturile animalului. Eliminarea zgomotului se realizează prin decuparea animalului sau animalelor care se doresc a fi găsite din imagini. Dacă acest zgomot nu este eliminat, există posibilitatea ridicată ca să se găsească rezultate fals-pozitive (de exemplu fundalul unei imagini poate fi clasificat ca un animal).

Zgomotul creat de luminozitate sau variații de culoare se împarte în două categorii : zgomot gaussian și zgomot salt & pepper. Pentru obținerea unui model cât mai bine antrenat, se recomandă ca imaginile din setul de date de antrenament să nu conțină zgomot deoarece acesta poate scădea calitatea trăsăturilor extrase.

Capitolul 3

Clasificatori în cascadă

Un clasificator este un algoritm sau un set de funcții matematice care implementează un clasificator ceea ce are ca scop împărțirea instanțelor dintr-un set de date în mai multe categorii, în funcție de anumite instanțe ale căror categorii se cunosc.

Termenul „în cascadă” al unui clasificator reprezintă modalitatea prin care acesta este construit, prin ansamblarea clasficatorilor simpli pentru a se crea un clasificator final. Clasificatorul final este construit din mai mulți clasificatori simpli sau de bază (numiți stagii) care sunt aplicați pe anumite regiuni, în funcție de cat de multe trăsături se găsesc în acea regiune. Clasificatorii de bază sunt clasificatori bazați pe arbori de decizie ce conțin măcar doua nivele, inputul fiind trăsăturile extrase din setul de date de antrenare. Pentru ansamblarea clasificatorilor simpli se folosește algoritmul adaboost.

3.1. Algoritmul Viola & Jones

Apărută în 2001, lucrarea „Rapid Object Detection using a Boosted Cascade of Simple Features” publicată de Paul Viola și Michael Jones conține o abordare bazată pe învățare automată pentru problema detecției obiectelor în timp real cu o acuratețe mare și timp de procesare mic pentru imagini.

Algoritmul prezentat în lucrare are patru etape care se execută succesiv pentru extragerea trăsăturilor, antrenare și ansamblare. Chiar dacă algoritmul a fost creat inițial pentru detectarea fețelor umane, el funcționează pentru fețele animalelor cât și pentru obiecte simple (cu o acuratețe mai slabă, deoarece trăsăturile sunt extrase într-un mod specific fețelor). [2]

1. Selectarea și extragerea trăsăturilor

Deoarece trăsăturile sunt esențiale în clasificarea imaginilor, procesul de creare a unui clasificator va începe cu selectarea și extragerea trăsăturilor. Motivele pentru care se lucrează cu trăsături și nu direct cu pixeli sunt performanța și imposibilitatea de a se face antrenarea pe un set atât de variat, deoarece fiecare pixel prezintă informații despre culoare și intensitate. În continuare vom discuta despre cele mai folosite tipuri de trăsături, Haar și LPB.

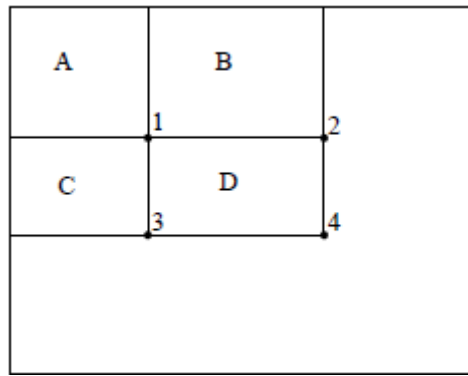
Trăsăturile de tip Haar sunt forme geometrice dreptunghiulare care sunt aplicate imaginilor din setul de antrenament și care selectează și extrag zonele cât mai bogate în trăsături din imagini. Câteva trăsături importante sunt : localizarea nasului și a gurii, tipul urechilor, forma nasului și a gurii. Alte trăsături mai pot fi culoarea regiunilor din împrejurul ochilor sau diferențele de culoare dintre diferite părți ale feței, dar aceste trăsături nu sunt neapărat valide la clasificarea animalelor deoarece culorile diferite ale pielii și părului pot afecta negativ clasificatorul.

Trăsăturile de tip LBP (local binary pattern) sunt grupări de pixeli clasificați prin verificarea vecinătăților pixelilor din imaginilor transformate în grayscale după niște praguri stabilite și scrierea valorilor de 0 pentru când valoarea vecinului este mai mică decât cea a pixelului după care se lucrează și 1 atunci când valoarea este mai mare. [4]

În aplicația prezentată se folosesc trăsături de tip Haar deoarece acestea o acuratețe mai ridicată față de trăsăturile LPB (15-20%). Chiar dacă antrenarea unui clasificator de tip Haar necesită o durată mai lungă, acuratețea clasificatorului este esențială pentru o cât mai bună funcționare iar antrenarea se realizează o singură dată, rezultatul fiind salvat într-un fișier pentru a putea fi folosit mai târziu la detecție.

2. Crearea unei imagini integrale

Intr-o zonă de rezoluție 24 x 24 dintr-o imagine, numărul trăsăturilor dreptunghiulare este peste 160.000 iar pentru a se putea calcula cât mai repede trăsăturile rectangulare se folosește o reprezentare temporară a imaginii numită imagine integrală. O imagine integrală la un anumit punct de coordonate x și y conține suma pixelilor de deasupra și din dreapta punctului respectiv.



Fie figura de mai sus, folosind o imagine integrală, valoarea imaginii integrale în punctul 1 este suma pixelilor din chenarul A. Pentru locația 2 valoarea este $A + B$, pentru locația 3 valoarea este $A + C$ iar pentru locația 4 valoarea este $A + B + C + D$. Suma pixelilor din chenarul D poate fi calculă ca fiind $4 + 1 - (2 + 3)$, unde fiecare locație a fost calculată precedent.[5]

3. Antrenarea folosind AdaBoost

Algoritmul Adaboost este folosit într-o variantă modificată pentru a selecta un set restrâns de trăsături și pentru a antrena clasificatorul. De exemplu pe o zonă de rezoluție 24×24 care conține peste 160.000 trăsături, numărul trăsăturilor pastrate este între câteva sute și câteva mii. Alegerea trăsăturilor se face în funcție de cât de bine acestea fac separarea între imaginile pozitive și imaginile negative.

Pentru trăsăturile alese în primele stagii ale antrenării, rata erorii este între 0.1 și 0.3 (deoarece valorile, dar cu cât se avansează în stagii, obținerea trăsăturilor unice devine tot mai complicată iar rata erorii poate crește până la 0.4. În funcție de modul în care au fost procesate imaginile (mai ales cele din setul de imagini pozitive), această rată poate crește și afecta negativ clasificatorul, prin eliminarea unor trăsături care puteau să ajute la detecție, dacă imaginea era procesată corect.

4. Crearea clasificatorului în cascadă

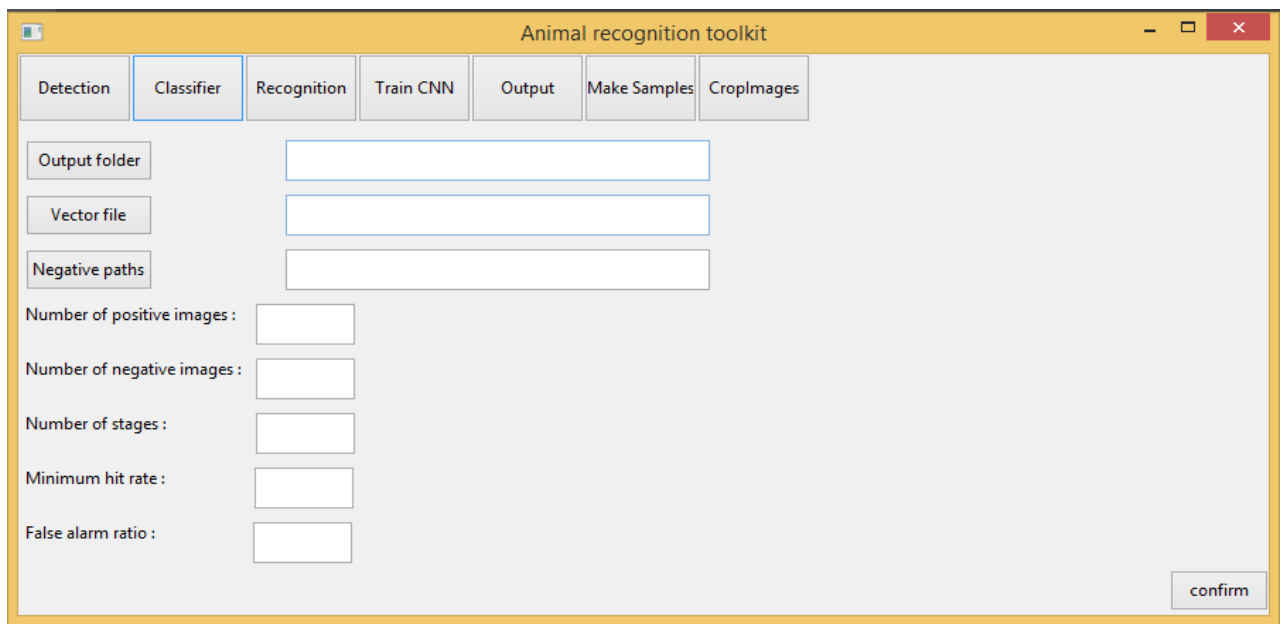
Deoarece fiecare stagiou este un clasificator puternic, toate trăsăturile sunt grupate în diferite stagii cu diferite trăsături. Fiecare stagiou are ca scop clasificarea dacă o anumită trăsătura reprezintă fața unui animal sau nu, iar trăsăturile ce nu îndeplinesc această condiție sunt imediat eliminate.

Antrenarea și crearea clasificatorului se încheie de obicei atunci când s-a ajuns la numărul de stagii precizate, dar și atunci când se ajunge la potențialul statistic al clasificatorului

(fie din cauză că sunt prea puține date sau parametrii antrenării au fost setați pentru a accepta o rată a clasificărilor false prea mare).

3.2. Antrenarea clasificatorului și detecția pe imagini

Pentru o mai ușoară folosire, aplicația prezintă o interfață care ajută utilizatorul să facă antrenarea și detecția animalelor folosind clasificatoare în cascadă deja antrenate sau antrenate chiar de către utilizator pe animalele dorite.



În imaginea de mai sus se găsește meniul principal, împreună cu panoul prin care se poate face antrenarea. Înainte de a putea începe, utilizatorul trebuie să precizeze parametrii necesari antrenării, împreună cu path-urile către diferite foldere și fișiere. Parametrii sunt :

Output : Conține path-ul către un folder unde va fi salvat outputul programului. Outputul constă în clasificatorul final, un fișier cu parametrii introduși la antrenare și câte un fișier ce conține progresul pentru fiecare stadiu în parte. Fișierele cu progresul pe stagii ajută atunci când vrem să continuăm antrenarea unui clasificator care a ajuns la un număr de stagii, sau când se dorește îmbunătățirea unui clasificator deja antrenat.

Vector : Conține path-ul către un fișier de tip vector (extensia .vec) ce conține imaginile pozitive folosite pentru antrenare. Acest fișier este creat folosind panoul „Create Samples” din

interfață, și necesită un fișier cu adnotările pentru imaginile pozitive. Fișierul cu adnotările este de fapt un fișier text ce conține pe fiecare linie path-ul către imagini și numărul de animale dintr-o imagine împreună cu locațiile unde se găsesc acestea.

Negative paths : Conține path-ul către un fișier ce este format din toate path-urile imaginilor negative, fiecare pe o linie separată. Crearea acestui fișier se poate face folosind panoul „Create Samples” din interfață, specificând fișierul cu imagini negative.

Number of positive images : Înainte de a începe antrenarea se va specifica numărul de imagini pozitive ce vor fi folosite pentru antrenare. Este bine de menționat faptul că acest număr nu trebuie să depășească numărul total de imagini din fișierul cu adnotări.

Nr. of negative images : Numărul de imagini negative este de asemenea specificat, iar acest număr nu trebuie să depășească numărul total de imagini pozitive din fișierul specificat la „Negative paths”.

Number of stages : Numărul de stagii de antrenare ale clasificatorului reprezintă cât de complex va fi clasificatorul final. Un număr cât mai mare de stagii la antrenare înseamnă ca rezultatul final va fi foarte bine antrenat, dar trebuie ținut cont și de faptul că suprapotrivirea (overfitting) poate să apară ca urmare a numărului tot mai mare de trăsături alese de algoritm. De asemenea, trebuie ținut cont și de timpul necesar la antrenare, deoarece acesta devine tot mai mare cu cât numărul de stagii crește.

Minimum hit rate : Acest parametru ne asigură că datele noastre de antrenare pozitive produc un rezultat cât mai bun. De exemplu o valoare de 0.9 ne spune că există posibilitatea ca 10% din datele noastre pozitive să fie etichetate greșit, ceea ce ar scădea acuratețea clasificatorului. O valoare uzuală este de 0.99 sau 0.995 ce indică o rată de până la 1%.

False alarm ratio : Acest parametru reprezintă probabilitatea ca o imagine să fie etichetată fals ca fiind negativă (să nu conțină animalul căutat, chiar dacă acest lucru nu este adevărat) sau ca imaginea să fie corect etichetată negativ. Cu cât această valoare este mai mică, cu atât vom avea clasificatoare mici mai slabe care vor conține multe trăsături ce trebuie verificate pentru a fi eliminate sau nu.

După ce s-au introdus toți parametrii, se poate apăsa butonul „confirm” pentru a începe

antrenarea. Este de menționat faptul că antrenarea nu poate începe dacă nu s-au introdus toți parametrii sau nu au fost introduși corect. Utilizatorul va primi diferite mesaje de eroare, în funcție de greșelile făcute, așa cum se poate observa în imaginea de pe următoarea pagină.

Antrenarea clasficatorului poate dura de la câteva minute la câteva zile sau chiar săptămâni, în funcție de cât de bine au fost procesate imaginile și în funcție de numărul de stagii. Rezultatul final este un clasificador ce poate fi folosit mai târziu pentru detecție.

Outputul realizat în procesul de antrenare poate fi vizualizat la orice moment folosind panoul „Output”. Pe lângă unele date statistice, se mai pot vedea și timpul trecut la antrenarea fiecărui stadiu cât și timpul estimat pentru antrenarea stagiului actual.

Capitolul 4

Rețele neuronale

Rețelele neuronale sunt seturi de algoritmi modelați după creierul uman, care au ca scop identificarea modelelor și încercarea de a simula interacțiunea cu mediul înconjurător. O rețea neuronală este formată din noduri, care se mai numesc și neuroni artificiali, poziționați pe mai multe straturi care se ocupă cu preluarea inputului, verificarea ponderilor datelor de intrare și producerea outputului.

Neuronii sunt operații matematice care primesc datele de input cu o anumite ponderi, înmulțesc fiecare input cu ponderea specifică, sumează toate valorile obținute și mai apoi trimit valoarea finală către alți neuroni, printr-o funcție de activare. [6]

Stratul conectat cu zona modelului de inputul și zona de output a modelului se mai numește și „Hidden layer” sau „Strat Ascuns”. Nodurile din stratul ascuns pot primi input doar de la alte noduri (noduri de input sau alte noduri din stratul ascuns), iar outputul acestora este trimis doar la noduri de output sau alte noduri din stratul ascuns. [7]

4.1 Rețele neuronale convoluționale

O rețea neuronală convoluțională este un tip de rețea neuronală care se folosește în problemele legate de analiza imaginilor deoarece este specializată în detecția tiparelor sau modelelor. Esențială la rețelele neuronale convoluționale este existența straturilor convoluționale, în stratul ascuns al rețelei, care au anumite filtre care se ocupă cu detecția colțurilor, formelor și obiectelor specifice cum ar fi animalele.

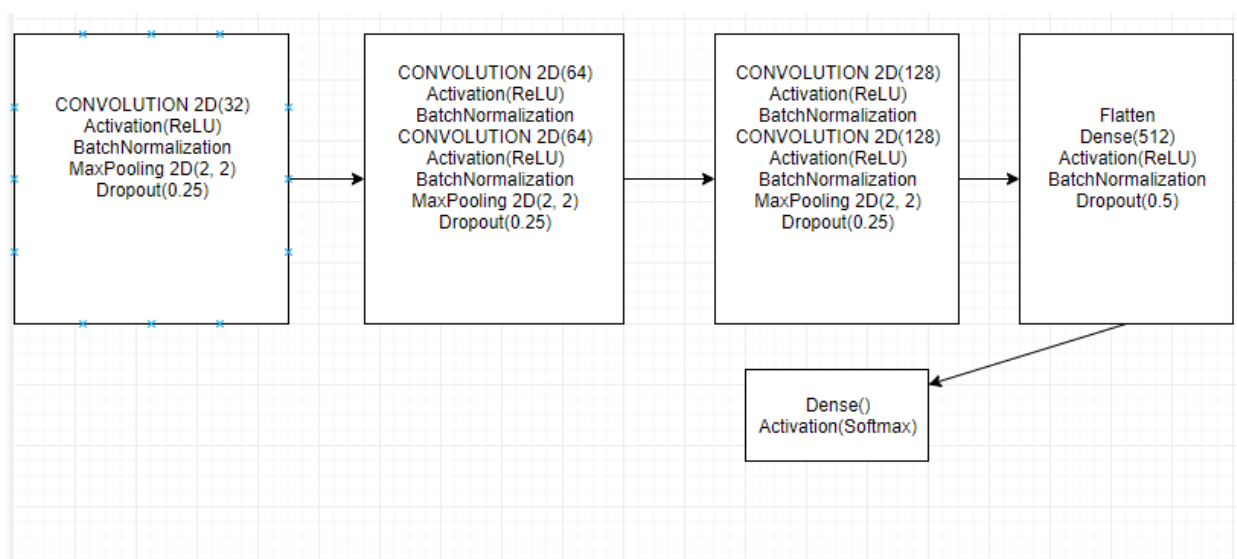
Prima operație a stratului convoluțional se numește nucleu. Deoarece imaginile sunt reprezentate ca o matrice de lungime x adâncime x număr de frecvențe ale culorilor (de exemplu 3 pentru frecvența RGB), nucleul declarat la crearea stratului (de exemplu o matrice de 3 x 3 x 1) se va mișca peste toată matricea imaginii pentru a extrage trăsături low level cum ar fi culori sau schimbările bruște ale intensității culorilor. Prin adăugarea de mai multe straturi, arhitectura rețelei se adaptează și la detecția trăsăturilor high level cum ar fi diferite obiecte.

4.2 Arhitectura modelului pentru recunoașterea animalelor

Înainte de a începe antrenarea trebuie specificată arhitectura rețelei neuronale. Ca schemă de plecare s-a folosit modelul „VGGNet”, câștigător al ILSVRC 2014 [8]. Cele mai importante trăsături ale acestui model sunt :

- Folosirea doar a filtrelor de dimensiuni 3x3.
- Straturile convoluționale sunt puse unul peste altul în zonele mai adânci ale rețelei neuronale, înainte să fie aplicată operația de pooling (punere în comun a trăsăturilor).

Arhitectura rețelei constă într-un strat de input, un strat de output și două straturi convoluționale, la care se aplică în final un strat dens.



Stratul de input prezintă 32 filtre de dimensiune 3x3, iar cele două straturi convoluționale prezintă 64 și 128 filtre. Funcția de activare folosită în straturile convoluționale este ReLU (Rectifier Linear Unit) deoarece este de până la 6 ori mai eficientă decât funcția de convergență Tanh. Această funcție de activare are și o limitare, ea putând fi folosită doar în straturile ascunse ale rețelei. [9]

În capitolul următor se va prezenta aplicație completă, împreună cu restul operațiilor asupra straturilor.

Capitolul 5

Descrierea soluției

Scopul aplicației este antrenarea și folosirea clasificatoarelor pentru detecția animalelor, cât și antrenarea modelelor și recunoașterea animalelor cu ajutorul lor.

Pentru clasificatoarele în cascadă sunt necesare anumite operații asupra setului de date pentru a se putea începe antrenarea. Imaginile pozitive trebuie decupate și decupate, apoi trebuie structurate într-un fișier de tip vector pentru a putea fi comparate cu imaginile negative și pentru a se putea extrage trăsăturile.

Pentru modelele create folosind rețele neuronale, imaginile nu sunt împărțite în imagini pozitive și imagini negative. Setul de date este împărțit în imagini de antrenare și imagini de testare (cu o rată furnizată de către utilizator), iar mai apoi fiecare imagine este aplicată secvențial straturilor rețelei, în final rezultând trăsăturile cele mai proeminente care sunt aplicate modelului și mai apoi salvate.

5.1 Arhitectura proiectului

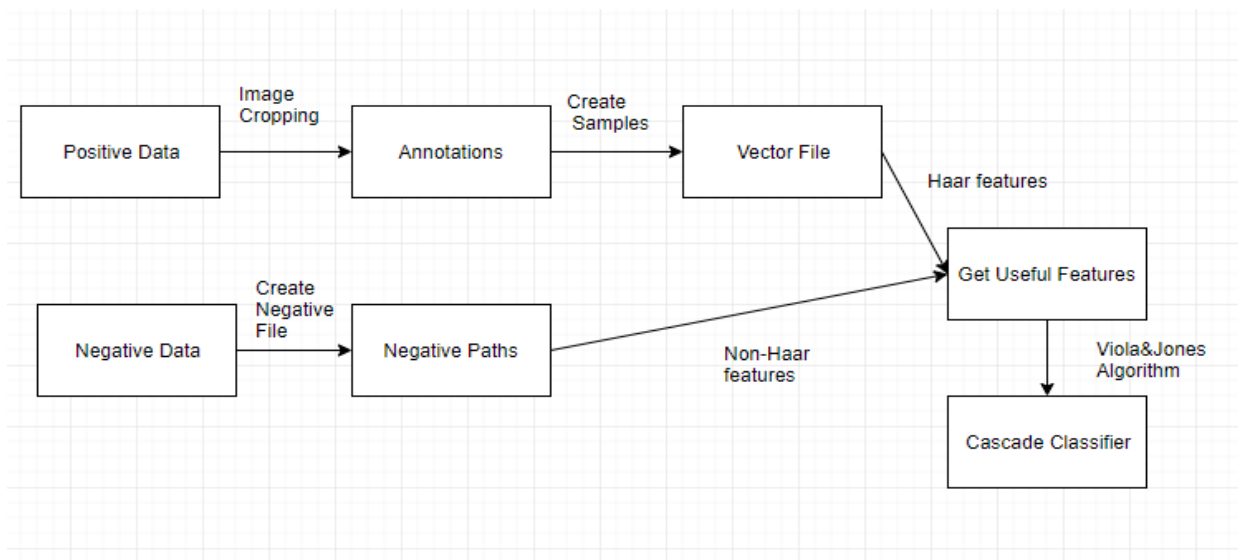
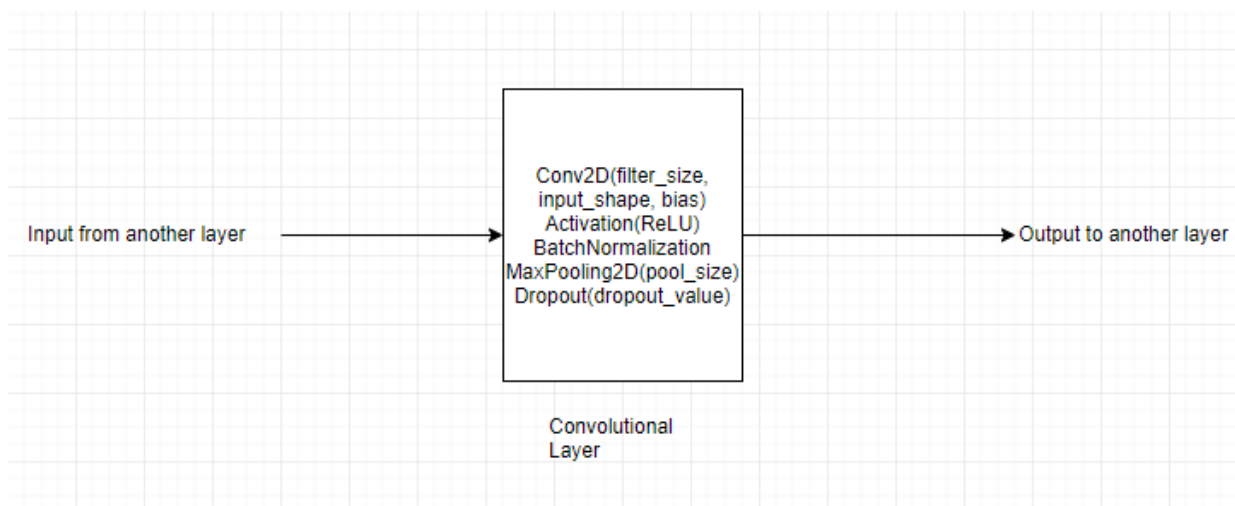


Diagrama de mai sus prezintă în mare operațiile făcute pe setul de date de antrenament pentru a se crea un clasificator în cascadă.

Metoda ce folosește rețele neuronale este împărțită în două module, unul high level care folosește Keras pentru a crea straturile rețelei convoluționale și alt modul low level care folosește Tensorflow pe partea de backend pentru a realiza operațiile concrete de procesare și extragere a trăsăturilor.



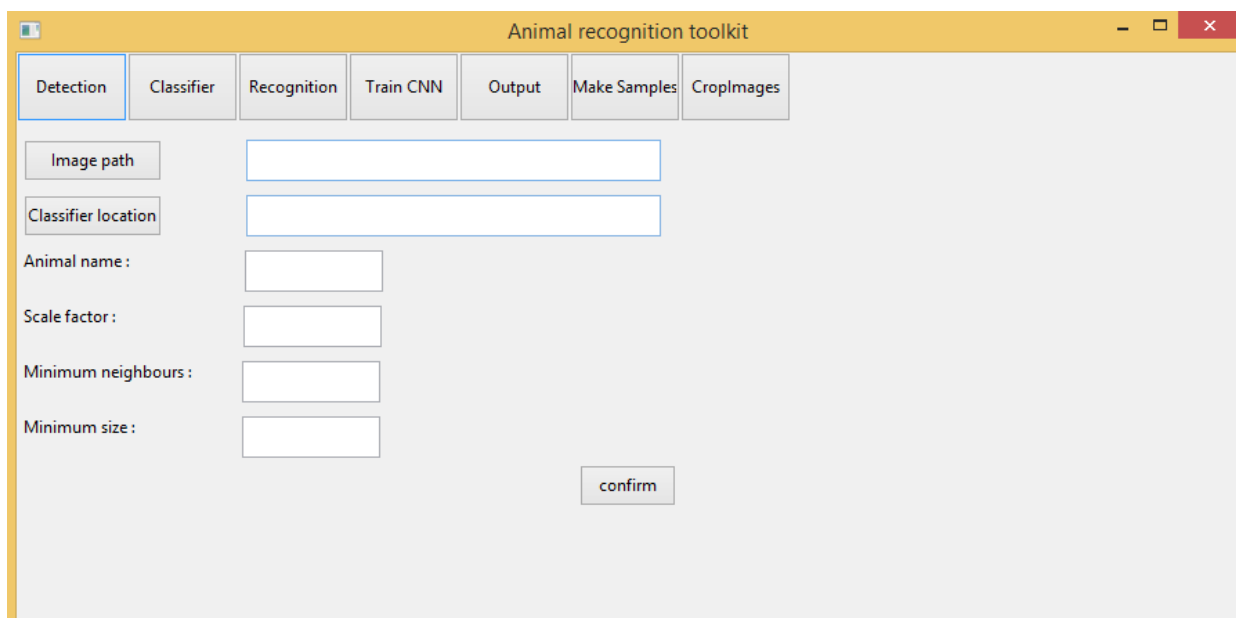
În diagrama de mai sus este prezentată structura unui strat convoluțional. Normalizarea se aplică după funcția de activare pentru a standardiza valorile trăsăturilor pentru a accelera procesul de învățare. De exemplu, avem un model antrenat pentru recunoașterea pisicilor dar folosim doar pisici negre pentru datele noastre de antrenament. Dacă încercăm să detectăm pisici cu diferite culori folosit modelul antrenat în exemplul precedent, rata de găsim a animalului scade (în unele cazuri nu găsește deloc pisica), chiar dacă antrenarea a fost făcută doar pe imagini cu pisici. Normalizarea poate reduce și overfitting-ul modelului, deoarece procesul adaugă puțin zgomot la activarea fiecărui strat. [10].

Straturile de punere în comun (Pooling Layers) sunt folosite pentru a reduce spațiul și dimensiunea reprezentărilor, astfel reducând numărul de parametri nu numărul de calcule ale rețelei. Cel mai folosit tip de pooling este MaxPooling, unde pentru fiecare input (reprezentat ca o matrice patratică de o anumită lățime și înălțime) se aleg regiunile specificate (de obicei o zonă de 2x2 sau 4x4), se ia valoarea maximă din acea regiune și se crează o nouă reprezentare a input-ului cu mai puține elemente. Straturile de punere în comun se plasează de obicei între straturile convoluționale consecutive. [11]

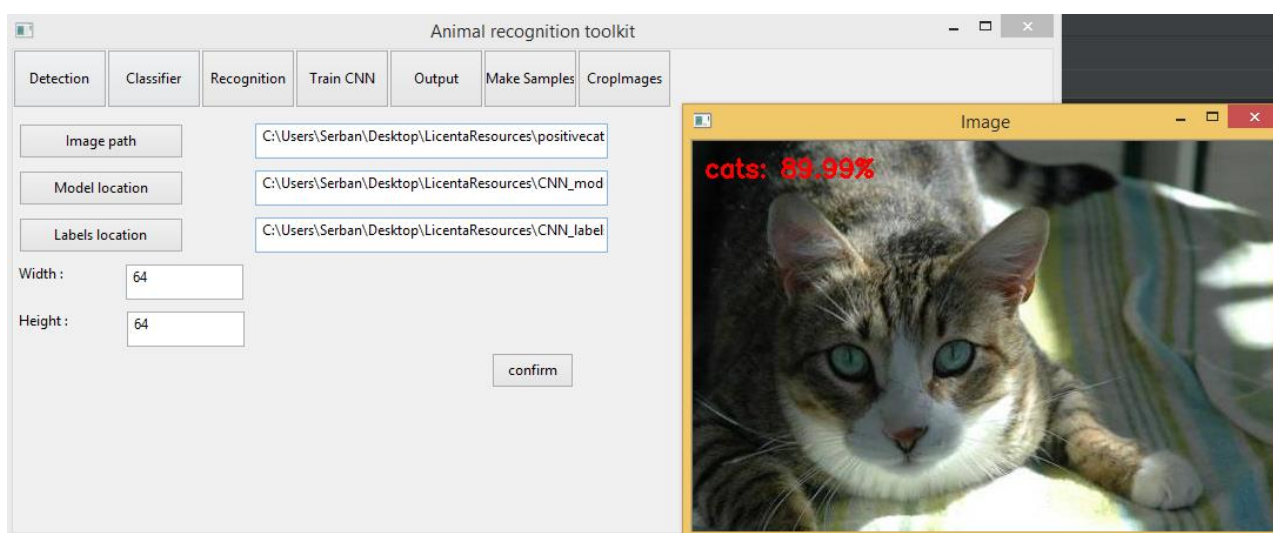
Stratul de dropout constă în deconectarea anumitor neuroni aleatori dintre straturi pentru a preveni overfitting-ul, crește acuratețea și ajută modelul să devină mai robust (să poată detecta mai bine imaginile mai nefamiliare, diferite de datele folosite la antrenament). [12]

5.2 Prezentare interfață, antrenare, statistici

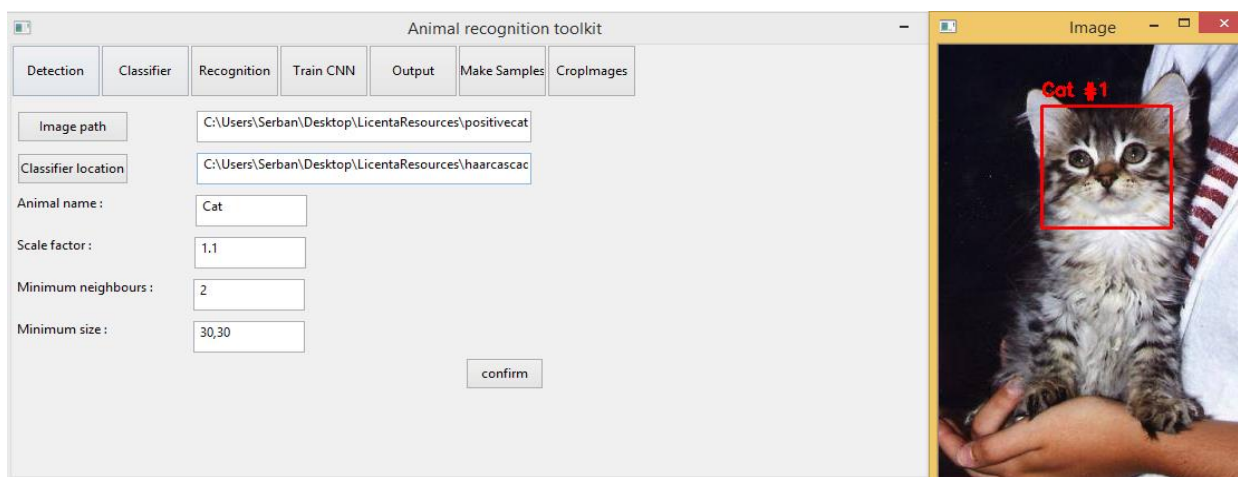
Interfața aplicației pune accent pe ușurința folosirii de către utilizator, accentul punându-se pe modul în care sunt create clasificatoarele și modelele, prin folosirea diferiților parametri sau prin alterarea setului de date de antrenament.



Panoul default al aplicației este panoul de detecție a unui animal dintr-o imagine, folosind clasificatori în cascadă deja antrenați. Celelalte panouri, în ordinea aranjării lor în meniu, sunt folosite pentru a antrena un clasificator, a recunoaște un animal folosit un model deja antrenat, antrenarea unui model folosind rețele neuronale convoluționale, prezentarea outputului aplicației, procesarea imaginilor pozitive și negative pentru clasificatoare și în final decuparea imaginilor, pentru o antrenare mai rapidă.



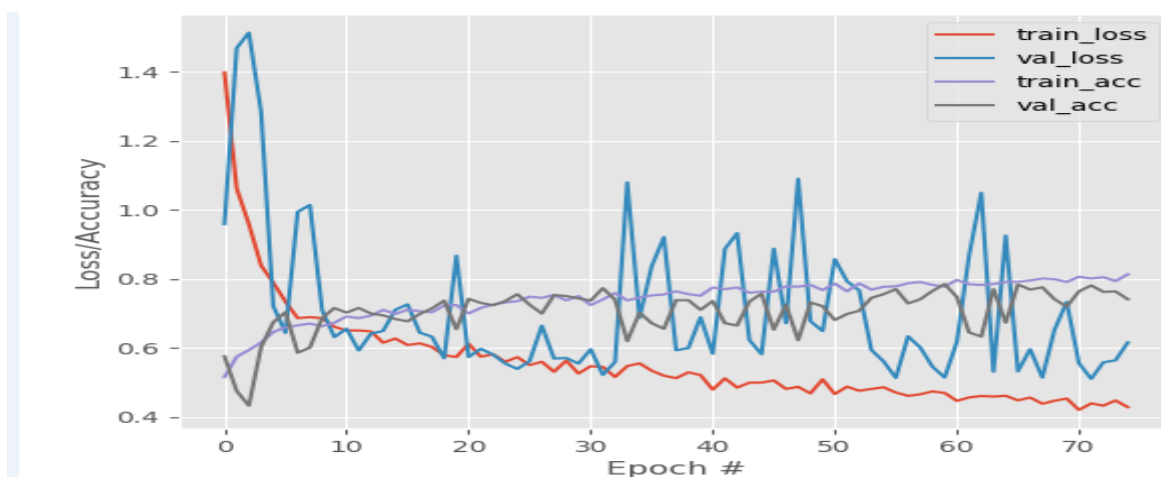
În imaginea de mai sus se poate vedea folosirea unui model antrenat pentru recunoașterea unei pisici. Parametrii width și height sunt dimensiunile la care e scalată imaginea înainte de aplicarea modelului pentru recunoaștere.



Pentru detecția folosind un clasificator în cascadă, parametrii specificați pot face diferența dintre detectarea unui animal în imagine și detectarea greșită a mai multor animale sau chiar eșuarea de a se detecta un animal.

Factorul de scalarizare este folosit pentru a se crea o piramidă a imaginii, astfel încât să se poată detecta fețe mari sau mici folosind aceeași fereastră de detecție. Folosind o valoare de 1.05, dimensiunea imaginii este redusă cu 5% de fiecare dată până se găsește o potrivire între imagine și model. [13]

Parametrul minimum neighbours reprezintă cât de apropiate trebuie să fie trăsăturile găsite de clasificator. O valoare mică (0-2) poate produce multe rezultate fals pozitive, detectând fețe acolo unde nu trebuie. Parametrul minimum size indică dimensiunea minimă a obiectelor ce pot fi detectate, obiectele mai mici fiind ignorate. Pentru detectarea fețelor, [30, 30] este o valoare acceptabilă de pornire.



Mai sus se poate găsi rezultatul statistic al antrenării unui model pe câini, pisici și panda. Training loss reprezintă media dintre toate funcțiile de cost la antrenare pe fiecare lot (batch), iar testing loss este calculat pentru fiecare epocă (epoch) folosind modelul în varianta de la finalul epocii. Acuratețea reprezintă cât de corect este modelul nostru la prezicerea animalelor, iar funcția de cost (val_loss) reprezintă probabilitatea cu care modelul a clasificat greșit datele.

Concluzi

Această lucrare a avut ca scop introducerea cititorului în domeniul procesării imaginilor și recunoașterii animalelor din respectivele imagini. Interfața prietenoasă a aplicației combinată cu modul în care a fost structurată lucrarea este perfectă pentru utilizatorii care nu au foarte multe cunoștințe legate de procesarea imaginilor dar și pentru utilizatorii avansați ce doresc să folosească o aplicație care le permite crearea unor modele foarte ușor, ei punând accentul pe antrenarea și perfecționarea acestor modele folosind diferiți parametri prezentați în aplicație și lucrare.

Modelele create au o acuratețe decentă, ele fiind capabile să găsească animalele precizate în diferite imagini, atât timp cât au fost folosite cât mai multe imagini pentru antrenare și acestea au fost procesate corect, în funcție de necesitățile modelului.

O direcție de viitor ar putea fi extinderea aplicației pe web sau pe mobile, pentru a acoperi o gamă cât mai largă de utilizatori. Codul a fost organizat pe clase pe partea de front-end și pe funcții pe partea de back-end, astfel adăugarea de noi funcționalități se poate face foarte ușor. Deoarece rețelele neuronale devin cât mai populare în domeniul recunoașterii obiectelor din imagini, noi modele și abordări pot ieși la suprafață și pot fi introduse în aplicație.

Bibliografie

- <http://aishack.in/tutorials/sift-scale-invariant-feature-transform-introduction/> [1]
- <http://aishack.in/tutorials/harris-corner-detector/> [2]
- Paul Viola & Michael Jones, **Rapid object detection using a Boosted Cascade of Simple Features**, 2001 [3]
- <https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b> [4]
- Paul Viola & Michael Jones, **Robust Real-Time Face Detection**, 2003 [5]
- <https://skymind.ai/wiki/neural-network> [6]
- <http://www.uta.fi/sis/tie/neuro/index/Neurocomputing2.pdf> [7]
- <https://medium.com/coinmonks/paper-review-of-vggnet-1st-runner-up-of-ilsvlc-2014-image-classification-d02355543a11> [8]
- <https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f> [9]
- <https://towardsdatascience.com/batch-normalization-in-neural-networks-1ac91516821c> [10]
- <https://medium.com/technologymadeeasy/the-best-explanation-of-convolutional-neural-networks-on-the-internet-fbb8b1ad5df8> [11]
- <https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5> [12]
- <https://answers.opencv.org/question/10654/how-does-the-parameter-scalefactor-in-detectmultiscale-affect-face-detection/> [13]
- Keras Documentation : <https://keras.io/>
- Tensorflow Documentation : https://www.tensorflow.org/api_docs/python
- Python Documentation : <https://www.python.org/doc/>
- OpenCV Documentation : https://docs.opencv.org/master/d9/df8/tutorial_root.html
- Wx Python Documentation : <https://wiki.wxpython.org/>

- Scikit Documentation : <https://scikit-learn.org/stable/index.html>
- Matplotlib Documentation : <https://matplotlib.org/tutorials/index.html>
- David Kriesel, **A brief introduction to neural networks**, 2005
- Michael Nielsen, **Neural Networks and Deep Learning**