**UNIVERSITATEA DIN BUCUREȘTI**



**FACULTATEA**

**DE**

**MATEMATICĂ ȘI INFORMATICĂ**

**SPECIALIZAREA MATEMATICĂ-INFORMATICĂ**

**Lucrare de licență**

**RECUNOAȘTEREA AUTOMATĂ A PLANTELOR DIN IMAGINI**

**Absolventă**

**Buican Laura Andreea**

**Coordonator științific**

**Conf. Dr. Alexe Bogdan**

**București, septembrie 2021**

**Rezumat**

Proiectul cuprinde, pe de o parte, implementarea și folosirea unor rețele specializate pe clasificarea imaginilor precum ResNet sau Inception, studiindu-se rezultatele și performanțele pe care le obținem în diferite situații precum folosirea sau evitarea augmentării, Batch normalization sau comparând rezultatele rețelelor nepreantrenate cu cele preantrenate. Pe de altă parte, este creată o aplicație de Android de recunoaștere a florei. Este o platformă educativă, putând fi folosită pentru a cataloga diferite specii de plante. Conține un meniu principal, o pagină de realizare a imaginilor și de clasificare a lor, o fereastră cu o galerie în care sunt organizate pe specii fotografiile cu plante recunoscute. În cazul în care nu este fotografiată o plantă sau imaginea conține o plantă necunoscută, aceasta va fi repartizată la categoria “Necunoscut”.

**Summary**

This work presents, in the first part, the implementation and use of neural networks for image classification such as ResNet or Inception and studies of the results and performance we obtain in different situations such as using or avoiding augmentation, Batch normalization or comparing the results of untrained networks with those pre-trained. In the second part, an Android flora recognition application is created. It is an educational platform that can be used to catalog different plant species. The application contains a main menu, a page for choosing images and classifying them, a window with a gallery in which photos of recognized plants are organized by species. If a photograph does not contain a plant or the image contains an unknown plant, it will be assigned to the "Unknown" category.

# Cuprins

Cuprins 4

Introducere 6

1. Structura lucrării de licență 7

2. Tehnologii utilizate 8

Capitolul I. Concepte teoretice 12

Capitolul II. Dezvoltarea rețelelor pe desktop 15

1. Prelucrarea datelor 16

2. SVM (Support Vector Machine) 17

3. ResNet 18

4. InceptionV2 22

5. Evaluarea 24

6. Exportul modelului și pregătirea pentru aplicația de Android 25

Capitolul III. Aplicația de Android 27

1. Dezvoltarea programului 27

2. Interacțiunea cu aplicația 28

Capitolul IV. Rezultate și comparații 34

1. Efectele augmentării 35

2. Efectele Batch normalization 36

3. Rețele preantrenate vs. nepreantrenate 37

4. Rețele alese pentru telefon 38

5. Comparații finale 39

Capitolul V. Concluzii 42

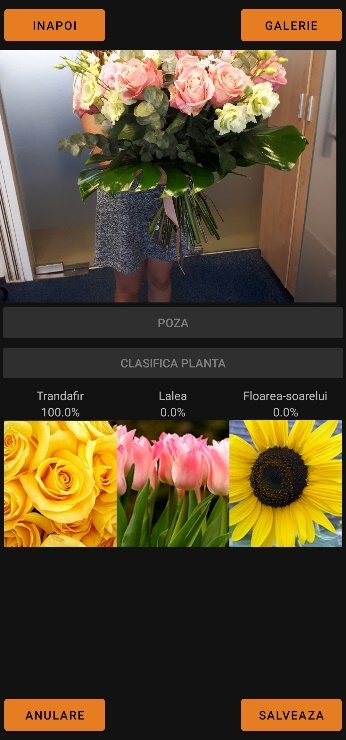
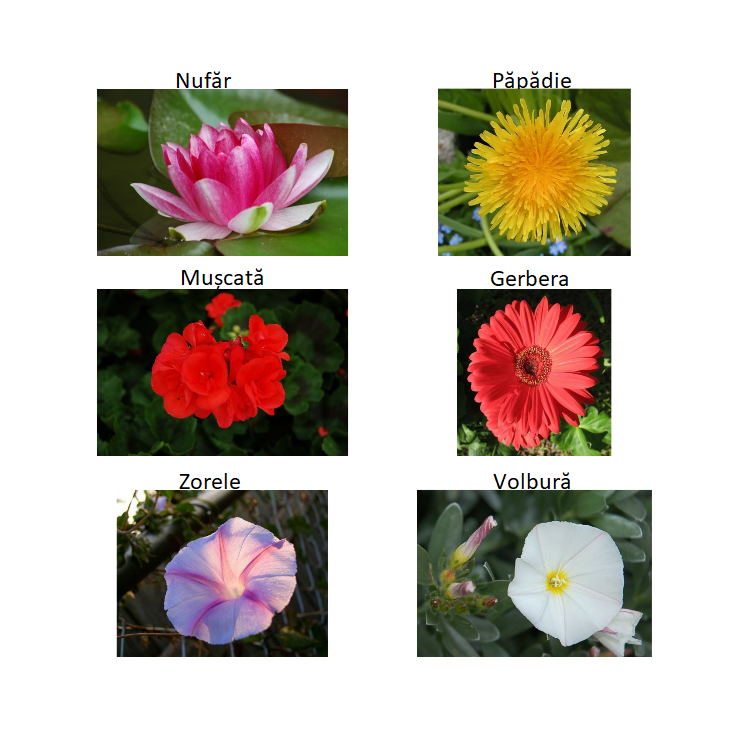
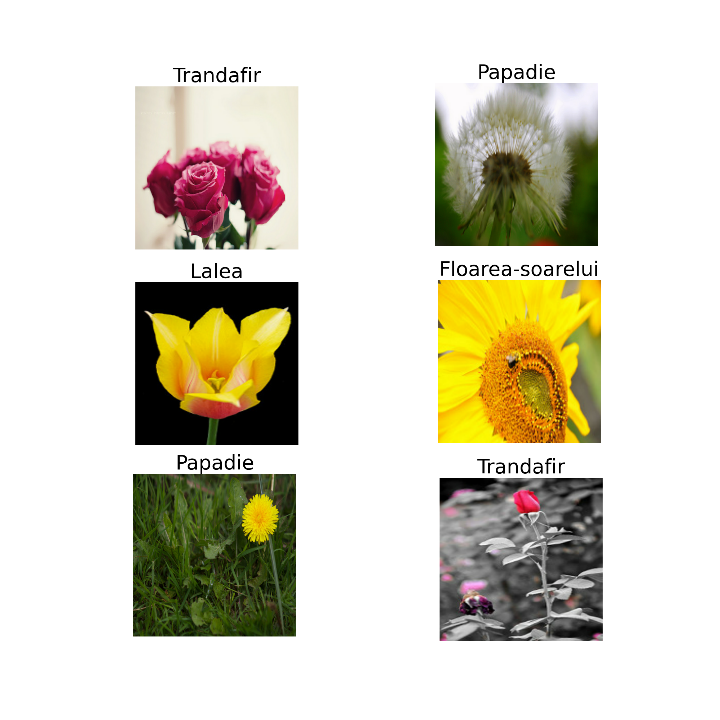
Bibliografie 43

Anexe 45

# Introducere

Lucrarea de față este încadrată în domeniul inteligenței artificiale, folosindu-se rețele neuronale convoluționale pentru învățarea și recunoașterea plantelor. S-au utilizat arhitecturi precum ResNet (Kaiming He, 2015) sau InceptionV2 (Christian Szegedy, 2015). Modelul învățat este apoi importat în aplicația de Android pentru a putea fi clasificate plante din imaginile oferite de utilizatori.

Am ales această temă deoarece domeniul recunoașterii plantelor este de viitor, se dezvoltă cu repeziciune și poate fi aplicat și utilizat în viața de zi cu zi. Am dorit să studiem felul în care reacționează inteligența artificială pe o problemă destul de simplă și să o încorporăm într-o aplicație educativă, ajutând curioșii care vor să afle mai multe, să învețe plante într-un mod atractiv și interactiv. Astfel, prin aplicația creată pentru telefon special pentru portabilitate, printr-un simplu click de clasificare, o rețea antrenată cu multe date ne poate scuti de minute bune de căutare, oferindu-ne un răspuns cât se poate de aproape de adevăr. În Figura 1 sunt prezentate mostre din aplicația de Android și din bazele de date utilizate.

Figură 1 Prezentare de început aplicație + baze de date

Contribuția proprie este studiul rețelelor pe două baze de date de plante și compararea rezultatelor în funcție de diferiți factori precum prezența sau lipsa augmentării sau a Batch normalization. De asemenea, realizarea unei aplicații de Android de recunoaștere a plantelor și organizare a datelor este cea de-a doua contribuție din acest proiect.

Primul set de date cu imagini de plante făcute cu telefonul este alcătuit din 10000 de imagini cu specii de plante din campusul Universității Forestiere din Beijing, denumită și BJFU100 și folosind ResNet26, s-a atins o rată de recunoaștere a plantelor de 91.78%. Baza de date este alcătuită din 100 de clase de specii de plante, fiecare specie având în jur de 100 de poze. Cercetătorii folosesc pentru recunoașterea plantelor caracteristici ale frunzelor, culori și forme ale florilor, iar mai rar, se folosesc și de fructe.

Sunt cunoscute deja aplicații de recunoaștere a plantelor precum LeafSnap (Appixi, 2021) sau PlantNet (PlantNet, 2021). Pentru a se rezolva problema dispariției sau explodării gradientului, s-a folosit ResNet care folosește skip-uri, făcând salturi printre straturi, fără a fi nevoit să parcurgă fiecare strat. În ultima perioadă, s-au alcătuit baze de date precum ImageNet (Stanford Vision Lab, 2021) sau MS COCO (COCO Consortium, 2015) ce conțin milioane de imagini aparținând multor clase. Cu ajutorul acestora, s-au antrenat rețele care au o acuratețe foarte mare în recunoașterea imaginilor.

## Structura lucrării de licență

Pentru a înțelege pe deplin conceptele teoretice, primul capitol va descrie detaliile teoretice ale proiectului. Se va notifica ce sunt rețelele neuronale convoluționale, precum și diferite alte noțiuni necesare creării programului de recunoaștere a plantelor.

În cel de-al doilea capitol va fi prezentat felul în care a fost gândit și construit programul de desktop, explicând despre SVM, ResNet, InceptionV2, precum și despre funcția de evaluare și pregătirea modelului astfel încât să fie compatibil cu aplicația de Android.

În al treilea capitol va fi descris în detaliu modul în care a fost dezvoltată aplicația de Android pentru recunoașterea plantelor și o prezentare a sa în interacțiunea cu utilizatorul.

Capitolul patru va cuprinde, pe de o parte, o comparație a tipurilor de rețele neuronale folosite, observând comportamentul lor în funcție de diverși parametrii, iar pe de altă parte, diferite rezultate și remarci ce trebuie menționate.

În ultimul capitol va fi regăsită concluzia referitoare la modul în care a fost realizat proiectul, câteva idei pentru cazul unor dezvoltări ulterioare și o opinie personală legată de rezultatele obținute pe parcursul cercetării și dezvoltării lucrării de licență.

## Tehnologii utilizate

**TensorFlow:**

S-a ales construirea aplicației în TensorFlow, deoarece este o bibliotecă gratuită creată de Google special pentru machine-learning, având atât o multitudine de instrumente și funcții deja implementate (precum baze de date, funcții de activare, funcții de pooling sau de afișare), cât și o sintaxă intuitivă și ușor de înțeles (TensorFlow, 2021). Am ales această bibliotecă deoarece, este un modul foarte cunoscut, astfel că s-au făcut deja destule proiecte și programe cu ajutorul lui, iar internetul conține multe exemple și tutoriale cu și despre această librărie. Poate fi rulat atât pe GPU, cât și pe CPU și există versiuni pentru multe sisteme de operare precum Linux, Mac, Windows sau Android.

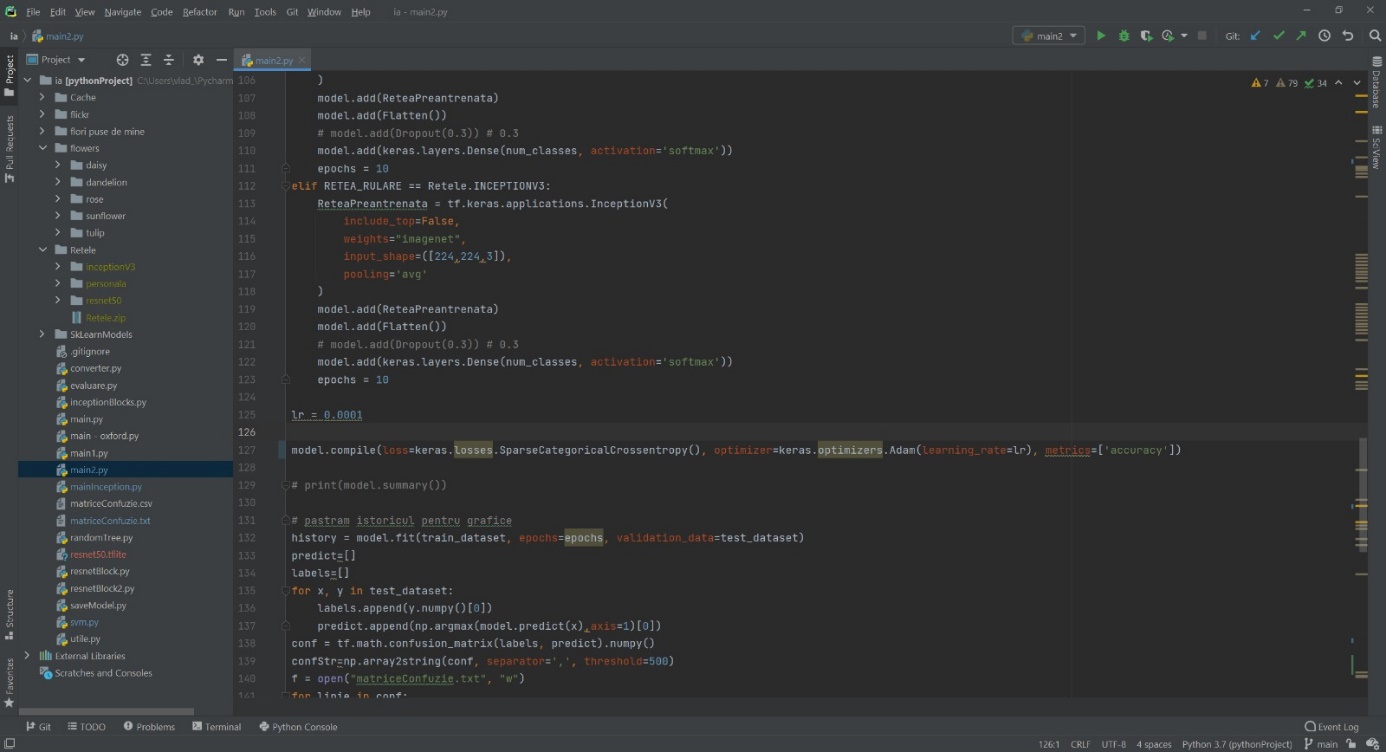
Totuși, principalul motiv pentru care am folosit TensorFlow este că datorită Framework-ului TensorFlow Lite, se poate exporta cu ușurință modelul de rețea neuronală creată pe desktop și importată în aplicația de Android.

**PyCharm:**

PyCharm este o platformă folosită de programatori pentru a construi cod în Python. Conține un portofoliu bogat în instrumente utile pentru dezvoltarea web și data science.

Am ales utilizarea acestei platforme deoarece, așa cum este specificat și în (JetBrains s.r.o., 2021), este gratuită pentru scopuri educaționale. Suportă versiuni ale limbajului Python precum: Python 2.7 sau de la 3.6 la 3.10. Este suportat de sisteme de operare precum Windows, Linux sau macOS. În varianta profesională se pot dezvolta aplicații Django, Pyramid sau Flask, precum și HTML.

Un motiv în plus pentru care am decis să dezvolt programul în PyCharm este acela că face dezvoltarea aplicațiilor cât mai ușoară, putând realiza modificări automate precum inițializarea câmpurilor sau adăugarea diverșilor parametrii.



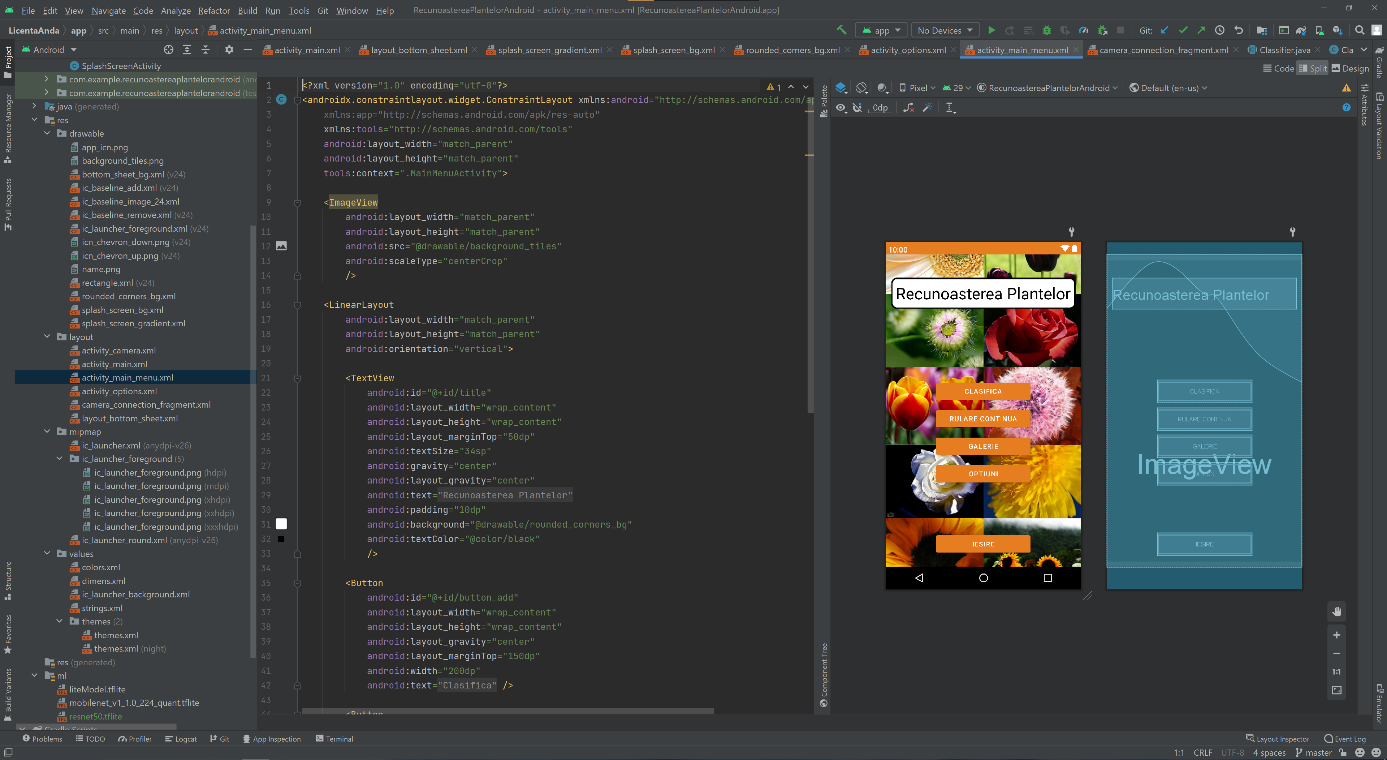
Figură 2 Interfață PyCharm

**Android Studio:**

Android Studio (Google, 2021) este o platformă de dezvoltare a aplicațiilor de Android ce suportă limbaje de programare precum Java sau Kotlin. Oferă o variație mare de beneficii precum: un emulator rapid și cu multe opțiuni, un mediu de dezvoltare a codului aplicațiilor, șabloane de cod ce ajută la facilitarea procesului de creare a programului. Este adaptabil la diferite sisteme de operare precum Windows, Linux sau MAC.

Am ales această platformă deoarece suportă sistemul de control al versiunilor GitHub, prin intermediul căruia am gestionat modificările și versiunile aplicației de-a lungul dezvoltării. Se pot instala mai multe versiuni de Android Studio pentru situațiile în care există erori pe o anumită versiune, astfel că se poate continua dezvoltarea programului într-o versiune stabilă.

De asemenea, un argument în plus pentru care am dorit să utilizăm Android Studio este faptul că există opțiunea de a dezvolta vizual, fără emulator, designul aplicației. Se pot face inclusiv modificări de front-end folosind doar fereastra care prezintă partea vizuală a aplicației, codul generându-se automat pe baza modificărilor aduse cu ajutorul acestei scurtături.



Figură 3 Interfață Android Studio

# Capitolul I. Concepte teoretice

Așa cum se specifică și în (Geron, 2019), rețelele neuronale au apărut în anii 1980 cu scopul recunoașterii imaginilor, în ultima perioadă evoluând și obținând performanțe uluitoare. Pe lângă obiectivul inițial, ajută și la activități precum recunoașterea vocii și procesarea limbajului natural sau mașini cu conducere automată. Varianta inițială a fost neocognitronul, inspirat din cortexul vizual, urmând ca mai apoi să evolueze în rețele neuronale convoluționale. În 1998 a fost introdusă arhitectura LeNet-5, folosită la scară largă de bănci pentru a putea recunoaște numerele de cecuri scrise de mână. Pe lângă straturile fully connected și funcțiile de activare (despre care vom discuta mai jos), sunt introduse două noi tipuri de straturi: straturile convoluționale și straturile de pooling. În rețelele convoluționale, neuronii din primul strat nu sunt conectați cu toți pixelii imaginii primite ca dată de intrare, ci doar cu cei din câmpul receptiv. Ierarhic apoi, fiecare neuron de pe un strat este conectat la un mic dreptunghi de pe stratul anterior. O astfel de structură ierarhică este comună și în imaginile din lumea reală, acesta fiind unul dintre motivele pentru care rețelele neuronale convoluționale funcționează așa de bine în recunoașterea imaginilor.

După cum este menționat și în (Geron, 2019), straturile de pooling au ca scop minimizarea masei de date care umple memoria, reducând sarcina de lucru prin micșorarea imaginilor și a numărului de parametri. Ca în cazul straturilor convoluționale, fiecare neuron este conectat la un câmp receptiv dreptunghic dintr-un strat anterior. Totuși, diferența este făcută de faptul că un neuron comun unește intrările printr-o funcție de conexiune.

Din articolul (Siddharth , et al., 2020), deducem că funcția de activare este des utilizată în rețelele neuronale, ajutând la transformarea datelor de intrare în date de ieșire. Comportându-se ca o stivă, datele de ieșire vor fi date de intrare pentru următorul strat. Într-o rețea neuronală, calculăm suma intrărilor înmulțite cu ponderile respective. Asupra rezultatului final vom aplica funcția de activare și îl vom folosi ca input pentru următorul strat.

Conform (Geron, 2019), Batch normalization are ca scop reducerea riscului dispariției sau explodării gradientului. Acesta adaugă înainte sau după fiecare funcție de activare o operație care centrează în zero și normalizează, apoi redimensionează și deplasează rezultatul folosind doi vectori de parametri noi pe strat: unul pentru scalare, celălalt pentru deplasare. Pentru aceasta, trebuie să estimeze media și deviația standard a fiecărei intrări. Astfel, el calculează media și deviația standard pentru datele curente.

Optimizatorii sunt algoritmi utilizați în rețelele neuronale care au ca scop micșorarea funcției de loss, folosindu-se de ponderi și de rata de învățare (Doshi, 2019). Exemplu de optimizatori: gradient descent ( întâlnit mai ales în algoritmii de clasificare), Adagrad (modifică rata de învățare). Cel mai folosit optimizator a fost Adam pentru că se micșorează puțin din viteza pentru o căutare mai minuțioasă și mai atentă, producând astfel unele dintre cele mai bune rezultate comparativ cu ceilalți optimizatori, însă având un cost computațional mare.

Conform (Geron, 2019), funcția de dropout este o tehnică foarte des întâlnită în rețelele neuronale. Ele ajută la creșterea acurateței unei rețele cu în jur de 1-2%. Pentru fiecare pas din antrenament, orice neuron are o probabilitate de abandon (notată cu p). La rețelele convoluționale, p este cuprins între 40 și 50%. Funcția de dropout se folosește de această probabilitate, putând scoate complet un neuron la un anumit pas al antrenării. Totuși, neuronul poate fi activat la unul dintre următorii pași de antrenare.

Augmentarea reprezintă o metodă de expandare a bazei de date pe care rețeaua neuronală o folosește (Geron, 2019). Scopul acesteia este de a crește volumul de date existent și odată cu el, și performanța modelului. Prin augmentare, se înțelege copierea datelor (în cazul de față, datele sunt imagini) și modificarea lor. Exemple de modificări: tăierea imaginii, rotirea, afișarea în oglindă, trecerea de la color la alb-negru, mărirea/micșorarea sau estomparea.

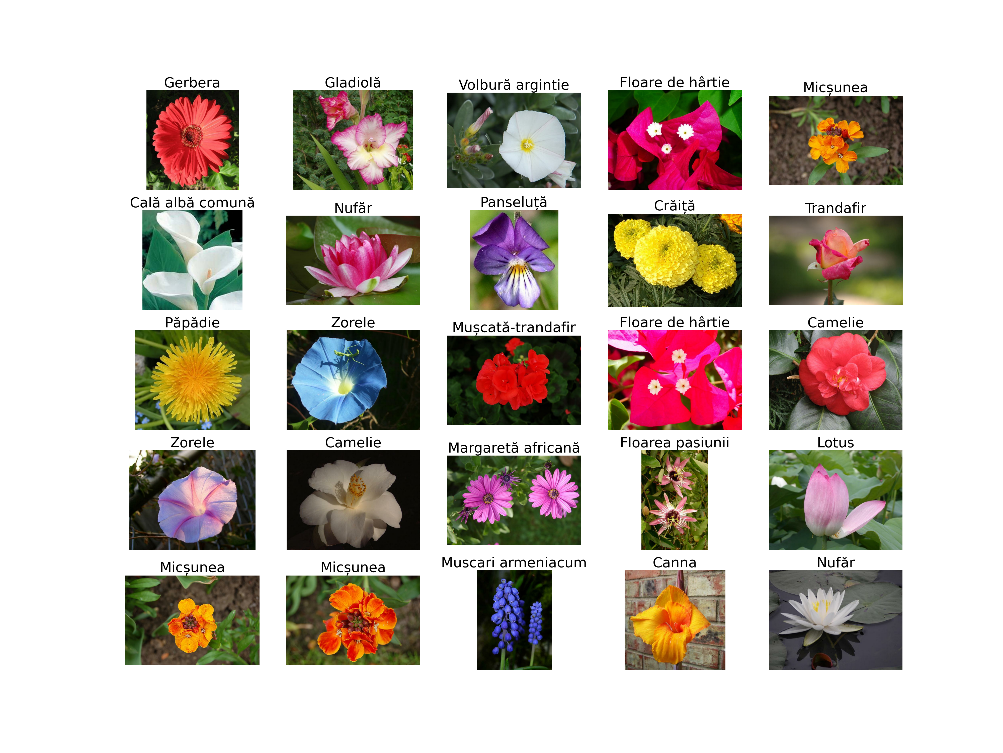
Pentru a putea evalua mai bine performanța unui program, se poate folosi o matrice de confuzie. Scopul ei este de a spune de câte ori o clasa de elemente este confundată cu o altă clasă de elemente. Este construită astfel: o matrice pătratică inițializată cu 0 ce are pe linii așezate clasele unui anumit model. Pe coloane, are aceleași clase, în exact aceeași ordine. În momentul în care clasa A este recunoscută ca fiind clasa B, valoarea de pe linia A coloana B crește cu 1. Așadar, cu cât avem numere mai mari pe diagonala principală a matricei, cu atât algoritmul confundă mai puțin clasele.

Funcția de loss este indicatorul care arată cât de departe este valoarea estimată față de cea reală. În cazul de față, s-a folosit funcția de loss SparseCategoricalCrossentropy pentru că am ales să reprezentăm clasele ca numere, nu ca vectori în care poziția numărului 1 este clasa prezisă, așa cum este în cazul funcției CategoricalCrossentropy. Formula funcției CategoricalCrossentropy este:

, unde este adevărata clasă, iar este clasa prezisă.

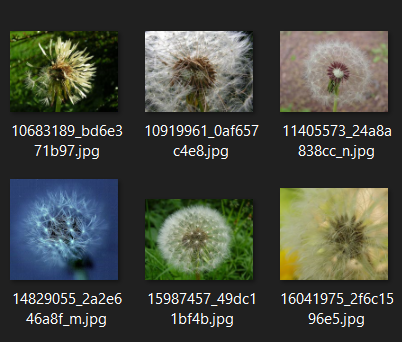
# Capitolul II. Dezvoltarea rețelelor pe desktop

Pentru început, s-a folosit o bază de date existentă în TensorFlow, și anume oxford\_flowers102. Baza de date cuprinde 102 clase cu plante, fiecare clasă conținând între 40 și 258 de poze. Imaginile sunt la diferite scalări sau variate tipuri de luminozități și cu echilibru al compoziției (unele categorii au plante foarte variate, pe când alte categorii au foarte multe poze cu plante asemănătoare). Setul este împărțit în 3 baze de date mai mici: una este pentru antrenare, una pentru validare și cealaltă pentru testare. Pentru setul de date de antrenare și pentru cel de validare, sunt extrase câte 10 fotografii din fiecare categorie, în timp ce pentru setul de testare se folosesc minim 20 de poze pe categorie.



Figură II‑1 Imagini din baza de date de la Oxford

Pentru a putea face o comparație, s-a făcut un studiu și pe o bază de date descărcată de pe (Mamaev, 2021) . Setul conține 5 categorii de plante: margarete, păpădii, trandafiri, floarea-soarelui și lalele și o categorie cu fotografii de fundal. Fiecare specie are în jur de 800-1000 de imagini, cu poze în mare măsură color (există pe alocuri și fotografii alb-negru) și de dimensiuni diferite. Pozele din acest set de date sunt preluate de pe Flickr, un site special pentru fotografii, folosindu-se o etichetă pentru fiecare categorie de imagini. Categoria de fundal a fost adăugată pentru cazul în care se cere clasificarea unei fotografii care nu conține o plantă sau care conține o plantă necunoscută.



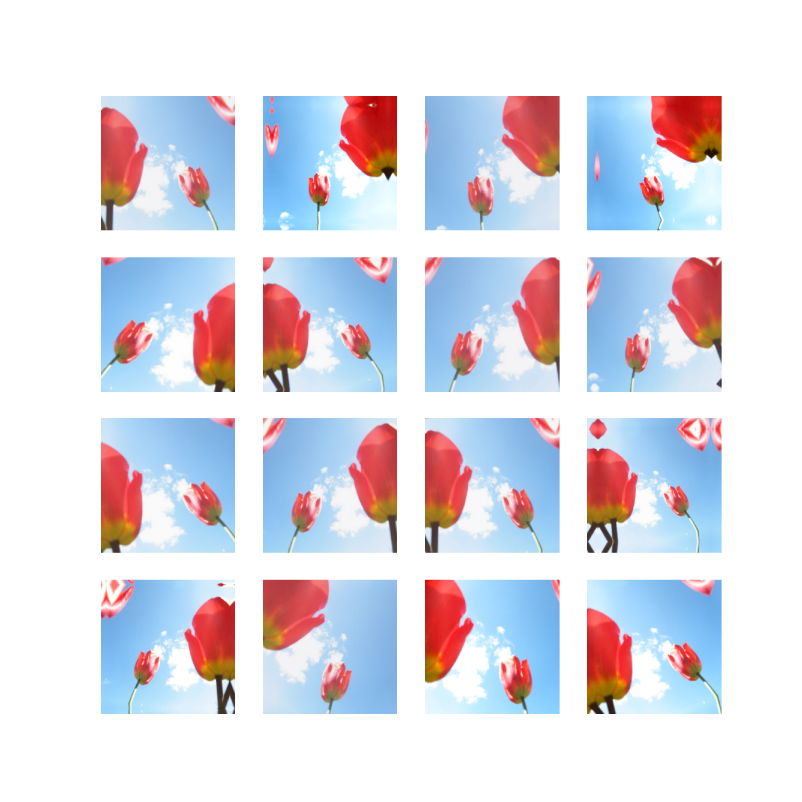
Figură II‑2 Imagini din baza de date descărcată de pe Flickr

## Prelucrarea datelor

Pentru a eficientiza procesul de învățare și pentru a “mări” setul de date disponibil, imaginile ce vor fi introduse în cele două rețele neuronale vor trece printr-un proces de normalizare și augmentare. Pentru normalizare, fiecare pixel a fost împărțit la 255 pentru a avea valori cuprinse între 0 și 1. La capitolul augmentări, am modificat pozele astfel încât să treacă de la color la alb-negru, am mărit sau micșorat contrastul cu o valoare aleatorie, am aplicat operația de zoom-in/zoom-out, am rotit și am modificat orientarea în oglindă, de la stânga la dreapta. Am observat că acuratețea a scăzut în cazul folosirii augmentării alb-negru, așa că am decis ca în varianta finală a programului să nu folosesc această augmentare. Am considerat că transformarea pozelor în alb-negru nu avantajează rețeaua, deoarece aceasta se ghidează printre alte caracteristici, și de culorile plantelor. Parametrii folosiți pentru augmentări sunt:

* Rotația pozei:
* Zoom:
* Contrast:

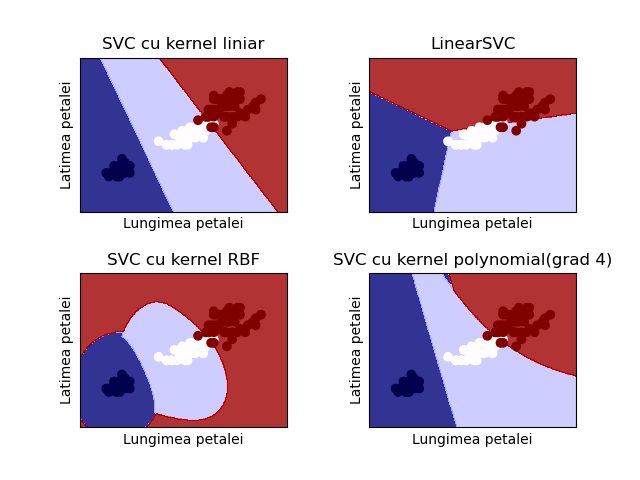
În Figura II‑3 sunt prezentate augmentările menționate mai sus aplicate asupra unei fotografii cu lalele.



Figură II‑3 Imagini cu augmentări

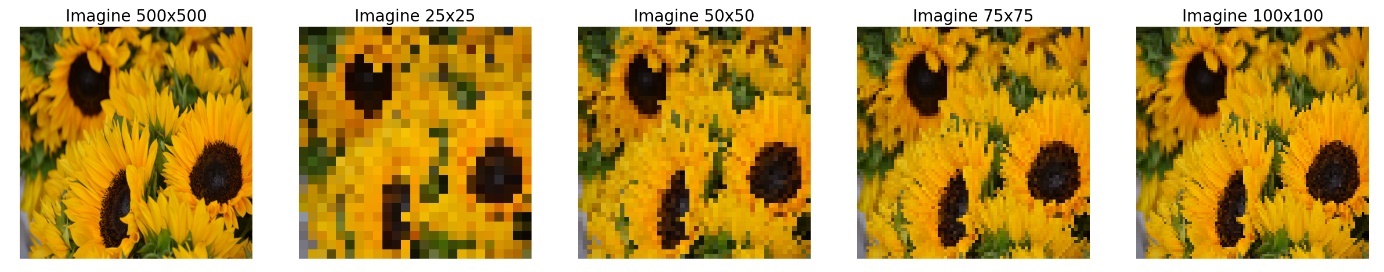
## SVM (Support Vector Machine)

Unul dintre modelele de învățare pe care le-am comparat este SVM, un model care împreună cu algoritmii de învățare, poate clasifica liniar sau neliniar. Este în general folosit în clasificarea datelor de dimensiuni mici sau mijlocii. În Figura II‑4 sunt prezentate diferite tipuri de SVM pe setul de date Iris (Fisher, 1936) ce conține trei clase. Am extras doi parametri: primul parametru este lungimea petalei, iar al doilea parametru este lățimea petalei. În acest program se va folosi SVM RBF, prezentat și în fotografia din stânga jos a figurii Figura II‑4.



Figură II‑4 Diferite tipuri de SVM

Pentru compararea cu o abordare diferită de rețelele neuronale, am ales să folosesc și SVM. Am stabilit, pentru început câteva detalii și caracteristici pe care am dorit să le aibă acest model: am testat 3 dimensiuni de intrare pentru poze (50x50, 75x75 și 100x100 ), cu 3 canale. După cum se poate vedea în Figura II‑5, reducerea dimensiunii imaginii sub 75x75, creează poze greu de recunoscut, însă la dimensiuni mari, SVM nu este optimizat ca rețelele neuronale pentru a înțelege anumite caracteristici și are scor chiar mai mic.

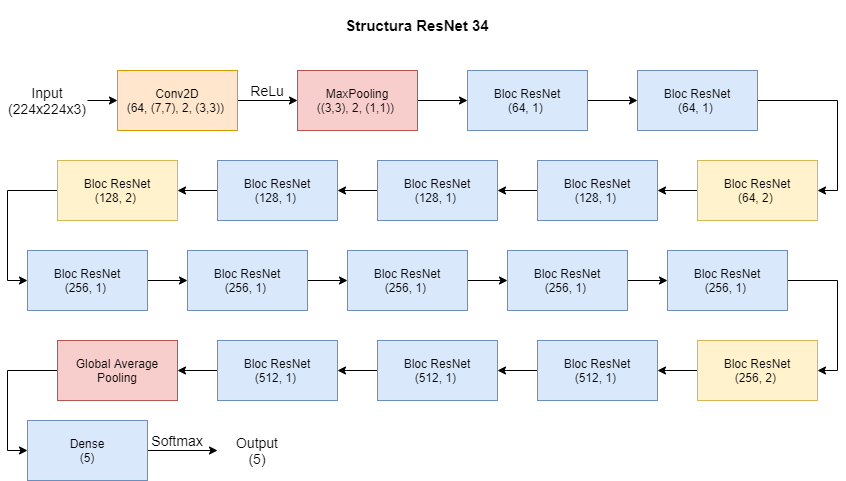


Figură II‑5 Imagini de diferite dimensiuni

Pentru a introduce datele în program, au fost citite pozele, categorie cu categorie într-un vector ce reține și căile lor, apoi au fost redimensionate la dimensiunile menționate anterior. Pentru a nu prelucra de fiecare dată datele de pe hard disk, am salvat varianta prelucrată, folosind funcția de salvare din Numpy. La rulări ulterioare, doar se încarcă datele gata prelucrate, folosind funcția Load. Baza de date a fost împărțită 80% pentru antrenare și 20% pentru testare. Pentru a găsi parametrii ce vor da o acuratețe cât mai bună, am folosit funcția GridSearchCV. Am rulat cu multiple valori, atât pentru parametrul gamma, cât și pentru parametrul C. Valorile încercate pentru gamma sunt 0.0001, 0.001, 0.1 și 1, iar pentru C sunt 1, 10 și 100. Am folosit biblioteca sklearn (Scikit-learn developers, 2020) pentru a putea utiliza funcțiile SVM (RBF) și GridSearchCV, precum și pentru a separa datele în antrenare și testare.

## ResNet

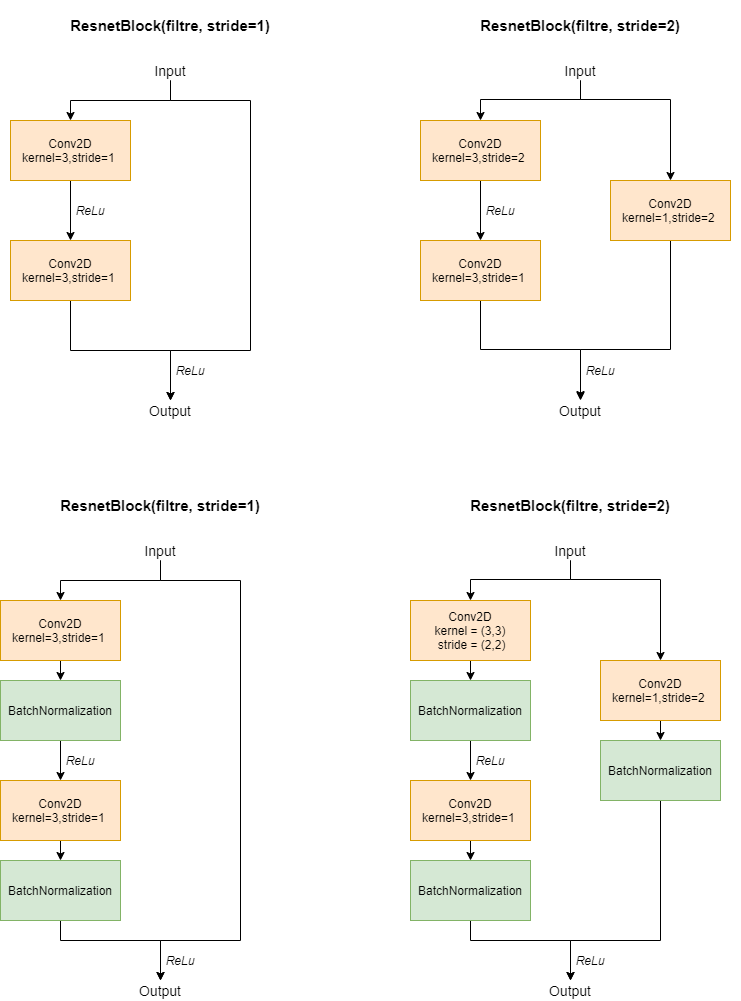
Particularitatea ResNet-ului conform (Geron, 2019) este de a avea conexiuni de tip skip. Skipurile sunt conexiuni de scurtătură, ajutând la parcurgerea în salturi a rețelei. Cu alte cuvinte, inputul de pe un anumit strat este legat de outputul altui strat, diferit de cel anterior. O rețea neuronală modelează o funcție f(x). Să presupunem că adăugăm un input x la ieșirea din rețea. Atunci, rețeaua va trebui să modeleze funcția f(x)-x, acest proces fiind denumit învățare reziduală. Ceea ce o face să aibă viteză de învățare mare este faptul că poate face skipuri fără a fi nevoie ca straturile sărite să înceapă deja să învețe.



Figură II‑6 Structura ResNet

Am început, așadar, cu antrenarea pe cele două baze de date menționate mai sus, prin folosind rețeaua neuronală ResNet.

Deoarece ResNet conține mai multe straturi identice, pentru a se evita scrierea repetitivă, a fost realizat un bloc rezidual care mai apoi va putea fi folosit cu ușurință pentru diferite tipuri de ResNet. Astfel, sunt implementate două cazuri: straturile principale și straturile skip. Pentru ca inputurile să treacă prin straturile principale și cele skip, se folosește o funcție care particularizează fiecare dintre cele două cazuri. La final, se adaugă outputurile și funcția de activare. Straturile principale sunt formate din: o convoluție2D cu un număr parametrizat de filtre, 3 canale, 2 pași pentru blocurile cu un număr diferit de filtre de intrare/ieșire, 1 pas altfel. Urmează un strat de BatchNormalization, apoi activare ReLu. Apoi, tot o convoluție2D cu un număr parametrizat de filtre, 3 canale, 1 pas și un strat de BatchNormalization. Dacă pasul este mai mare decât 1, straturile skip conțin o convoluție2D cu un număr parametrizat de filtre, 3 canale, 2 pași pentru blocurile cu un număr diferit de filtre de intrare/ieșire, iar apoi tot un strat BatchNormalization. Pentru procesarea inputului de către un bloc ResNet se vor apela secvențial straturile principale, iar pentru cazul în care avem straturi skip se vor apela și acestea cu inputul, la final outputul fiind format din activarea sumei outputului straturilor principale și skip.

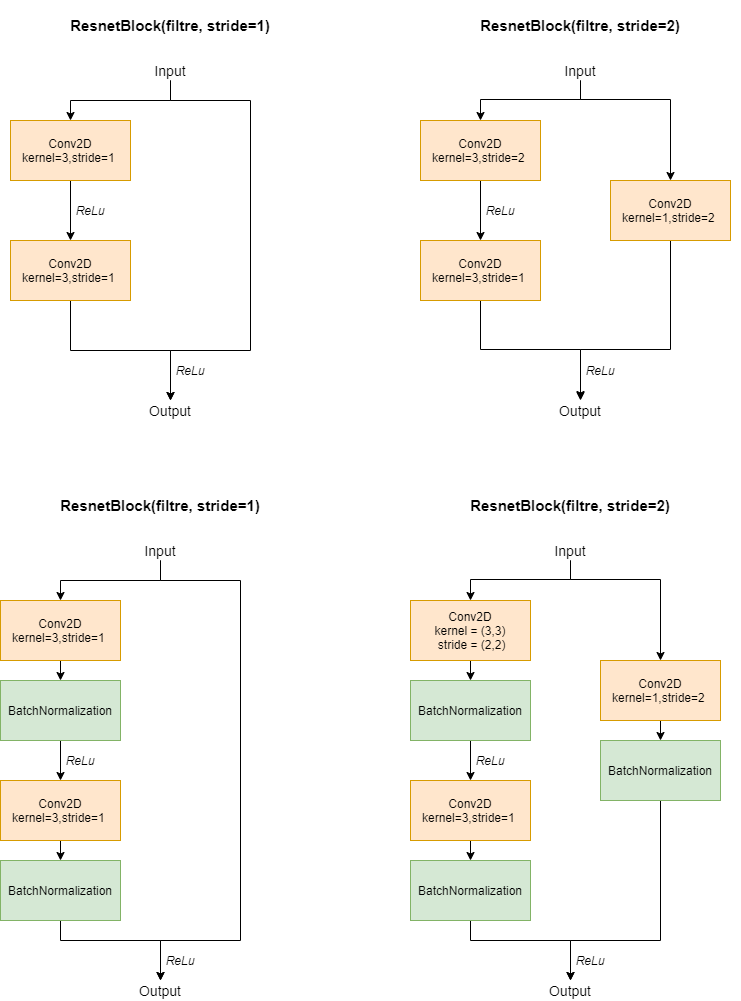


Figură II‑7 Structură bloc rezidual fără Batch normalization pentru stride=1 (stânga) și stride=2 (dreapta)

În Figura II‑7, în stânga se poate observa structura blocului rezidual cu stride=1 folosit pentru a păstra dimensiunea inputului, în timp ce în partea dreaptă este ilustrat blocul rezidual cu stride=2, folosit pentru a reduce la jumătate dimensiunea datelor de intrare.

Primul pas în antrenarea rețelei este citirea datelor pentru învățare și validare și aducerea lor într-un format potrivit. Se extrage din director baza de date, având etichete de tip int (prima clasă are asociat numărul 0, a doua clasă are numărul 1 ș.a.m.d.), culorile pozelor sunt de tipul RGB, imaginile sunt redimensionate la 224x224. Am testat diferite dimensiuni și un input de 224x224 aduce rezultate bune în timp ce păstrează rețeaua de dimensiune potrivită. Baza de date este folosită în proporție de 80% pentru antrenare și 20% pentru validare.

În Figura II‑8 este prezentată structura blocului rezidual folosind stratul de Batch normalization.



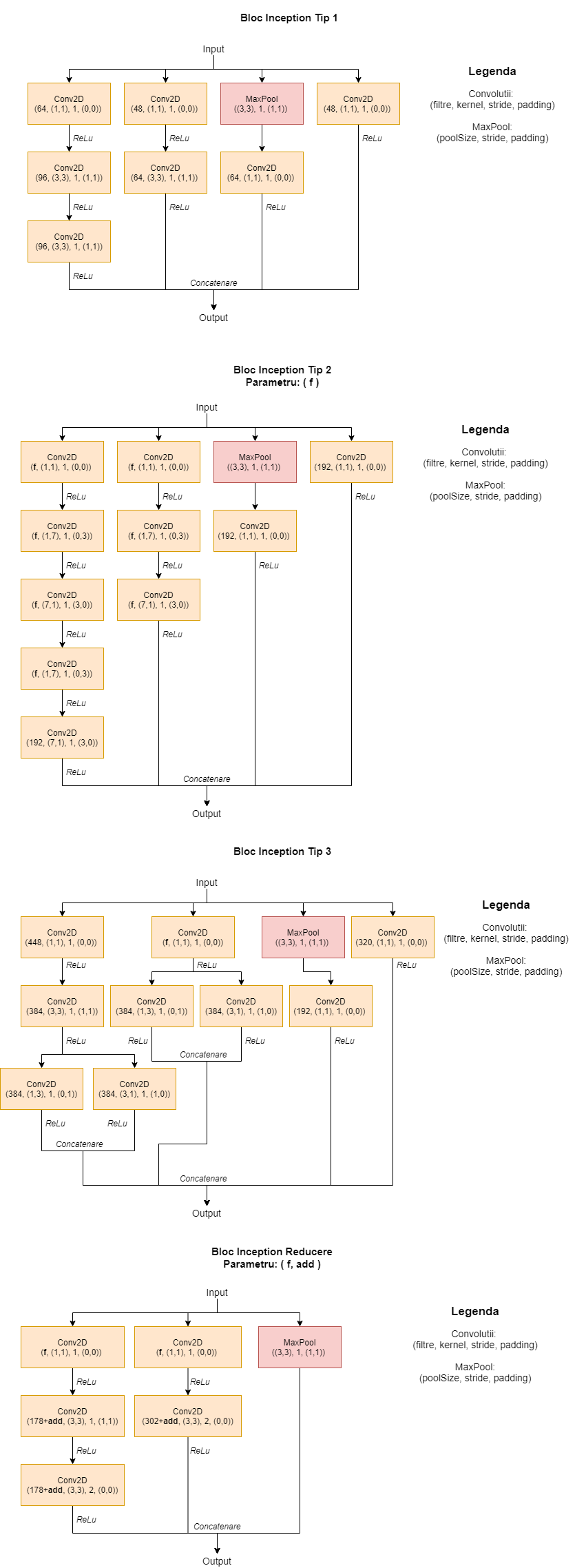
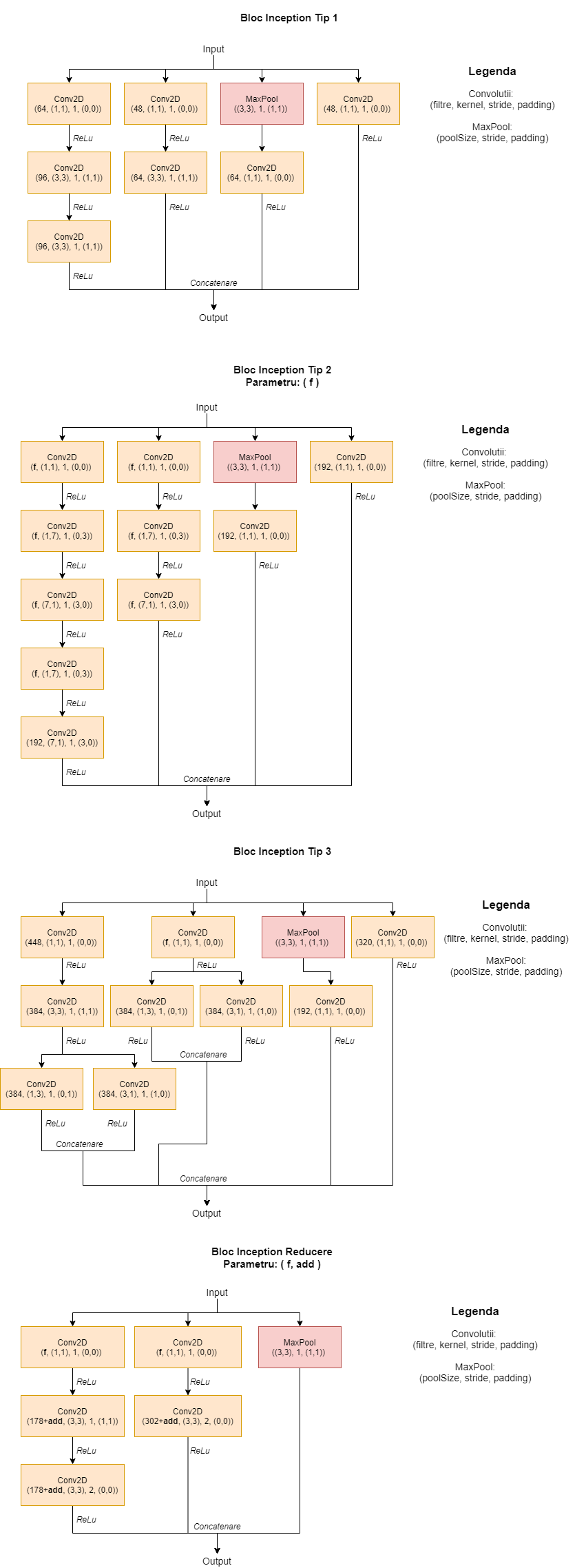
Figură II‑8 Structură bloc rezidual cu Batch normalization pentru stride=1 (stânga) și stride=2 (dreapta)

Rețeaua în sine a fost implementată astfel: am adăugat un strat convoluțional2D cu 64 de filtre, kernel de dimensiune 7x7 și stride de dimensiune 2x2, dimensiunea inputului este de 224x224 cu 3 canale. Pentru a nu reduce dimensiunea datelor de intrare de către convoluții cu kernel diferit de 1, stratul va folosi padding, adăugând pixeli de culoare neagră. Se adaugă activarea ReLu și un strat de MaxPooling cu dimensiunea de 2x2. Am creat apoi un vector cu mai multe valori de filtre. Am testat diferite valori, am eliminat și adăugat filtre, îndepărtându-ne puțin de varianta originală de ResNet34. Următorul pas a fost să parcurgem vectorul, pentru fiecare pas făcându-se următoarele verificări: dacă nu suntem la prima valoare din vector și dacă valoarea filtrului este diferită de valoarea filtrului de dinaintea sa, apelăm blocul rezidual cu filtrul curent și stride 2, altfel se apelează blocul rezidual cu filtrul curent și stride 1. După ce este parcurs acest for, se adaugă un strat de pooling de tip GlobalAveragePooling2D, un strat Flatten iar la final, un strat dens de dimensiune egală cu numărul de clase și cu activare SoftMax. După mai multe încercări de a găsi un număr bun de epoci, tot procesul de învățare a fost organizat în 35 de epoci. Am folosit funcția de loss SparseCategoricalCrossentropy și optimizatorul Adam.

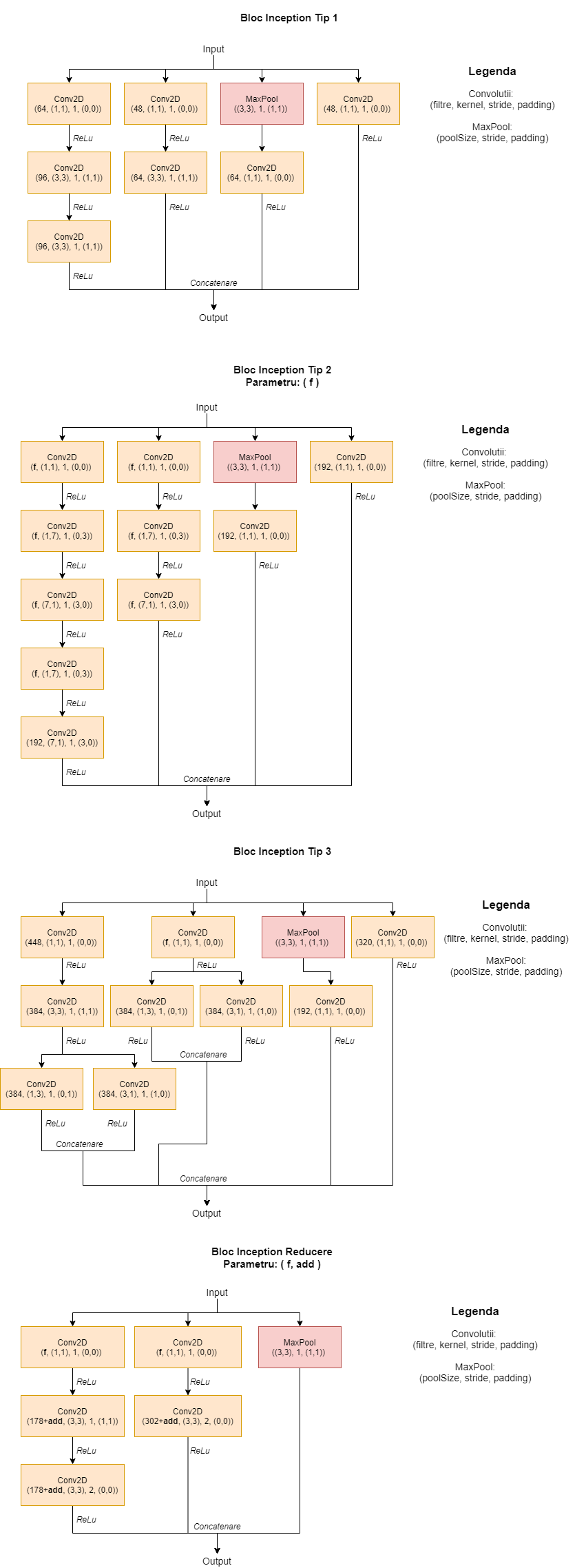
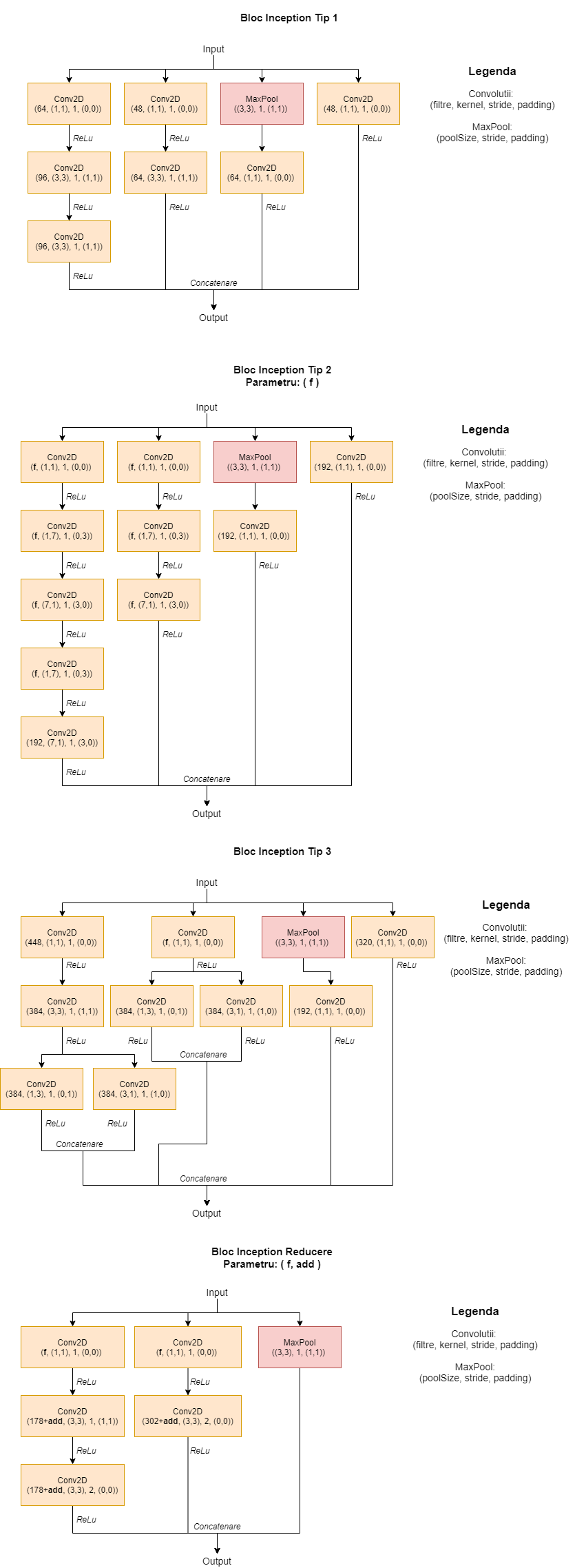
Pentru a putea verifica progresul și a valida mulțimea de parametrii folosiți ( numărul de epoci, numărul de straturi, rata de învățare, etc.), am păstrat într-un istoric datele de învățare. Pentru a putea vedea confuzia dintre clase, am făcut o matrice de confuzie din care se poate observa capacitatea programului de a diferenția categoriile.

## InceptionV2

La fel ca în cazul implementării rețelei ResNet, am ales să implementăm patru blocuri pe care le vom putea folosi ulterior în diferite structuri care se repetă în cadrul rețelei Inception. Trei dintre cele patru blocuri sunt blocuri cu diferite căi și abordări pentru învățare, iar ultimul bloc este un bloc folosit special pentru reducerea dimensiunii inputului odată cu mărirea numărului de filtre. Toate aceste blocuri folosesc mai multe drumuri care vor forma prin concatenare outputul. În implementare am separat fiecare astfel de drum folosind vectori și le-am unit folosind stratul Concatenate din Keras. În Figura II‑9 și Figura II‑10 se poate observa structura celor patru blocuri utilizate. Sub fiecare strat sunt notate toate detaliile necesare implementării. În partea dreaptă a primei figuri este o legendă valabilă pentru toate blocurile.



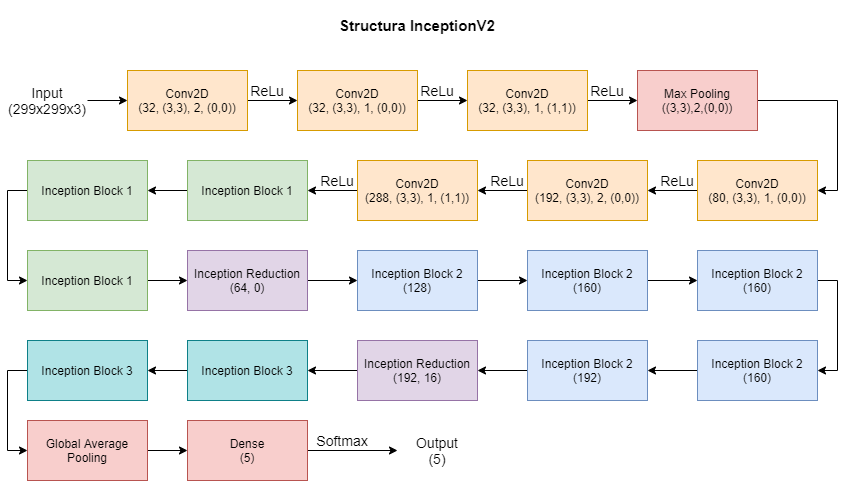
Figură II‑9 – Bloc Inception tip 1 și tip 2



Figură II‑10 – Bloc Inception tip 3 și Bloc Inception Reducere

Pentru citirea datelor din baza de date și aducerea în forma potrivită, am folosit aproximativ aceleași tehnici ca la ResNet. Am citit datele din directorul cu poze, i-am asociat fiecărei clase un număr întreg ce începe de la 0 și crește pentru fiecare clasă nouă, pozele sunt color și dimensiunile sunt de 299x299. Partea de început cuprinde câteva straturi convoluționale astfel: un strat convoluțional2D cu 32 de filtre, dimensiunea inputului de 299x299 și cu 3 canale, dimensiunea kernelului de 3x3 și stride 2; un strat convoluțional2D cu 32 de filtre, dimensiunea kernelului de 3x3 și stride 1; un strat convoluțional2D cu 64 de filtre, dimensiunea kernelului de 3x3, stride 1 și padding; un strat de MaxPooling cu dimensiunea de 3x3 și stride 2; un strat convoluțional2D cu 80 de filtre, dimensiunea kernelului de 3x3 și stride 1; un strat convoluțional2D cu 192 de filtre, dimensiunea kernelului de 3x3 și stride 2 și un strat convoluțional2D cu 288 de filtre, dimensiunea kernelului de 3x3, stride 1 și padding. Apoi continuăm cu 3 blocuri de tip 1 (Figura II‑9), urmând să reducem dimensiunea cu ajutorul blocului de reducere. Continuăm cu 5 blocuri de tip 2 (Figura II‑9), reducem dimensiunea și în final, 2 blocuri de tip 3 (Figura II‑10). La final, adăugăm un strat de pooling, folosind funcția GlobalAveragePooling2D, un strat Flatten și un strat dens cu activare Softmax. Structura aceasta poate fi observată și în Figura II‑11. Modelul se compilează cu funcția de loss SparseCategoricalCrossentropy și optimizatorul Adam.

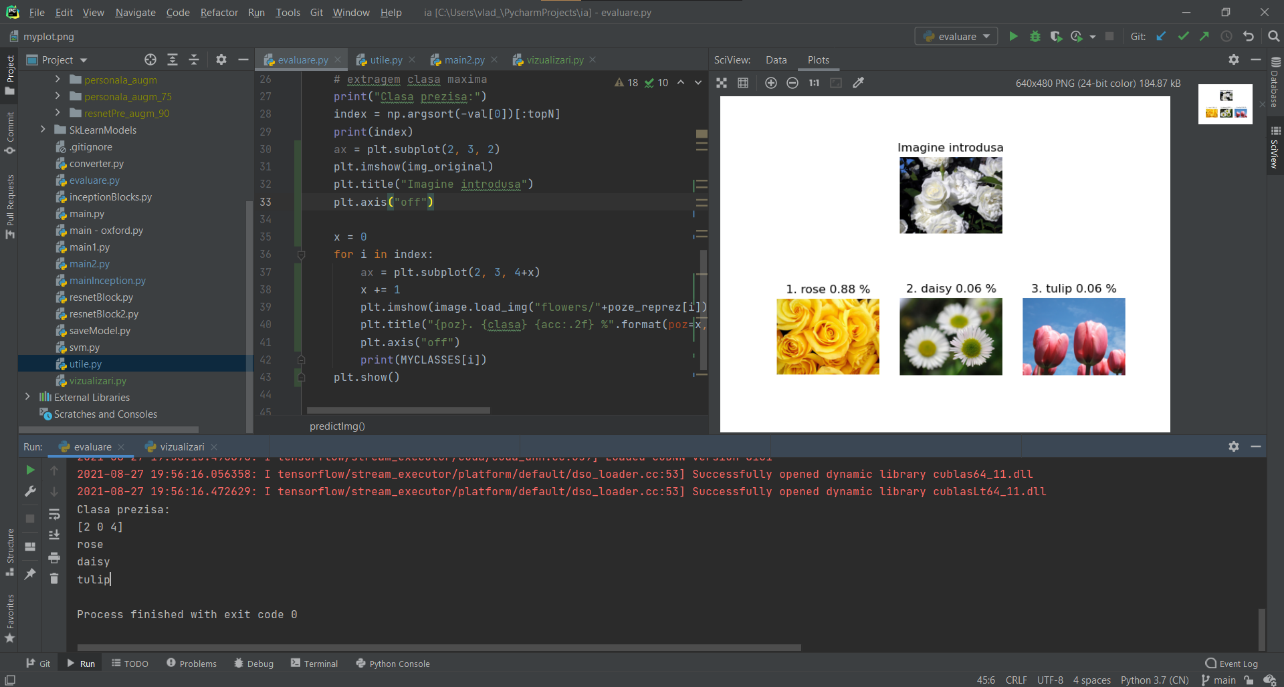
Precum la rețeaua anterioară, se urmărește evoluția algoritmului prin intermediul istoricului ce prezintă funcția de loss și acuratețea și prin intermediul matricei de confuzie pentru capacitatea de diferențiere a claselor.



Figură II‑11 Structura InceptionV2

## Evaluarea

Scopul evaluării este de a testa dacă algoritmii de recunoaștere clasifică bine, înainte de a folosi aplicația de Android. Astfel, programul citește dintr-un director cu adresa bine definită denumirea pozei pe care o dorim clasificată. Poza poate fi de orice fel, de pe internet sau făcută personal cu telefonul și cu orice tip de extensie. Imaginea este transformată într-un vector, normalizată, iar cu ajutorul modelului de rețea salvată, se va face predicția. Din vectorul de probabilități, extragem descrescător primele trei clase cu cea mai mare probabilitate de a fi ceea ce este în imaginea evaluată. Un exemplu de evaluare se poate observa în Figura II‑12.



Figură II‑12 Rezultat evaluare poză

## Exportul modelului și pregătirea pentru aplicația de Android

Am ales să exportăm modelul rețelei ResNet și InceptionV2. Am salvat modelele care aveau cea mai bună acuratețe și le-am convertit în rețele mai mici, speciale pentru a rula pe dispozitive mobile. Pentru convertire am folosit funcția TFLiteConverter din TensorFlow. Totuși, în funcție de complexitatea rețelei, chiar dacă va avea o acuratețe mai bună, de multe ori aceasta are automat și mai mulți parametrii, rezultând o dimensiune mai mare a modelului. Acest lucru nu este neapărat favorabil, deoarece pe telefon avem nevoie de dimensiuni și complexitate a calculelor cât mai mici. Astfel, acuratețea nu este întotdeauna un criteriu eliminatoriu în alegerea modelului exportat pe Android.

# Capitolul III. Aplicația de Android

## Dezvoltarea programului

Aplicația de Android de recunoaștere a plantelor a fost scrisă în limbajul JAVA și conține un ansamblu de rețele utilizate pentru clasificare. În momentul deschiderii aplicației, am realizat un SplashScreen care să dureze două secunde. Apoi, va apărea meniul principal, SplashScreenul închizându-se. Meniul are o fotografie de fundal făcută din imagini din interiorul bazei de date folosite de noi și conține 5 butoane, iar în momentul în care se apasă pe ele, programul va deschide ferestrele, folosind referințe către paginile respective.

În pagina de clasificare, în momentul apăsării butonului “Poză”, se va construi un dialog cu două opțiuni: opțiunea de a deschide camera foto sau cea de a alege o fotografie din galeria telefonului. Orice acțiune s-ar alege, vom verifica înainte ca aplicația să aibă drepturile necesare. În cazul în care nu are, se vor cere aceste drepturi, iar utilizatorul le va accepta (Anexă 2). În funcție de metoda de introducere aleasă de utilizator, încărcarea și afișarea pozei se va face în moduri diferite. Dacă se alege fotografierea, aplicația va avea calea și drepturile de a deschide poza creată pe o perioadă nedeterminată, astfel că este suficient să o afișăm folosind Uri-ul acesteia. Pentru cazul în care se alege o imagine din galerie, sistemul de operare ne va da acces prin Uri doar în rularea curentă a aplicației. Din acest motiv, trebuie să cerem drepturi permanente folosind:

final int parametri = date.getFlags()  
 & (Intent.*FLAG\_GRANT\_WRITE\_URI\_PERMISSION* | Intent.*FLAG\_GRANT\_READ\_URI\_PERMISSION*);  
this.getContentResolver().takePersistableUriPermission(uriImagine, parametri);

Astfel, la fiecare deschidere a aplicației, vom putea deschide și afișa pozele în cadrul acesteia.

Pentru clasificare, în momentul apăsării butonului “Clasifică imaginea”, fotografia aleasă va fi convertită într-un Bitmap care va trece prin aceleași preprocesări ca în cazul antrenării pe desktop (redimensionare la 224x224/299x299 și normalizare la 255). Este foarte important ca preprocesarea să fie făcută în același fel în procesul de recunoaștere și în cel de antrenare. De exemplu, dacă normalizarea folosește o deviație sau o medie diferită, datele de intrare nu vor avea sens pentru rețea și clasificarea va fi greșită. Pentru a putea da utilizatorului opțiunea de a selecta rețeaua ce va recunoaște planta, s-a implementat o clasă ce conține datele necesare și funcțiile de procesare particularizate pentru fiecare dintre rețelele disponibile în aplicație. Astfel, pentru a clasifica imaginea trebuie doar să construim clasa ținând cont de alegerea utilizatorului făcută în pagina de opțiuni. După clasificare, vom sorta descrescător vectorul care conține probabilitățile categoriilor, ținând cont de indexul fiecărei clase. Vom extrage primele trei valori din vectorul sortat și vom afișa pentru fiecare dintre aceste clase, o imagine reprezentativă ( salvată în resursele drawable ).

Pentru a salva imaginea în galerie, va trebui să reținem calea, data curentă și clasa acesteia. Am construit o clasă special pentru păstrarea datelor despre poze și rețeaua aleasă pe care o citesc din memoria destinată aplicației la deschiderea acesteia și pe care o salvez la fiecare modificare adusă, pentru a nu se pierde datele. Pentru partea de ștergere din Opțiuni, se golește acest vector de date.

Rularea continuă a fost dezvoltată pe baza unui exemplu de pagină din documentația TensorFlow Lite și are ca scop testarea vitezei de rulare a diferitelor rețele.

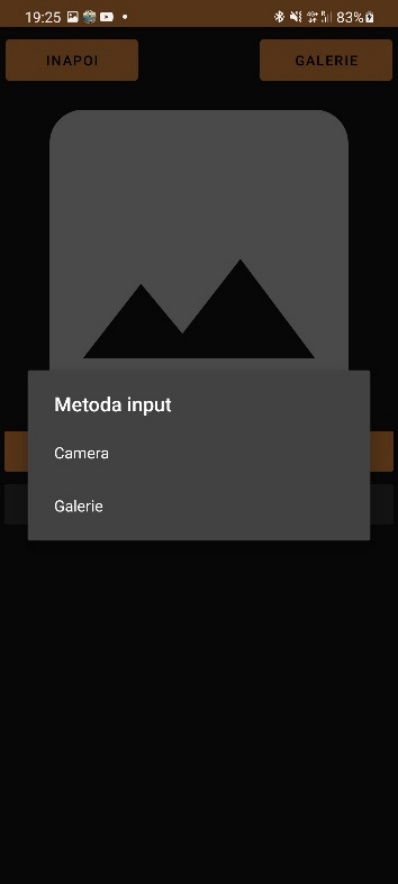
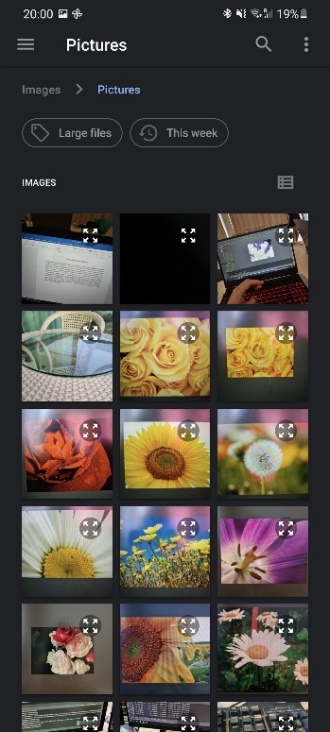
## Interacțiunea cu aplicația

Pentru a putea folosi aplicația pe telefon, este nevoie de un dispozitiv smartphone cu sistemul de operare Android, minim versiunea 6.0. La momentul deschiderii aplicației cu shortcutul din Figura III‑1-stânga, va apărea pentru două secunde un ecran introductiv ce conține logoul și numele autorului precum în Figura III‑1-mijloc. Se va deschide, apoi, pagina principală ce conține meniul aplicației, precum în Figura III‑1- dreapta. Meniul conține cinci butoane, iar dacă se selectează butonul din partea inferioară a aplicației, numit “Ieșire”, se va ieși din aplicație.

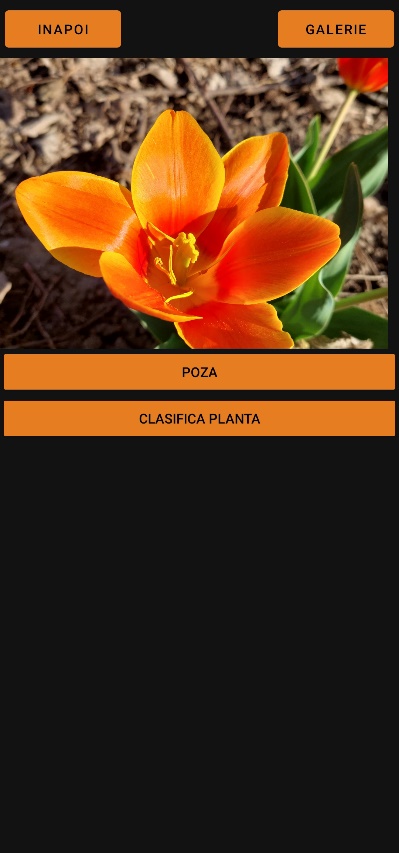
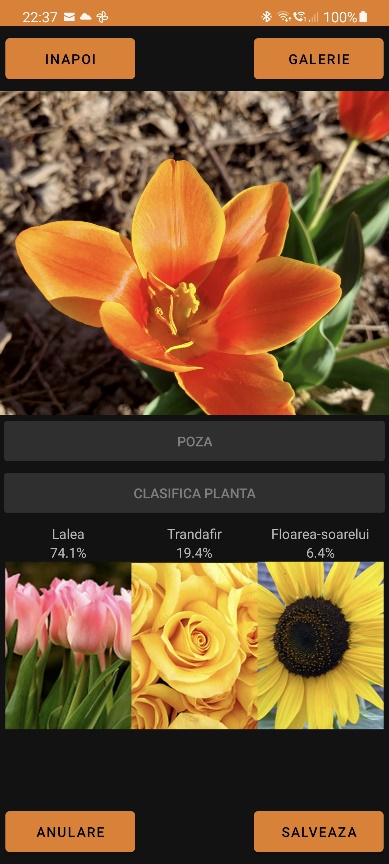
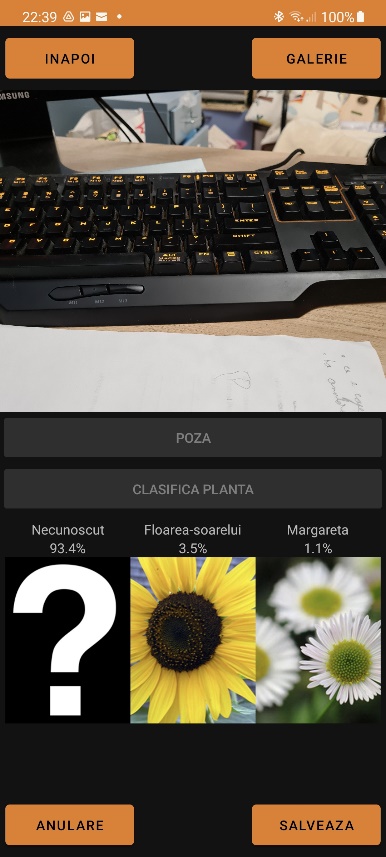
Figură III‑1- Stânga: Shortcut aplicația de plante. Mijloc: Ecran introductive. Dreapta: Meniu

În momentul apăsării butonului “Clasifică”, va fi afișată o nouă pagină, exact ca în Figura III‑2- stânga. Se poate observa în partea de sus a dispozitivului un buton în stânga ce trimite utilizatorul înapoi în pagina principală, un buton în dreapta ce va duce în galeria aplicației și dedesubt, o imagine goală, iar sub ea un buton cu denumirea “Poză” și unul cu numele “Clasifică planta”, dezactivat până la apariția unei fotografii. În momentul selectării butonului de poză, se va deschide o fereastră de unde se poate selecta modalitatea de a introduce datele de intrare, după cum se poate vedea și în Figura III‑2- mijloc. Metoda “Cameră” va redirecționa utilizatorul spre camera foto a telefonului. Dacă aplicația este folosită pentru prima dată, i se va cere acordul utilizatorului de a folosi camera dispozitivului mobil. După acceptare, se va deschide aparatul de fotografiat al telefonului și se va face o fotografie care să încadreze în centru obiectul ce se dorește a fi recunoscut.

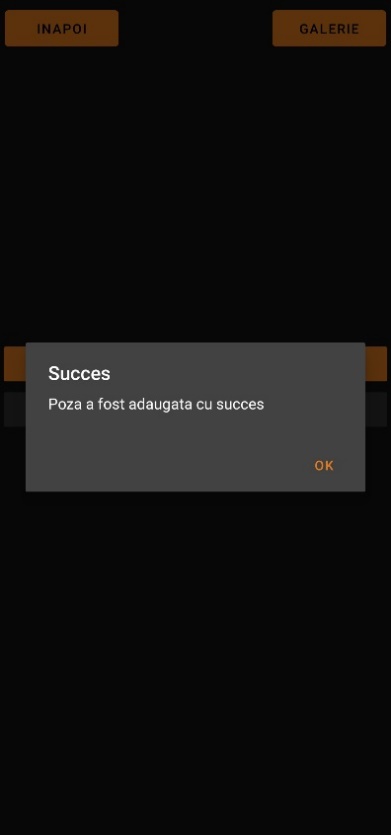
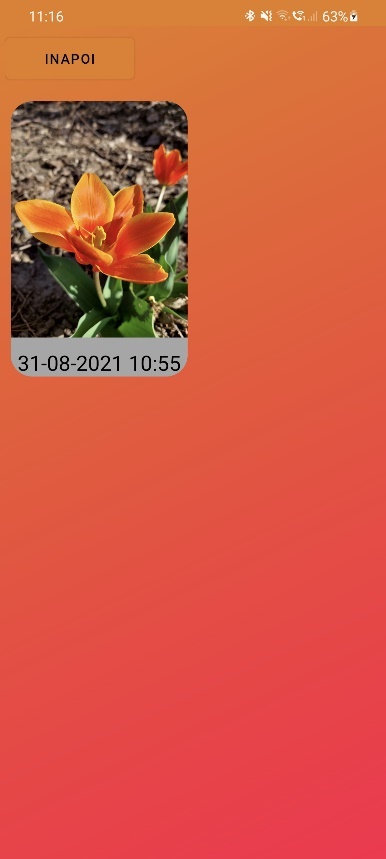
  

Figură III‑2 Stânga: Pagina de clasificare. Mijloc: Metode de introdus datele de intrare. Dreapta: Galeria foto a telefonului.

Există și metoda “Galerie” care reprezintă accesarea galeriei foto a dispozitivului și opțiunea de a selecta imaginea pe care o dorim clasificată, după cum este vizibil și în Figura III‑2- dreapta. Odată aleasă/ realizată o fotografie precum în Figura III‑3- stânga, se poate clasifica imaginea, folosind butonul “Clasifică planta”, iar dedesubt vor apărea în ordine descrescătoare categoriile din care face parte sau de care este cel mai apropiat subiectul fotografiat, alături de probabilitatea de a fi din categoria respectivă și o poză reprezentativă pentru specia prezisă. Un astfel de exemplu se poate observa în Figura III‑3- mijloc. Pentru cazul în care imaginea nu va conține o plantă sau va conține o plantă necunoscută, aceasta va fi încadrată în categoria Necunoscut, după cum se poate observa și în Figura III‑3- dreapta. Mai multe clasificări se pot vedea în Anexa 1. În același timp cu clasificarea apar și două butoane în partea de jos a ecranului. Butonul din stânga este pentru anularea pozei și revenirea la poziția inițială, aceea în care se alege cum se va face poza. Butonul din dreapta este pentru a păstra fotografia și a o adăuga într-o galerie a aplicației, despre care vom discuta în curând.

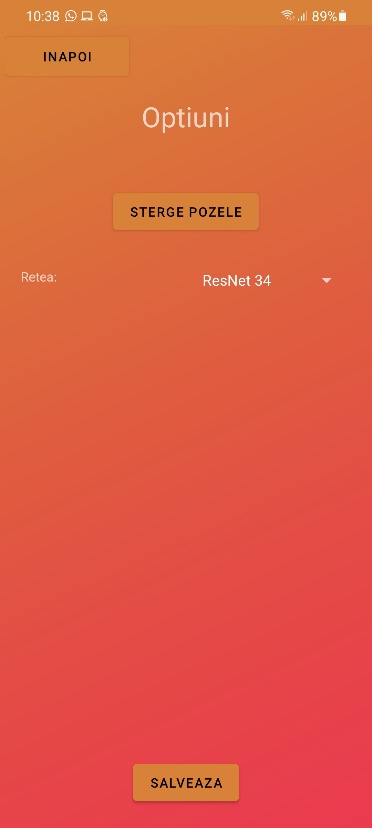
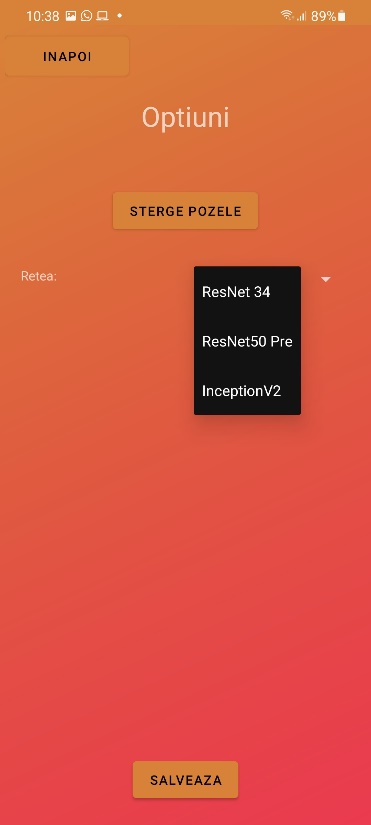
Figură III‑3 Stânga: Imagine aleasă/realizată. Mijloc: Imagine clasificată. Dreapta: Imagine clasificată ca necunoscută.

Figură III‑4 Stânga: Adăugare imagine cu succes. Mijloc: Galerie cu plante recunoscute. Dreapta: Fotografii salvate

După ce fotografia a fost adăugată cu succes, așa cum se poate vedea și în Figura III‑4- stânga, se poate intra în galeria aplicației fie apăsând butonul poziționat în dreapta sus a paginii de clasificare, fie mergând înapoi în meniul principal și selectând butonul “Galerie”. Vor fi afișate mai multe grupulețe ce reprezintă categoriile de fotografii cu plante recunoscute și salvate. Dacă bara unei categorii este gri, înseamnă că acea categorie nu a recunoscut și salvat nicio plantă. În momentul în care este recunoscută și adăugată o fotografie cu o plantă, bara categoriei va deveni verde, iar numărul de fotografii conținute va crește cu 1. Un exemplu concret este în Figura III‑4- mijloc. Selectând una dintre categorii, vom intra într-o nouă pagină, locul în care sunt prezentate toate fotografiile din specia respectivă realizate de către utilizator, cum se poate vedea și în Figura III‑4- dreapta.

Dacă se dorește ștergerea fotografiilor, în meniul principal există butonul de Opțiuni, care dacă este selectat, va deschide o nouă fereastră ce cuprinde câteva alegeri pe care le putem face. În partea de sus a paginii, se poate găsi un buton numit “Șterge pozele” care, așa cum sugerează și numele, elimină din galeria aplicației toate fotografiile, după cum este prezentat și în Figura III‑5- stânga. Tot aici există și opțiunea de a alege rețeaua cu care să fie clasificate plantele, așa cum este prezentat și în Figura III‑5- mijloc. După ce se aleg modificările dorite, se vor păstra apăsând butonul din partea de jos a paginii, numit “Salvează”.

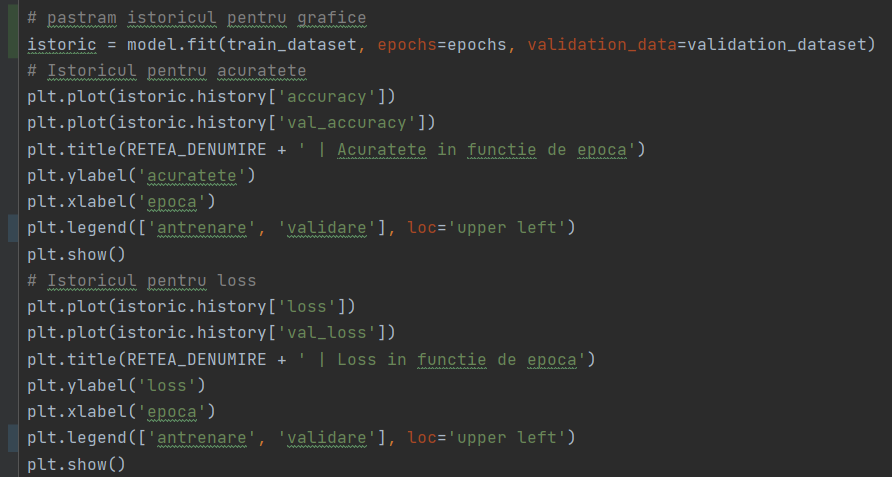
Figură III‑5 Stânga: Pagina de opțiuni. Mijloc: alegerea rețelei pentru clasificare. Dreapta: Rularea continuă

Rularea continuă este pentru cazul în care nu se dorește realizarea unei fotografii, ci doar deschiderea camerei și recunoașterea plantei din fața obiectivului în timp real. Cu ajutorul ei, se poate observa performanța rețelelor utilizate. Interfața paginii de rulare continuă poate fi vizualizată în Figura III‑5- dreapta.

# Capitolul IV. Rezultate și comparații

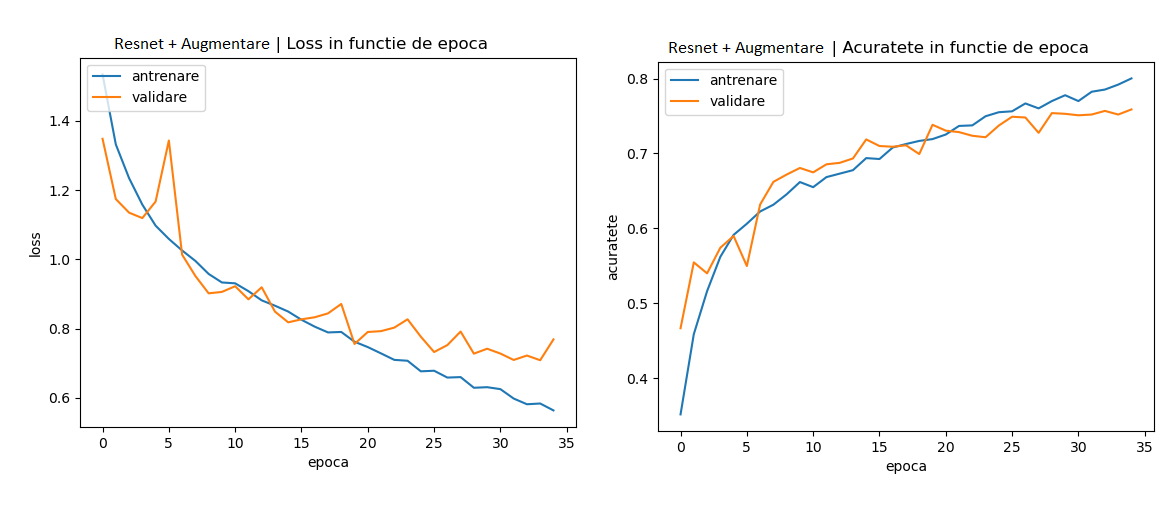
Din studiile efectuate în (Yu, et al., 2017), pentru învățarea bazei de date din campusul Universității Forestiere din Beijing, s-a folosit ResNet26, datele fiind împărțite 80% pentru antrenare și 20% pentru testare, cu inputuri RGB, dimensiunile imaginilor redimensionate la 224x224 pixeli și normalizate la 255. BJFU are pe ResNet18 acuratețe aproximativă de 90%, pe ResNet34 în jur de 88% și pe ResNet50 aproximativ 86%. Mai există o bază de date de frunze denumită FLAVIA, cu exact același algoritm de antrenament ca al Universității Forestiere din Beijing, având cu ResNet18 acuratețe de aproximativ 99%, ResNet26 peste 99%, ResNet34 tot aproximativ 99%, dar totuși mai puțin decât ResNet18, iar ResNet50 în jur de 98%.

În cazul de față, rezultatele vor fi făcute pe baza setului de date format din 5 clase de plante (Mamaev, 2021) și o clasă adăugată de noi pentru separarea imaginilor necunoscute. A fost aleasă această bază de date pentru comparații, deoarece are un număr redus de clase și putem prezenta matricele de confuzie. În cazul setului de date de 102 clase, ar fi foarte dificil să introducem toate datele matricelor de confuzie într-un tabel. Pentru a compara cu ușurință performanțele și rezultatele proiectului, am folosit un istoric care arată în timp real evoluția funcției de loss și a acurateței, așa cum se poate observa în Figura IV‑1.



Figură IV‑1 Istoric

## Efectele augmentării

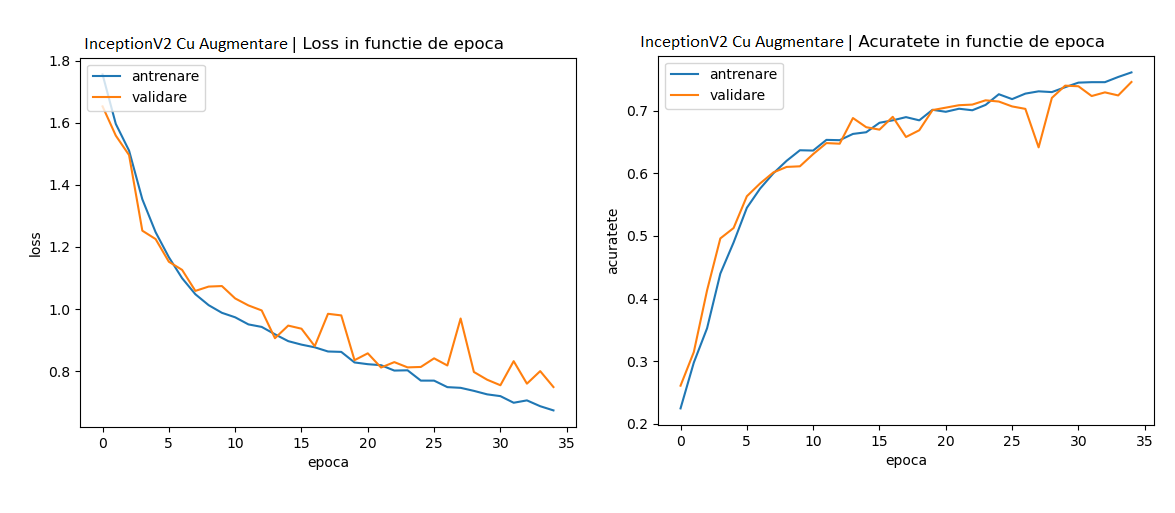


Figură IV‑2 – ResNet funcția de loss și acuratețea cu augmentare

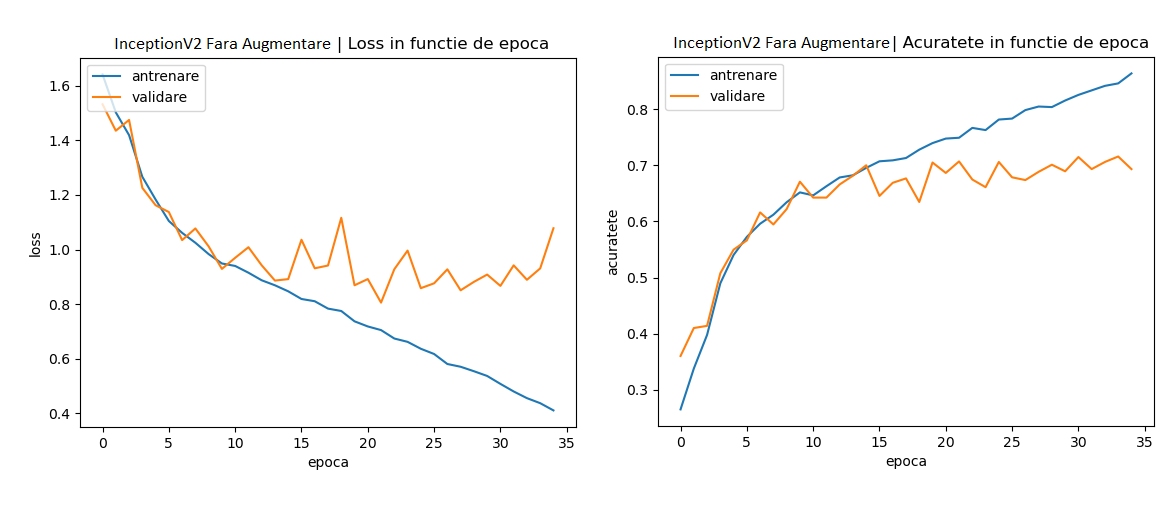


Figură IV‑3 – ResNet funcția de loss și acuratețea fără augmentare

În Figura IV‑2 se poate observa evoluția pentru rețeaua ResNet a funcției de loss și a acurateței în funcție de epocă, atunci când folosim augmentări. În Figura IV‑3 sunt loss-ul și acuratețea fără augmentări. Este limpede că este o diferență foarte mare între cele 2 figuri, primul caz fiind varianta cea mai bună. Augmentarea ajută funcția de loss să scadă și pe antrenare și pe validare, iar acuratețea să crească sigur și constant. Fără augmentare, învățarea atinge un punct în jurul zonei de 20 de epoci, în care lossul crește și acuratețea scade, semnificând tendințe de overfitting.



Figură IV‑4 – InceptionV2 funcția de loss și acuratețea cu augmentare

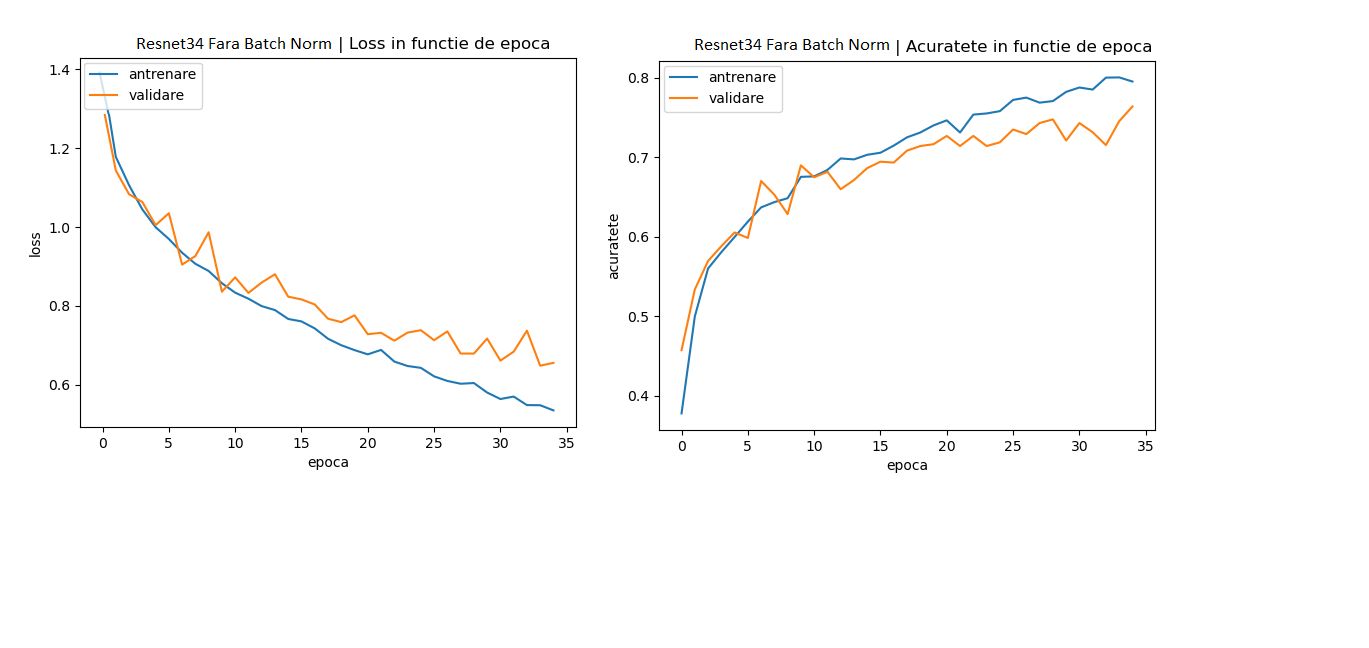


Figură IV‑5 – InceptionV2 funcția de loss și acuratețea fără augmentare

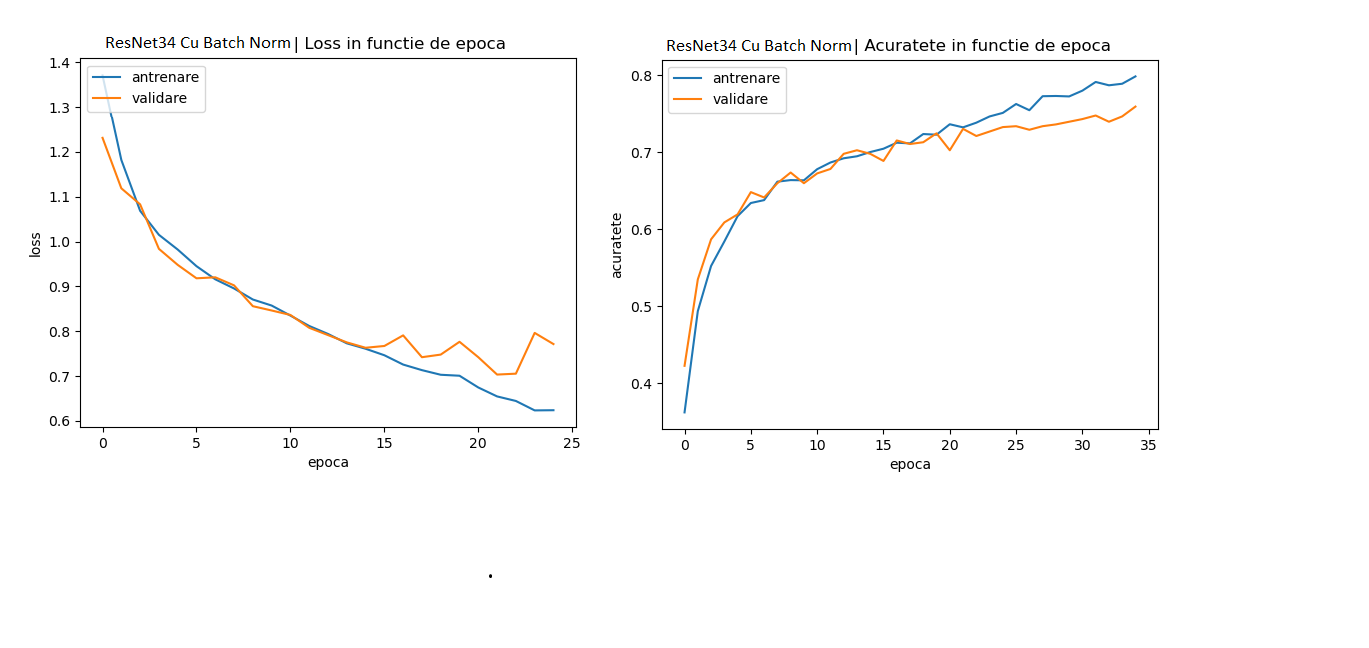
La fel ca în cazul ResNet, InceptionV2 are performanțe mai mari atunci când folosește augmentări, față de atunci când nu le utilizează. În Figura IV‑4 este foarte vizibil faptul că funcția de loss scade constant, fără prea multe oscilații. Aceeași situație este valabilă și în cazul acurateței. Dacă observăm Figura IV‑5, se poate vedea același fenomen de overfitting precum la ResNet, lossul și acuratețea plafonându-se în jurul epocii 10.

## Efectele Batch normalization

Întrucât rezultatele au fost mai bune cu augmentare, comparațiile referitoare la Batch normalization vor fi făcute păstrând această tehnică.



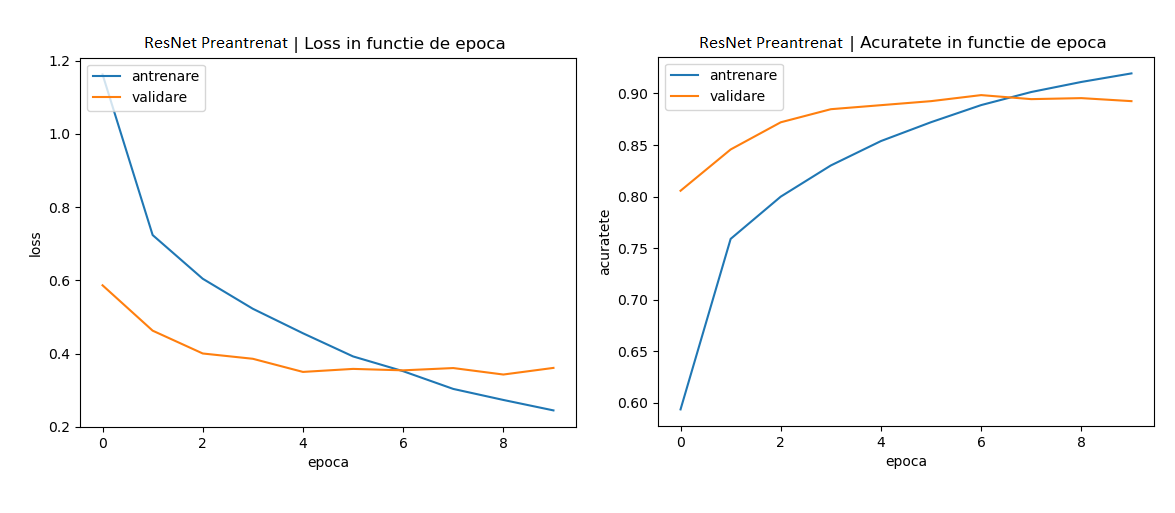
Figură IV‑6 – ResNet funcția de loss și acuratețea fără Batch normalization



Figură IV‑7 – ResNet funcția de loss și acuratețea cu Batch normalization

În Figura IV‑6 sunt ilustrate funcția de loss și acuratețea pentru rețeaua ResNet fără straturi de Batch normalization, iar în Figura IV‑7 cu Batch normalization (Figura II‑8). Se observă că diferențele sunt infime, iar în teste, pe diferite rețele, rezultatele au fost asemănătoare.

## Rețele preantrenate vs. nepreantrenate



Figură IV‑8 – ResNet preantrenat cu funcția de loss și acuratețea

În Figura IV‑7 este prezentat graficul pe care îl parcurge funcția de loss și acuratețea în rețeaua neuronală ResNet nepreantrenat. În Figura IV‑8 se observă traseul funcției de loss și a acurateței pentru ResNet preantrenat. În cazul rețelei nepreantrenate, acuratețea ajunge la în jur de 70-75%, în timp ce la ResNet preantrenat, acuratețea începe direct de la peste 80% și se plafonează la 87%.



Figură IV‑9 Inception preantrenat cu funcția de loss și acuratețea

În Figura IV‑4 este ilustrat Inception nepreantrenat, cu funcția de loss și acuratețea, iar în Figura IV‑9 este prezentată rețeaua Inception preantrenată. În prima figură, acuratețea ajunge la peste 80%, în timp ce la rețeaua preantrenată, acuratețea este în jur de 90%.

Din cele două comparații rezultă clar faptul că rețelele prenatrenate au rezultate mult superioare celor nepreantrenate. Rețelele au fost preantrenate cu baza de date ImageNet (Stanford Vision Lab, 2021).

## Rețele alese pentru telefon

Pentru telefon s-au ales 3 rețele neuronale, și anume: Inception cu preantrenare, ResNet fără preantrenare și ResNet preantrenat. Vom ilustra matricea de confuzie a fiecărei dintre aceste rețele pentru a putea observa cum se diferențiază clasele.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Margaretă | Păpădie | Necunoscut | Trandafir | Floarea-soarelui | Lalea |
| Margaretă | 125 | 7 | 16 | 4 | 10 | 2 |
| Păpădie | 5 | 149 | 10 | 5 | 9 | 3 |
| Necunoscut | 4 | 4 | 144 | 12 | 2 | 0 |
| Trandafir | 1 | 7 | 23 | 102 | 3 | 19 |
| Floarea-soarelui | 0 | 6 | 5 | 1 | 129 | 3 |
| Lalea | 4 | 7 | 18 | 49 | 8 | 128 |

Tabel IV‑1 ResNet fără preantrenare

După cum se observă, majoritatea claselor nu au un număr sesizabil de confuzii, singura excepție fiind lalelele ce conțin 49 de imagini care au fost confundate cu clasa trandafirilor. (Tabel IV‑1)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Margaretă | Păpădie | Necunoscut | Trandafir | Floarea-soarelui | Lalea |
| Margaretă | 144 | 4 | 8 | 0 | 5 | 3 |
| Păpădie | 0 | 171 | 3 | 2 | 2 | 3 |
| Necunoscut | 0 | 3 | 162 | 1 | 0 | 0 |
| Trandafir | 2 | 3 | 12 | 127 | 1 | 10 |
| Floarea-soarelui | 1 | 1 | 6 | 2 | 132 | 2 |
| Lalea | 1 | 5 | 19 | 10 | 1 | 178 |

Tabel IV‑2 ResNet cu preantrenare

La fel ca în cazul ResNetului nepreantrenat, în cazul preantrenării majoritatea plantelor nu se confundă, excepție făcând laleaua care se confundă cu clasa necunoscut de 19 de ori. Numărul de confuzii este, totuși, în toate clasele mult mai mic datorită preantrenării.(Tabel IV‑2)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Margaretă | Păpădie | Necunoscut | Trandafir | Floarea-soarelui | Lalea |
| Margaretă | 145 | 4 | 1 | 1 | 6 | 7 |
| Păpădie | 1 | 170 | 7 | 1 | 1 | 1 |
| Necunoscut | 0 | 2 | 164 | 0 | 0 | 0 |
| Trandafir | 0 | 0 | 7 | 138 | 1 | 9 |
| Floarea-soarelui | 1 | 2 | 3 | 0 | 136 | 2 |
| Lalea | 0 | 0 | 5 | 7 | 0 | 202 |

Tabel IV‑3 Inception cu preantrenare

Din Tabel IV‑3, se observă că matricea de confuzie a rețelei Inception preantrenată are caracteristici ușor mai bune decât ResNet preantrenat, consolidat și de acuratețea puțin mai mare.

## Comparații finale

Pentru a putea compara corect performanțele și duratele de antrenare ale rețelelor, am antrenat și testat următoarele rețele pe aceleași resurse hardware și software:

* Desktop:
  + Procesor AMD Ryzen 7 3700X ( 8 nuclee 4 GHz)
  + 32 GB RAM 3200 MHz
  + Placă grafică Nvidia GeFore GTX 1070 (8 GB)
  + TensorFlow 2.5.0
  + Keras 2.4.3
  + NumPy 1.19.5
* Mobil (Samsung S20+):
  + Procesor Samsung Exynos 990 ( 8 nuclee 2.73 GHz)
  + 8 GB RAM
  + Placă grafică Mali-G77
  + Android 11

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Nr. Parametri | Timp Procesare | Timp Antrenare | Dimensiune | Acuratețe |
| SVM | - | - | 9 m 7 s | - | 57.46% |
| ResNet34 | 21,279,174 | 180 ms | 31 m 50 s | 83 MB | 70.51% |
| ResNet34 + aug | 21,279,174 | 180 ms | 36 m 22 s | 83 MB | 75.88% |
| ResNet50 pre | 23,577,094 | 190 ms | 13 m 30 s | 92 MB | 89.26% |
| InceptionV2 | 23,167,362 | 290 ms | 63 m 26 s | 90 MB | 74.61% |
| InceptionV2 pre | 23,167,362 | 290 ms | 17 m 53 s | 90 MB | 93.26% |

Tabel IV‑4 Analiza tuturor rețelelor folosite

În Tabel IV‑4, “Nr. Parametri” reprezintă numărul de parametri pe care îi are rețeaua. “Timp Procesare” se referă la timpul de procesare al unei imagini pe dispozitivul mobil. “Timp Antrenare” reprezintă timpul de antrenare al unei rețele pe desktop. “Dimensiune” înseamnă dimensiunea pe care o are rețeaua în momentul exportării în format TensorFlow Lite. “Acuratețe” reprezintă procentul de recunoaștere exactă a clasei fotografiei.

Analizând cele trei rețele utilizate în cadrul proiectului, din Tabel IV‑4 se poate observa faptul că InceptionV2 preantrenat are acuratețea cea mai mare, dar și dimensiunea modelului și timpul de procesare sunt pe măsură. ResNet50 preantrenat este, în opinia mea, cel mai bun model, având în vedere faptul că timpul de procesare este considerabil mai mic decât rețelele Inception, însă acuratețea este doar cu puțin mai mică.

# Capitolul V. Concluzii

În cadrul acestui proiect de licență, am studiat un ansamblu de rețele neuronale ce au ca scop recunoașterea plantelor din imagini. Au fost făcute studii ale algoritmilor în diferite situații și influențați de mulți parametri. Au fost exportate și introduse trei dintre cele mai bune rețele într-o aplicație de mobil de recunoaștere a plantelor. Aplicația poate fi folosită în scop educativ, elevii având ca temă să găsească anumite tipuri de plante, dar și pentru curioșii care doresc să afle ce fel de plantă au în fața lor. Pozele recunoscute care sunt adăugate în galeria aplicației, sunt organizate pe categorii, în funcție de specia din care fac parte.

Posibile dezvoltări ulterioare ale aplicației:

* Aplicația de telefon poate fi extinsă, creându-se conturi de elevi și profesori. Profesorii au posibilitatea de a crea liste de teme cu plante pe care elevii trebuie să le găsească și să le fotografieze. În momentul asocierii unei plante fotografiate cu o categorie considerată potrivită de algoritm, s-ar putea crea opțiunea de a citi despre specia din care face parte planta recunoscută, acesta fiind un mod interactiv de a învăța mai multe despre plante.
* S-ar putea dezvolta mai mult baza de date de plante, utilizatori putând să contribuie la extinderea ei. Astfel, în momentul introducerii unei imagini cu o plantă, dacă algoritmul nu recunoaște specia din care face parte această plantă, se va crea un folder nou cu planta necunoscută. Apoi, dacă un utilizator recunoaște acea plantă, va putea adăuga numele. Alți utilizatori vor avea posibilitatea de a acredita printr-un like sau o bifare a numelui dat, iar denumirii cu scorul cel mai mare îi va fi păstrat numele. După acest proces, rețeaua va fi reantrenată, astfel extinzând-se baza de date. În final, se poate exporta pentru doritori noua bază de date.

În opinia mea, recunoașterea imaginilor cu ajutorul inteligenței artificiale este una dintre cele mai utile tehnologii ce pot fi folosite în viața de zi cu zi, iar introducerea AI-ului în cât mai multe aplicații va duce la o dezvoltare uluitoare a domeniului. Prin studiile făcute pe baza seturilor de date de plante și crearea unei aplicații de recunoaștere a florei, consider că am adăugat încă un pas în marea evoluție a domeniului. Întotdeauna este, totuși, loc de mai mult, iar rezultatele obținute pot fi îmbunătățite cu răbdare și perseverență.

# Bibliografie

Appixi, 2021. *LeafSnap.* [Interactiv]   
Available at: https://plantidentifier.info/#block51  
[Accesat 15 August 2021].

Christian Szegedy, V. V. S. I. J. S., 2015. Rethinking the Inception Architecture for Computer Vision. *arXiv:1512.00567v3 [cs.CVî.*

COCO Consortium, 2015. *MS COCO.* [Interactiv]   
Available at: https://cocodataset.org/#home  
[Accesat 10 August 2021].

Doshi, S., 2019. *Towards Data Science.* [Interactiv]   
Available at: https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6  
[Accesat 5 Iulie 2021].

Fisher, R. A., 1936. The use of multiple measurements in taxonomic problems. *Annals of Eugenics,* Volume 7(2), pp. 179-188.

Geron, A., 2019. *Hands- on Machine Learning with Scikit-Learn, Keras & TensorFlow.* 2nd ed. s.l.:O'Reilly.

Google, 2021. *Documentație Android.* [Interactiv]   
Available at: https://developer.android.com/reference  
[Accesat 2 Mai 2021].

JetBrains s.r.o., 2021. *Documentatie PyCharm.* [Interactiv]   
Available at: https://www.jetbrains.com/help/pycharm/quick-start-guide.html  
[Accesat 12 Mai 2021].

Kaiming He, X. Z. S. R. J. S., 2015. ResNet. *arXiv:1512.03385 [cs.CV].*

Mamaev, A., 2021. *Baza de date cu 5 categorii de plante.* [Interactiv]   
Available at: https://www.kaggle.com/alxmamaev/flowers-recognition  
[Accesat 3 Ianuarie 2021].

PlantNet, 2021. *PlantNet.* [Interactiv]   
Available at: https://plantnet.org/en/  
[Accesat 5 August 2021].

Scikit-learn developers, 2020. *Documentatie Sklearn.* [Interactiv]   
Available at: https://scikit-learn.org/stable/user\_guide.html  
[Accesat 21 Mai 2021].

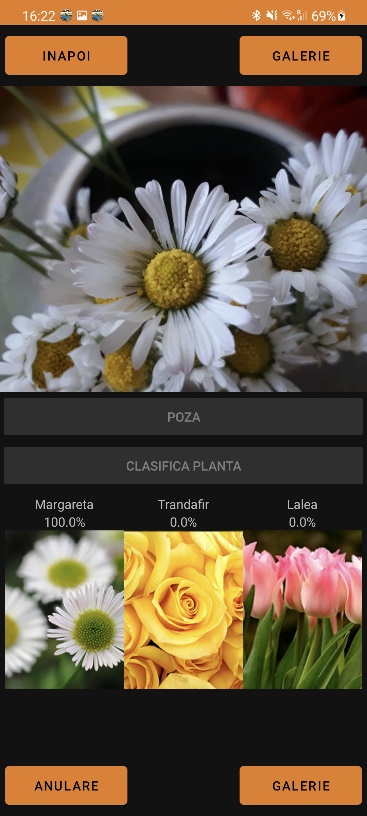
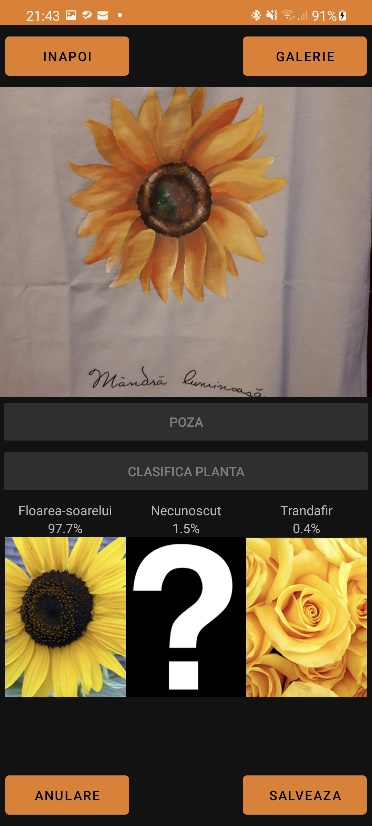
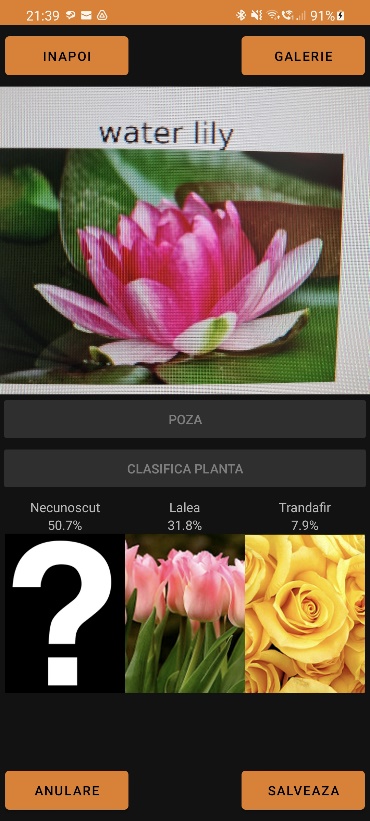
Siddharth , S., Simone , S. & Anidhya, A., 2020. Activation functions in neural networks. *International Journal of Engineering Applied Sciences and Technology,* 4(2), pp. 310-316.

Stanford Vision Lab, 2021. *ImageNet.* [Interactiv]   
Available at: https://image-net.org/index.php  
[Accesat 10 August 2021].

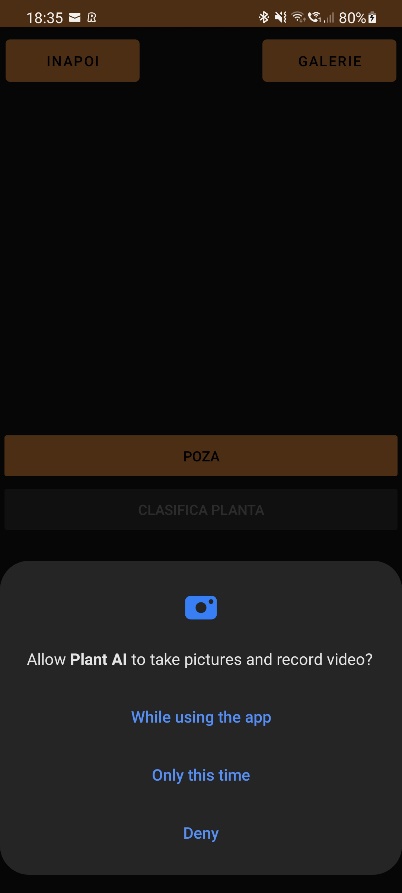
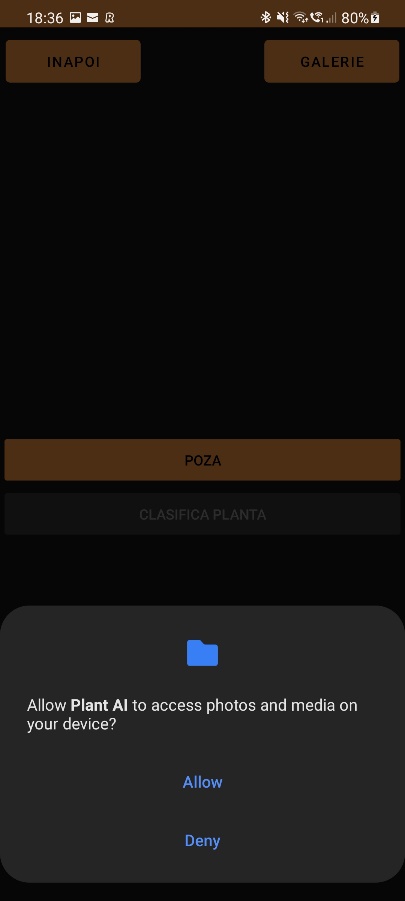
TensorFlow, 2021. *Documentatie TensorFlow.* [Interactiv]   
Available at: https://www.tensorflow.org/api\_docs/python/tf  
[Accesat 16 Iulie 2021].

Yu, S., Yuan , L., Guan , W. & Haiyan, Z., 2017. Deep Learning for Plant Identification in Natural Environment. *Computational Intelligence and Neuroscience,* Volumul 2017.

# Anexe

Anexă 1 – Clasificare de imagini

Anexă – Cerere drepturi pe telefon