UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

Clasificarea fructelor

propusă de

Laurențiu-Valentin Burluc

Sesiunea: *Februarie, 2021*

Coordonator științific

Conf. Dr. Anca Ignat

UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAȘI

FACULTATEA DE INFORMATICĂ

Clasificarea fructelor

Laurențiu-Valentin Burluc

Sesiunea: *Februarie, 2021*

Coordonator științific

Conf. Dr. Anca Ignat

Cuprins

Introducere

Computer Vision este un domeniu din inteligență artificială, care se concentrează pe reproducerea unor părți ale complexității sistemului de viziune umană și permite calculatoarelor să înțeleagă lumea vizuală. Folosind videoclipuri, imagini și modele de deep learning, mașinăriile pot identifica cu acuratețe și clasifica obiecte. Datorită inovațiilor în deep learning, progreselor în inteligență artificială și rețelelor neuronale, acest domeniu a evoluat mult în ultimii ani și a reușit să depășească oamenii în anumite sarcini legate de detectarea și etichetarea obiectelor. Această evoluție a avut un impact masiv asupra companiilor din toate industriile, de la comerț până la medicină. Primele experimente în computer vision au început în anul 1950, iar, în anul 1970, prima utilizare comercială a computer vision a fost interpretarea textului tastat sau scrisului de mână, folosind recunoașterea optică a caracterelor. În ultimii ani au fost dezvoltate aplicații, precum urmărirea mingii și a jucătorilor în sport, clasificarea celulelor, detectarea cancerului și a tumorilor în medicină, monitorizarea animalelor, automatizarea fermelor, recunoașterea plantelor și a fructelor în agricultură, conducere autonomă, evitarea accidentelor, analiză de trafic și siguranță în transport, urmărirea clienților, detectare furtului, numărarea oamenilor și analize de productivitate. Aplicații de computer vision au fost dezvoltate și cu scopul de a gestiona mai ușor pandemia SARS-CoV-2, precum distanțarea socială, detectarea măștii și clasificarea de tomografii computerizate.

Motivația alegerii temei

Clasificarea imaginilor este un proces de etichetare a imaginilor în funcție de categoriile predefinite. Procesul de clasificare a imaginilor se bazează pe învățare supravegheată. Un model de clasificare a imaginilor este antrenat pe baza unui set de imagini specifice categoriilor predefinite. Pe baza acestui set de imagini, algoritmul învață din ce clasă fac parte imaginile din setul de testare și să prezică clasa corectă a imaginilor care vor fi date ca intrare. Clasificarea se referă la imaginile în care apare un singur obiect ce trebuie analizat. Aceasta este una dintre problemele esențiale în computer vision, care, în ciuda simplității sale, are o mare varietate de aplicații practice, de la clasificarea celulelor în medicină până la identificarea fructelor si legumelor în agricultură, iar rețelele neuronale convoluționale excelează la acest tip de sarcină.

Astfel, simplitatea acestei probleme, varietatea de aplicații practice care pot fi făcute, precum și eficiența rețelelor neuronale convoluționale m-au motivat să abordez acest tip de problemă. Pentru rezolvarea problemei am decis să abordez clasificarea fructelor si legumelor.

Plata fructelor sau a legumelor în supermarketuri necesită în mod normal ca acestea să fie identificate manual de către clienți. Majoritatea produselor au cod de bare pentru ca identificarea acestora să se facă automat prin scanare. Acest fapt scurtează timpul petrecut de către clienți atunci când au cumpărături de făcut. Identificarea fructelor și a legumelor folosind rețelele neuronale convoluționale pot scuti efortul și timpul clientului de a obține un cod de bare pentru tipul specific de fruct sau legumă.

Obiectivele generale ale lucrării

Clasificarea fructelor este o problema complexă din cauza tuturor variațiilor care pot fi întâlnite. În general, pot fi identificate două probleme de clasificare. Prima problemă este de clasificare a fructelor de diferite tipuri, iar ca exemplu ar fi diferențierea între portocale, banane și mere. A doua problemă este clasificarea aceluiași soi de fruct, de exemplu diferența dintre merele golden și red. Lucrarea de față are ca scop rezolvarea celor doua probleme cu ajutorul mai multor configurații de rețele neuronale convoluționale, precum și cu transfer de învățare de la modele deja antrenate și compararea rezultatelor. Obiectivele generale ale lucrării este de a descoperi configurații cât mai potrivite pentru acest tip de problemă și de a veni cu niște concluzii.

Descrierea sumară a soluției

Identificarea fructelor si legumelor se va efectua prin rețele neuronale convoluționale și prin transfer de învățare de la modelele VGG16, MobileNetV2 și Xception, care au fost antrenate pe mai mult de un milion de imagini din baza de date ImageNet și pot să clasifice 1000 de categorii. Setul de date pentru antrenarea modelelor este format din mai multe imagini cu diferite categorii de fructe și legume. Fiecare imagine are dimensiunea de 100x100 pixeli și conține un singur obiect bine încadrat, iar restul pixelilor sunt albi. Diferite configurații de set de imagini și modele vor ajuta la stabilirea unor rezultate reprezentate prin grafice și precizia prezicerilor, care vor ajuta la alegerea unor soluții favorabile în rezolvarea problemelor descrise în Obiectivele generale ale lucrării.

Structura lucrării

Lucrarea va fi structurată în trei capitole. În primul capitol se va descrie problema care trebuie să fie rezolvată. În capitolul doi se vor descrie câteva abordări anterioare ale clasificării fructelor și legumelor, iar capitolul trei va fi alcătuit din mai multe subcapitole, care vor prezenta detaliat soluția propusă. Aceste subcapitole sunt:

3.1 Tehnologii utilizate

3.2 Set de date

3.3 Metode

3.4 Rezultate

După cele trei capitole vor urma concluziile lucrării, unde se vor regăsi cele mai importante aspecte din lucrare, precum si opinia personală privind rezultatele obținute în lucrare.

Contribuții personale

În cadrul acestei lucrări, următoarele contribuții personale m-au ajutat să îndeplinesc obiectivele propuse:

- Realizarea setului de testare neîncadrat pe un fundal alb și setului de testare cu fructe bine încadrat.
- Implementarea unei arhitecturi de rețea neuronală convoluțională și testarea acesteia.
- Testarea aplicațiilor Keras pe setul de testarea nebalansat, setul pe care s-a făcut undersampling și setul normal.
- Testarea modelelor cu mai multe tipuri de straturi dense la final de arhitectură.
- Analiza și interpretarea rezultatelor
- Compararea modelelor pe imagini clasificate pentru prima dată.
- Tehnica de undersampling folosită pentru a balansa setul de imagini.
- Explorarea unor noi domenii: cel al computer vision și cel al rețelelor neuronale convoluționale.

1 Descrierea problemei

Lanțurile de supermarketuri și hipermarketuri se bazează pe casele de marcat sau pe casele unde clienții își scanează singuri cumpărăturile pentru ca tot procesul să dureze cât mai puțin. În prezent, recunoașterea produselor cu ajutorul codului de bare este cea mai utilizată tehnologie în industrie și cercetare. Scanarea codului de bare de pe fiecare produs ajută mult la gestionarea eficientă a mărfurilor. În mod normal, aproape fiecare articol de pe piață are un cod de bare corespunzător. Excepție fac fructele și legumele care sunt identificate manual de către clienți.

Identificare manuală se face cu ajutorul cumpărătorilor care trebuie să pună fructele pe cântar și să găsească în lista afișată pe un display cu ecran tactil tipul fructului pentru a primi codul de bare corespunzător, iar acest proces poate să dureze atunci când sunt multe produse de căutat. O altă metodă folosită pentru obținerea codului de bare este aceea de a memora un număr unic afișat în dreptul tipului de produs. Acest număr este introdus ulterior în sistem pentru a obține codul de bare. Această metodă, față de prima, necesită mai mult efort din partea clientului, mai ales dacă a cumpărat mai multe tipuri de fructe și trebuie să le identifice manual, deoarece sunt multe numere de reținut și șansa de a le uita este mai mare. Primul caz vine cu lista cu produse, care scutește clientul să se deplaseze și să afle numerele.

Dar cum am precizat și în Obiectivele generale ale lucrării, clasificarea fructelor și legumelor este o problemă complexă din cauza tuturor variațiilor care pot fi întâlnite, iar această problemă de identificare va fi împărțită în trei cazuri. Primul caz este acela când fructele diferă ca formă, textură și culoare cu următorul exemplu: banane, portocale și mere. Al doilea caz este reprezentat de soiuri diferite de fructe cu exemplul: tipuri diferite de mere sau portocala sau mandarine. Ultimul caz le cuprinde pe primele două, crescând dificultatea prezicerii categoriei.

Un aspect important este că identificarea fructelor se vor face fără ca acestea să fie în pungi și pentru a găsi categoria corespunzătoare va fi nevoie doar de un singur fruct din clasa respectivă.

2 Abordări anterioare

În prima lucrare [1] este prezentată o aplicație care folosește o metodă bazată pe o rețea neuronală convoluțională compactă cu scopul de a accelera procesul de identificare a produselor. Setul de date este format din imagini cu mere, banane și portocale. Fructele au fost așezate pe o foaie de oțel inoxidabil pentru a reproduce mediul din magazin, iar fotografiile au fost făcute de sus. Dimensiunea imaginilor este de 128x128 pixeli, deoarece MobileNetV2 va fi antrenat cu învățare prin transfer folosind *weights* dintr-un model deja antrenat pe baza setului de date ImageNet, iar dimensiunea minimă cerută pentru rezultate cât mai bune pentru acest model este de 128x128 pixeli. Modelul este antrenat pe două versiuni de set de date: imagini cu fructe și imagini cu fructe in pungi de plastic. Pe lângă MobileNetV2, lucrarea mai prezintă și câteva variații ale acestuia, care sunt modele cu multiple date de intrare: MobileNetV2 cu o singură culoare RGB și MobileNetV2 cu histogramă RGB, precum și un model hibrid MobileNetV2 cu K-Means. Cele mai bune rezultate au fost obținute de MobileNetV2 cu o singură culoare RGB, iar cele mai slabe rezultate au fost obținute de MobileNetV2.

În a doua lucrarea [2] este prezentat un robot cu un braț programat să localizeze, detecteze și să culeagă fructele din pom fără a le provoca daune. Acest robot este echipat cu o cameră color și detector MultiBox Single Shot, pentru a detecta în doua dimensiuni poziția fructului. Detectorul este una dintre metodele de detectare a obiectului ce folosește rețele neuronale convoluționale și poate determina culoarea și forma. O poziție tridimensională trebuie obținută pentru ca brațul robotului să poată culeagă fructele. O cameră stereo este folosită pentru a măsura poziția tridimensională a fructului detectat bidimensional. Brațul robotului recoltează fructul prin apucarea și rotirea acestuia fără a deteriora nimic în jurul acestuia. Algoritmul cuprinde trei pași: detectarea bidimensională a fructului, detectarea tridimensională și calcularea cinematicii inverse. Detectorul bidimensional are la bază o rețea neuronală convoluțională, care detectează obiectul din imagine folosind o singură rețea neuronală profundă. Celelalte metode de detecție sunt Faster R-CNN și You Only Look Once. Primul pas folosește VGG net pentru a extrage hărți cu caracteristici. Modelul prezice scorurile categoriei, iar localizarea se realizează cu filtre mici convoluționale aplicate pe hărțile de caracteristici.

3 Descrierea soluției

În acest capitol vor fi prezentate tehnologiile utilizate, setul de date, metodele folosite și rezultatele obținute în urma antrenării modelelor prin învățare cu transfer sau fără.

3.1 Tehnologii utilizate

Ca mediu de lucru am decis să folosesc Google Colaboratory Pro, deoarece este un mediu de notebook-uri Jupyter online, bazat pe cloud, care permite utilizatorului să antreneze modelele de învățare automată și de deep learning pe procesoare și plăci video performante, astfel încât timpii de antrenare devin mai mici. Orice utilizator care folosește Google Colaboratory Pro dispune de următoarele plăci video: Tesla T4 sau Tesla P100 și de 27 GB RAM. Limbajul de programare utilizat în Google Colaboratory este Python, care este un limbaj de nivel înalt, ce face codul ușor de citit și de înțeles. Majoritatea bibliotecilor populare sunt instalate implicit, iar cele folosite pentru a realiza această aplicație sunt keras, numpy, sklearn, os, matplotlib.pyplot, google.colab și mlxtend.plotting. Pentru a putea interacționa cu Google Drive a trebuit să folosesc librăria google.colab cu scopul de a exporta modelele și graficele și de a importa setul de date. Librăriile sklearn și mlxtend.plotting le-am utilizat cu scopul de a obține matrice de confuzie. Librăria os a fost folosită pentru a interacționa cu fișierele din sistem, iar matplotlib.pyplot a fost folosit pentru a afișa toate imaginile. Un rol important în această aplicație îl are librăria Keras.

Keras este un API de rețele neuronale de nivel înalt, care rulează deasupra TensorFlow și este foarte faimos în domeniul de deep learning. Am ales Keras pentru că este simplu și foarte puternic pentru începătorii care dau startul învățării profunde. Este ușor să proiectezi modele de rețele neuronale. Oferă numeroase modele pre-antrenate precum: VGG16, VGG19, Xception, NASNet, MobileNet, MobileNetV2, InceptionV3 și InceptionResNetV2, iar unele dintre ele vor fi folosite pentru transfer de învățare în această aplicație. Rețelele neuronale sunt construite în câteva linii de cod, iar Keras are o comunitate mare de susținere. Funcțiile folosite din librăria Keras au fost de a crea un generator de imagini pentru a augmenta setul de date și a-l face mai mare, de a citi imaginile și a le transforma în matrici, de a crea și antrena modele de rețele neuronale convoluționale, de a salva și încărca modele și de a prezice obiectele din imagini.

3.2 Setul de date

Cele mai mari dificultăți în rezolvarea problemelor de clasificare le constituie seturile de date nebalansate. Un set de date nebalansat poate fi reprezentat de unele clase care conțin mai multe sau mai puține imagini decât restul claselor. Antrenarea pe un set de imagini nebalansat poate duce la predicții nefavorabile. Prin urmare, pentru a obține rezultate favorabile am decis să caut un set de imagini bine balansat.

3.2.1 Prezentare

Setul de date fruits-360 [3] a fost obținut de pe platforma kaggle și conține imagini cu fructe și legume. Numărul de clase pe care setul de date îl conține este de 131 de clase, unde diferite soiuri de fructe aparțin de clase distincte. Dimensiunea imaginilor este de 100x100 pixeli. Fructele și legumele au fost plantate în axul unui motor cu viteză de 3 rotații pe minut și a fost înregistrat un film scurt de 20 de secunde. Camera care a filmat fructele este un Logitech C902. Fundalul fructelor a fost reprezentat de o foaie albă, dar din cauza variației luminii, fundalul nu a fost uniform și un algoritm a fost aplicat, astfel încât fundalul vechi să fie înlocuit de pixeli albi. Numărul total de imagini este de 90483, dintre care 67692 de imagini fac parte din setul de antrenare și 22688 de imagini fac parte din setul de testare. Pe lângă acest set de date, am mai creat un set de imagini de testare pentru a vedea ce rezultate obținem. Acest set conține imagini făcute pe un fundal alb cu blițul pornit, precum și imagini cu diverse fructe de pe internet. În fruits-360 [3] din cele 131 de categorii doar 67 sunt fructe de tipuri diferite, iar restul sunt soiuri diferite ale aceluiași fruct.

Pe lângă acest set de date, am mai creat un set de testare format din patru categorii: mere, portocale, pere și banane. Pozele au fost făcute cu telefonul, unde fundalul a fost reprezentat de o foaie albă. Cu ajutorul acestui set de imagini am mai creat încă un set cu aceleași imagini, doar că fructele sunt bine încadrate.

Din setul de imagini fruits-360 [3] am extras anumite categorii pentru a forma încă două seturi de date. Primul set de date va fi format din categoriile: mere, pere, portocale și banane. Al doilea set va fi format din toate cele 13 tipuri de soiuri de măr găsite în fruits-360 [3]. Scopul acestor seturi este cel de a vedea izolat dacă modelele antrenate identifică mai precis categoriile.

Figura 1 – Imagini din setul de date fruits-360

Figura 2 - Imagini bine încadrate din setul de testare

3.2.2 Preprocesare imaginilor

Toate imaginile din setul de date obținut de pe platforma kaggle au dimensiunea de 100x100 pixeli. Augmentarea setului de imagini a fost făcut cu ajutorul clase ImageDataGenerator, oferit de biblioteca Keras, iar această tehnică este folosită pentru a extinde artificial mărimea setului de antrenament. Inițial, setul de date era împărțit în 75% imagini pentru antrenare și 25% imagini pentru testare. Cu ajutorul tehnicii de augmentare am normalizat setul de date de la intervalul 0-255 la intervalul 0-1 prin împărțirea tuturor pixelilor la valoare 255. Am aplicat întorsături verticale și orizontale și am împărțit setul de antrenament în 90% set de antrenament, iar restul de 10% set de validare. Pe setul de testare s-a aplicat doar normalizarea 0-1. Cu funcția flow_from_directory am pregătit seturile de imagini pentru modele ce urmează să fie antrenate. Am amestecat setul de antrenament pentru o generalizare mai bună a categoriilor și imaginile au fost preluate de către model în batch-uri de câte 32 de imagini. Nu am aplicat zoom-uri aleatorii pe imagini deoarece forma fructului contează, iar procesul de zoom poate deforma fructul.

3.3 Metode

Acest subcapitol va conține descrierea arhitecturilor rețelelor neuronale convoluționale utilizate. În prima parte a subcapitolului se va prezenta structura rețelei neuronale convoluționale fără transfer de învățare, iar în a doua parte se vor prezenta structurile modelelor pre antrenate care vor fi utilizate cu sau fără transfer de învățare.

3.3.1 Structura arhitecturii rețelei neuronale convoluționale

Arhitectura de bază a rețelei neuronale convoluționale este compusă din patru straturi convoluționale, patru straturi de MaxPooling și straturi complet conectate cu funcții de activare ReLU. Ultimul strat complet conectat are funcție de activare de tip softmax.

În Figura 3 - Arhitectură CNN putem observa arhitectura rețelei, dar în mod detaliat rețeaua neuronală convoluțională este formată din:

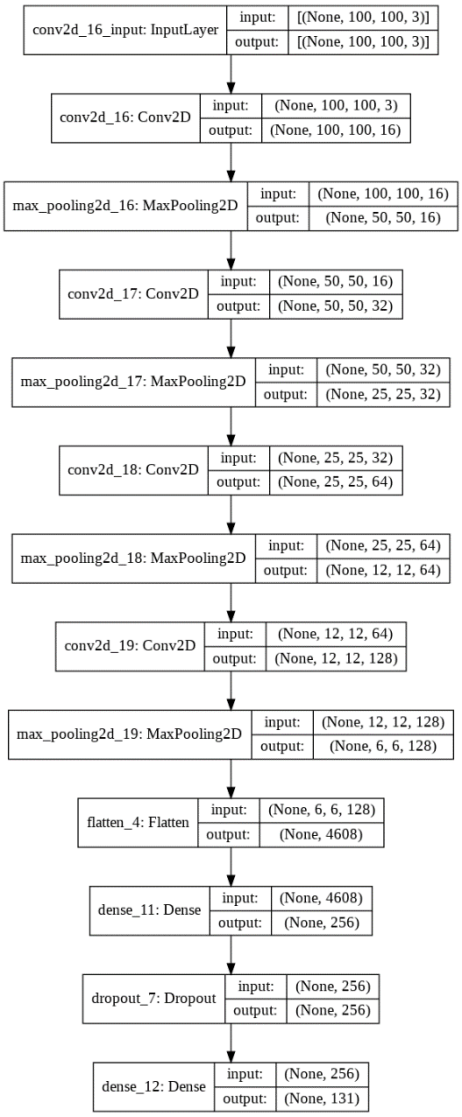


Figura 3 - Arhitectură CNN

- Un strat convoluțional (Conv2D) cu dimensiunea nucleului de 3x3 pixeli și 16 de filtre
- Un strat de maxpool (MaxPool2D) cu pool size de 2x2 și stride de 2x2.
- Un strat convoluțional (Conv2D) cu dimensiunea nucleului de 3x3 pixeli și 32 de filtre
- Un strat de maxpool (MaxPool2D) cu pool size de 2x2 și stride de 2x2.
- Un strat convoluțional (Conv2D) cu dimensiunea nucleului de 3x3 pixeli și 64 de filtre
- Un strat de maxpool (MaxPool2D) cu pool size de 2x2 și stride de 2x2.
- Un strat convoluțional (Conv2D) cu dimensiunea nucleului de 3x3 pixeli și 128 de filtre
- Un strat de maxpool (MaxPool2D) cu pool size de 2x2 și stride de 2x2.

Fiecare strat convoluțional are o activare ReLU (Rectified Liner Unit), astfel încât toate valorile negative să nu fie trecute la următorul strat. Urmează funcția de flatten, care convertește data într-un vector unidimensional, permițând ca ieșirea finală să fie procesată de stratul următor complet conectat.

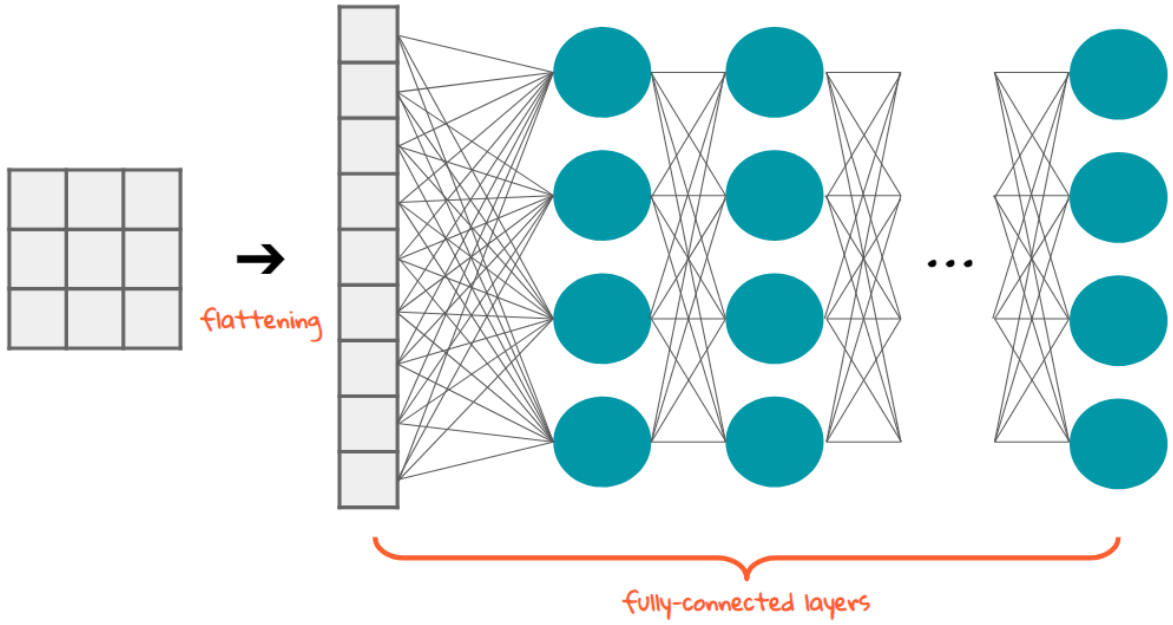


Figura 4 - Aplicarea funcției flatten

Urmează un strat dens de 512 de unități cu funcția de activare ReLU. Peste acest strat se aplică funcția Dropout [4]. Aceasta este cea mai simplă metodă de regularizare a rețelelor neuronale, care dezactivează un procent de neuroni de pe un strat. Acest lucru îmbunătățește generalizarea deoarece forțează stratul să învețe cu diferiți neuroni aceleași caracteristici. Rata de Dropout aleasă de mine este de 30%.

Ultimul strat din structura rețelei neuronale convoluționale este stratul de ieșire, care cuprinde o funcție de activare softmax și conține 131 de neuroni, câte unul pentru fiecare categorie de fruct sau legumă.

Pentru optimizare, am ales o metoda populară ce calculează rata de învățare adaptivă pentru fiecare parametru și anume Adam (Adaptive Estimation Moment).

3.3.2 Transfer de învățare

Învățarea prin transfer constă în preluarea caracteristicilor învățate cu privire la o problemă și utilizarea acestora într-o nouă problemă similară. De exemplu, caracteristicile unui model care a învățat să identifice portocalele pot fi utile pentru a începe un model menit să identifice

mandarine. Învățarea prin transfer se face de obicei pentru sarcini în care setul de date are prea puține date de antrenament.

Lisa Torrey și Jude Shavlik [5] descriu în capitolul lor de învățare prin transfer trei beneficii posibile pe care trebuie să le ai în vedere atunci când utilizați învățarea prin transfer.

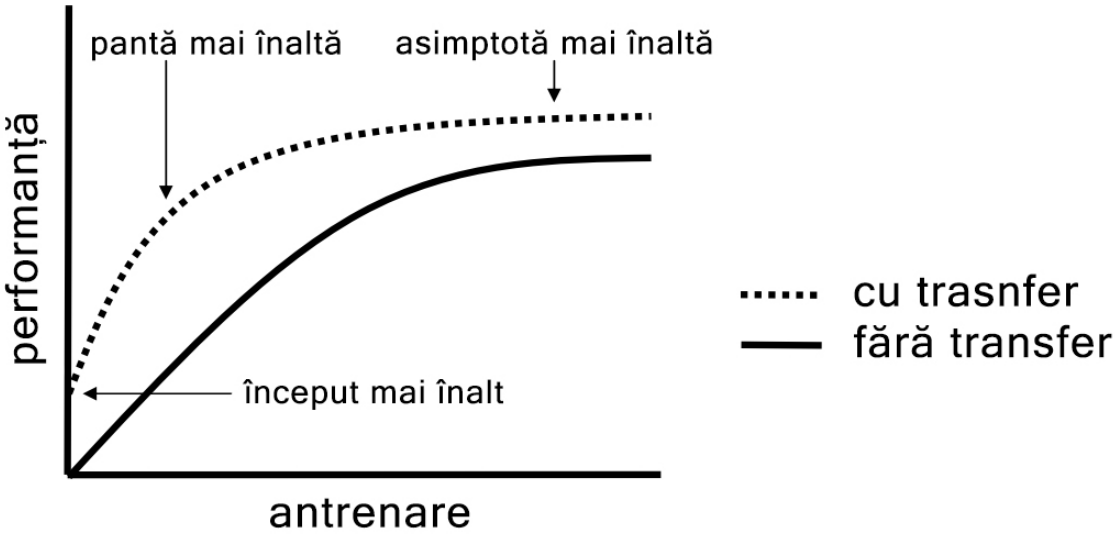


Figura 5 - Ilustrarea beneficiilor transferului prin învățare

Primul beneficiu este un început mai înalt, care este mult mai înalt la un model cu transfer de învățare. Al doilea beneficiu este reprezentat de o pantă mai înaltă, care reprezintă o rată de îmbunătățire a performanței mai abruptă în timpul antrenamentului. Ultimul beneficiu este o asimptotă mai înaltă, care este reprezentată de o abilitate convergentă a modelului antrenat mai bună.

Cea mai comună încarnare a învățării prin transfer [6] în contextul învățării profunde este reprezentată de următorii pași:

1. Obținerea de straturi dintr-un model antrenat anterior.
2. Înghețarea straturilor, astfel încât să evităm distrugerea oricărei informații din straturi în timpul următoarelor antrenamente.
3. Adăugarea de straturi noi antrenabile deasupra celor înghețate, care vor învăța să transforme vechile caracteristici în predicții pentru un nou set de date.
4. Instruirea noilor straturi pe setul de date.

Un ultim pas opțional îl reprezintă fine-tuning, care constă în dezghețarea întregului model obținut cu pașii de mai sus sau o parte din el și reantrenarea acestuia pe noile date cu o rată de învățare foarte scăzută. Acest lucru poate obține îmbunătățiri semnificative prin adaptarea treptată a caracteristicilor pre antrenate la noile date.

Am folosit aplicațiile Keras [7], care sunt modele de deep learning care sunt puse la dispoziție alături de weights pre antrenate. Modele alese pentru această lucrare sunt VGG16, Xception și MobileNetV2, deoarece, din cum se poate observa din tabelul de mai jos, Xception are cea mai mare acuratețe și o dimensiune medie, VGG16 are o dimensiune mare și o arhitectură mai simplă și MobileNetV2 are cea mai mică dimensiune și o acuratețe egală cu VGG16, iar dimensiunea îl avantajează într-o aplicație de identificat fructe și legume într-un supermarket.

Model	Dimensiune	Top-1 acuratețe	Top-5 acuratețe	Parametri	Adâncime
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
MobileNetV2	14 MB	0.713	0.901	3,538,984	88

Tabel 1 - Aplicații Keras

La aceste aplicații vom prelua arhitectura de bază și vom adăuga peste unul sau mai multe straturi dense cu funcția de activare ReLU, funcții Dropout peste aceste straturi pentru a generaliza mai bine setul de date și la sfârșit un strat dens cu activare softmax și cu numărul de unități egal cu numărul de categorii de fructe.

Arhitectură VGG16

VGG16 [8] este o arhitectură de rețea neuronală convoluțională, care a fost folosită pentru a câștiga competiția ILSVR (ImageNet) în anul 2014.

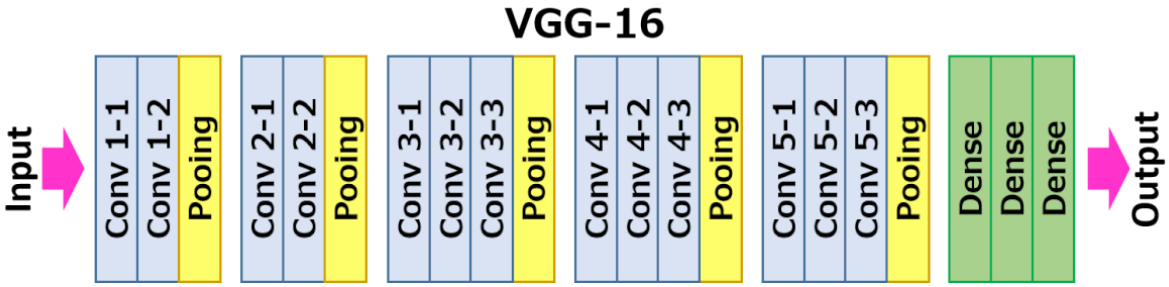


Figura 6 - Arhitectură VVG-16

Imaginile de intrare au o dimensiune fixată de 224x224 pixeli RGB. Arhitectura VGG16 este cea mai simplă arhitectură și este formată din:

- Două straturi convoluționale (Conv2D) cu 64 de filtre, kernel size de 3x3 și același padding;
- Un strat de maxpool (MaxPool2D) cu pool size de 2x2 și stride de 2x2;

- Două straturi convoluționale (Conv2D) cu 128 de filtre, kernel size de 3x3 și același padding;
- Un strat de maxpool (MaxPool2D) cu pool size de 2x2 și stride de 2x2;
- Trei straturi convoluționale (Conv2D) cu 256 de filtre, kernel size de 3x3 și același padding;
- Un strat de maxpool (MaxPool2D) cu pool size de 2x2 și stride de 2x2;
- Trei straturi convoluționale (Conv2D) cu 512 de filtre, kernel size de 3x3 și același padding;
- Un strat de maxpool (MaxPool2D) cu pool size de 2x2 și stride de 2x2;
- Trei straturi convoluționale (Conv2D) cu 512 de filtre, kernel size de 3x3 și același padding;
- Un strat de maxpool (MaxPool2D) cu pool size de 2x2 și stride de 2x2.

Fiecare strat convoluțional are o activare ReLU (Rectified Liner Unit), astfel încât toate valorile negative să nu fie trecute la următorul strat. După urmează o aplatizare a vectorului rezultat și ultimele trei straturi sunt:

- Un strat dens de 4096 unități cu activare ReLU;
- Un strat dens de 4096 unități cu activare ReLU;
- Un strat dens de 1000 de unități cu activare softmax.

Arhitectură Xception

Arhitectura Xception [9] are 36 de straturi convoluționale care formează baza de extracție a caracteristicilor. Cele 36 de straturi convoluționale sunt structurate în 14 module, toate având conexiuni reziduale liniare în jurul lor, cu excepția primului si ultimului modul. Arhitectura este reprezentată de imaginea de mai jos, unde datele trec mai întâi prin fluxul de intrare, apoi ajunge în fluxul de mijloc care se repetă de opt ori și în cele din urmă trece prin fluxul de ieșire. Toate straturile convoluționale și convoluționale separabile sunt urmate de o serie de batch normalization. Toate straturile convoluționale separabile utilizează un multiplicator de adâncime unu, fără expansiune de adâncime.

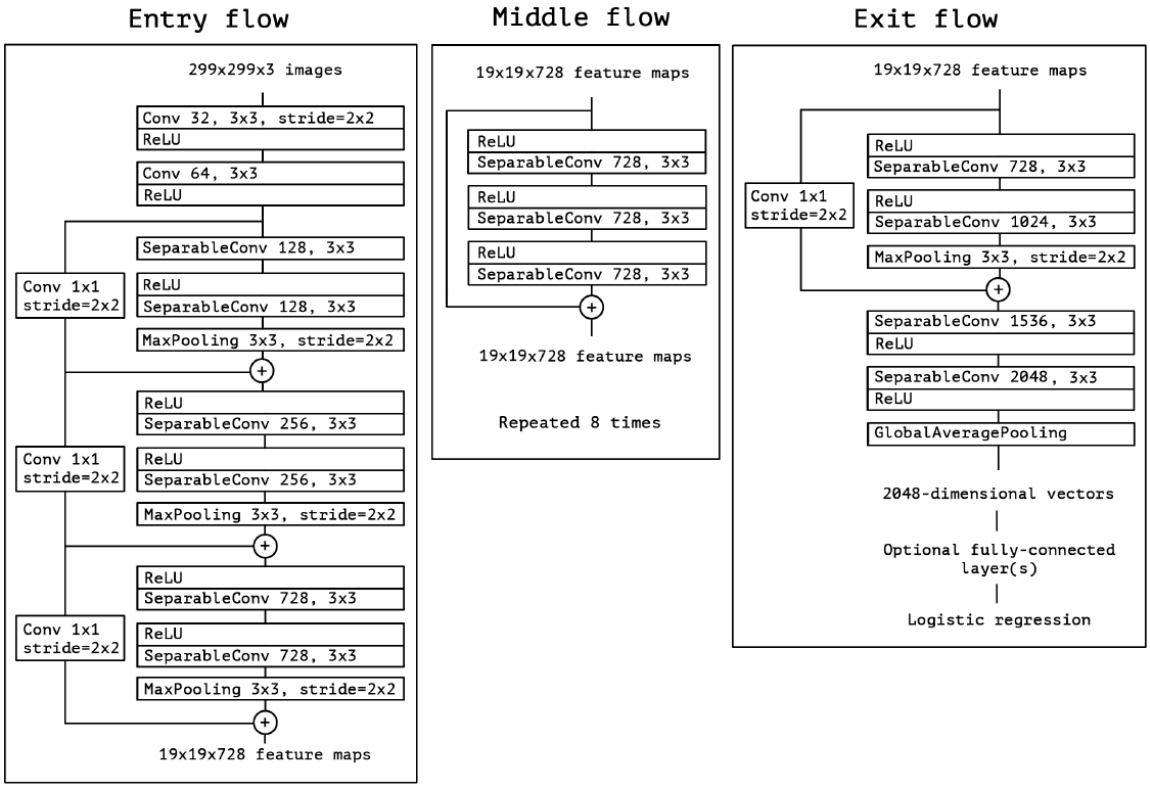


Figura 7 - Arhitectură Xception

Arhitectură MobileNetV2

MobileNetV2 [10] este o arhitectură de rețea neuronală convoluțională cu cea mai mică dimensiune ocupată. Straturile predominante în MobileNetV2 sunt reprezentate de straturi reziduale inversate, iar cu ajutorul acestora se obțin rezultate foarte bune în aplicațiile mobile. În imagine de alături este reprezentată arhitectura acestui model, unde t reprezintă rata de expansiune a canalelor, c reprezintă numărul canalelor de intrare și n reprezintă numărul de repetiții în arhitectură. Ultimă coloană s ne spune dacă prima repetare a unui bloc a folosit stride de doi pentru downsampling.

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Figura 8 - Arhitectură MobileNetV2

3.4 Rezultate

În acest subcapitol vom aborda clasificarea tipurilor de fructe, clasificarea fructelor de același soi și clasificarea ambelor cazuri în același set de date.

3.4.1 Clasificarea soiurilor de fructe

În această problemă vom extrage toate categoriile de mere din fruits-360 [3] și vom forma un nou set de date. Numărul soiurilor de mere este de 13. Setul de date va fi împărțit în antrenare, validare și testare.

Figura 9 - Soiurile de mere

Arhitecturile folosite sunt aceleași ca cele prezentate în 3.3, doar că vom antrena modelele de bază cu diferite straturi dense la final. Cele trei cazuri sunt:

- Un strat dens cu funcția softmax cu 13 unități;
- Un strat dens cu funcția de activare ReLU cu 256 de unități, Dropout de 20% și un strat dens cu funcția softmax cu 13 unități;
- Un strat dens cu funcția de activare ReLU cu 512 de unități, dropout de 20%, un strat dens cu funcția de activare ReLU cu 256 de unități, dropout de 20% și un strat dens cu funcția softmax cu 13 unități.

Arhitectura rețelei neuronale convoluționale

Am antrenat toate cele trei tipuri de modele la 50 de epoci, cu optimizatorul Adam și callback-ul ReduceLROnPlateau, cu rata de așteptare de 3 epoci și factorul de reducere a învățării de 0.5 până la un minim de 0.00001.

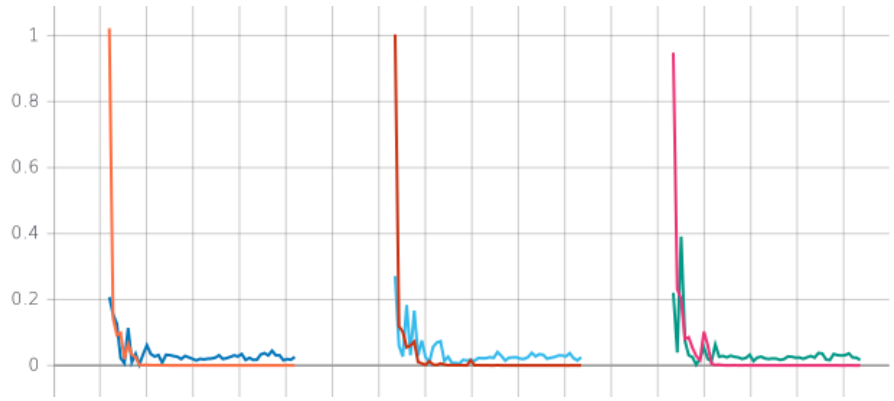


Figura 10 - Valorile loss la antrenare CNN (Simplu, Un strat dens , două straturi dense)

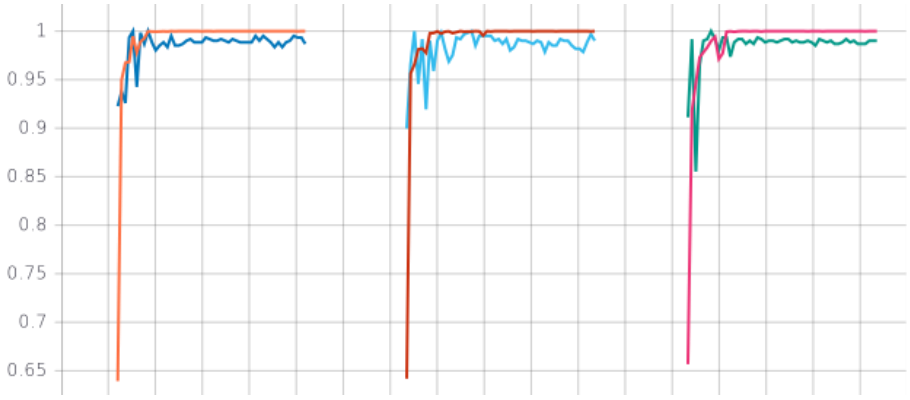


Figura 11 - Valorile acurateței la antrenare CNN (simplu, un strat dens , două straturi dense)

În figurile de mai sus se observă că loss și accuracy ating valori foarte favorabile, iar val_loss și val_accuracy tind spre valoarea 1 la validare, deoarece toate clasele sunt bine balansate. Diferența se vede la setul de testare care a fost normalizat. Rezultatele obținute sunt:

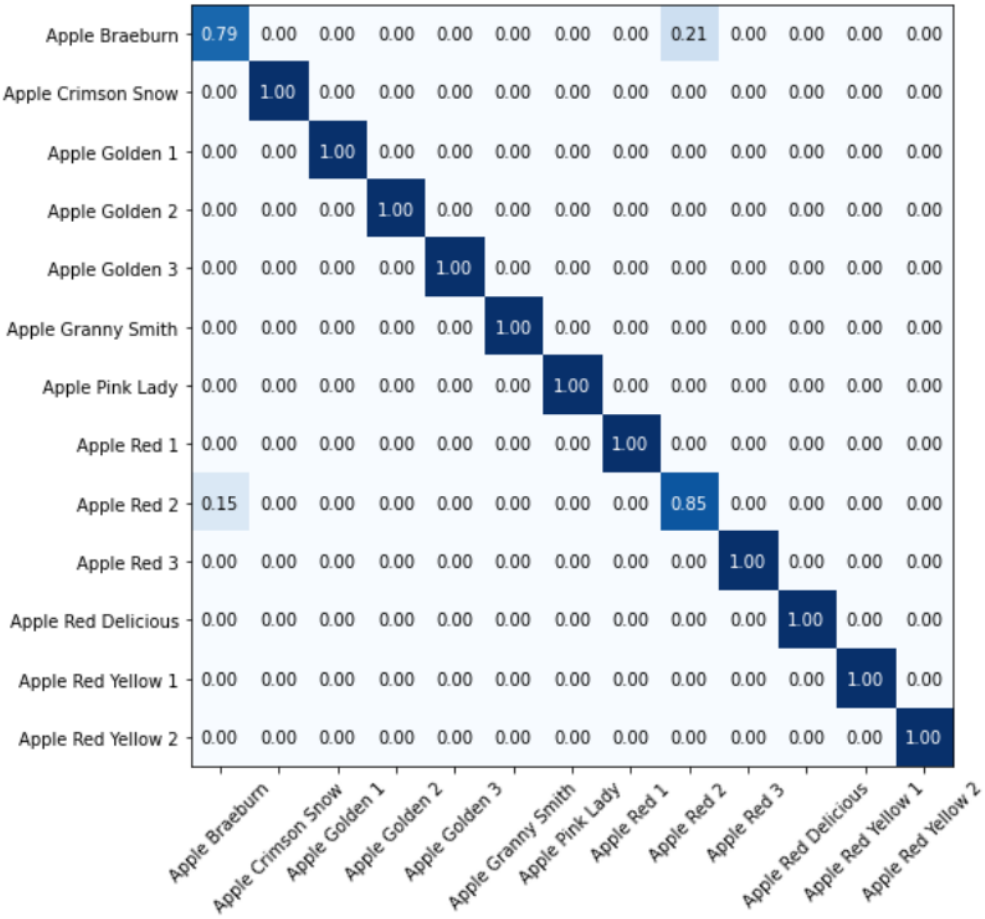
CNN	simplu	1 strat dens	2 straturi dense
loss	0.1131	0.1196	0.0956
accuracy	0.9724	0.9813	0.9709

Tabel 2 - Valori loss și acurateţe CNN (simplu, un strat, două straturi)

Din tabelul de mai sus se observă că modelul evaluat pe setul de testare care are cea mai mare acuratețe este cel cu un strat dens urmat de cel fără straturi, dar valoarea loss cea mai mică o are cel cu două straturi dense.

Din matricele de confuzie se observ că toate modelele clasifică greșit merele braeburn cu mere roșii de tipul 2 cu procentele de 13%, 21% și 22%. În afară de modelul cu un strat, celelalte modele mai clasifică greșit merele roșii de tipul 2 cu cele de tipul 3. Aceste

clasificări incorecte nu sunt așa de semnificative deoarece în cazul merelor roșii de tipul 2 și tipul 3 fac parte din aceeași categorie de mere roșii.



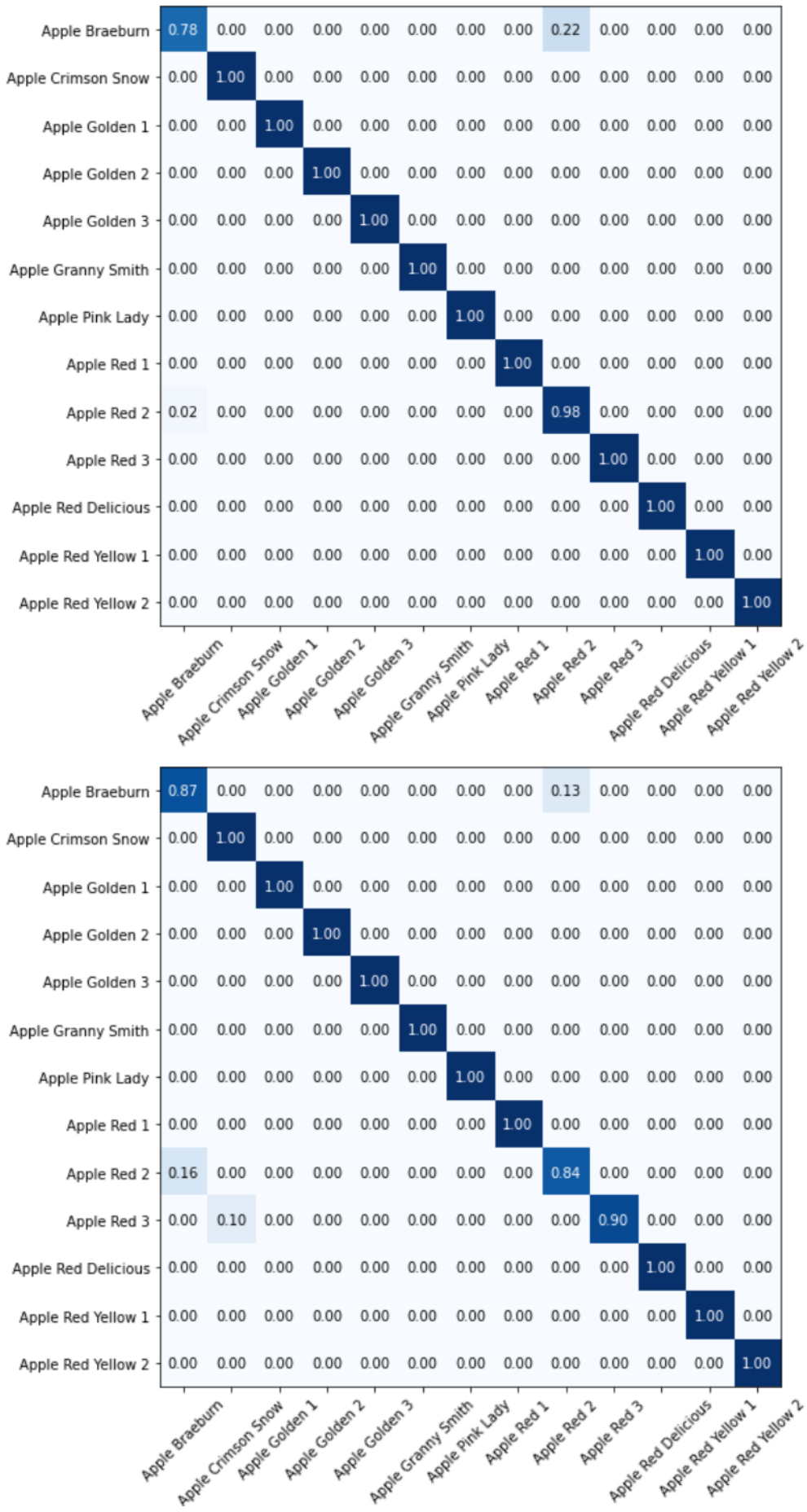


Figura 12 - Matrici de confuzie CNN (simplu, un strat dens, doua straturi dense)

Transfer de învățare

Vom antrena MobileNetV2, VGG16 și Xception în toate cele trei cazuri. Prima data se va îngheța arhitectura de bază și se va antrena la 50 de epoci cu optimizatorul Adam și callback-ul ReduceLROnPlateau. După cele 50 de epoci vom dezgheța toate straturile si îl vom antrena până la 75 de epoci cu o rată de învățare foarte mică de 1e-5. Se va folosi callback-ul EarlyStopping pentru a opri antrenamentul atunci când modelul începe după trei epoci să nu mai generalizeze bine setul de date.

Arhitectura VGG16

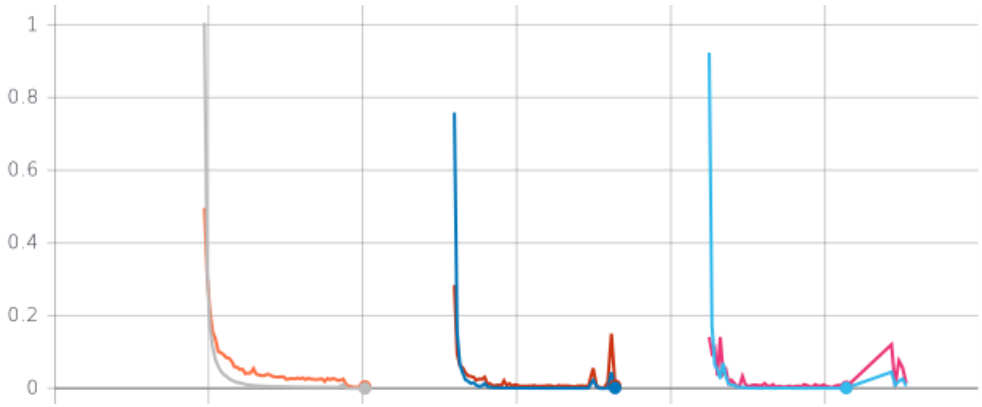


Figura 13 - Valorile loss la antrenare VGG16 (simplu, un strat dens, două straturi dense)

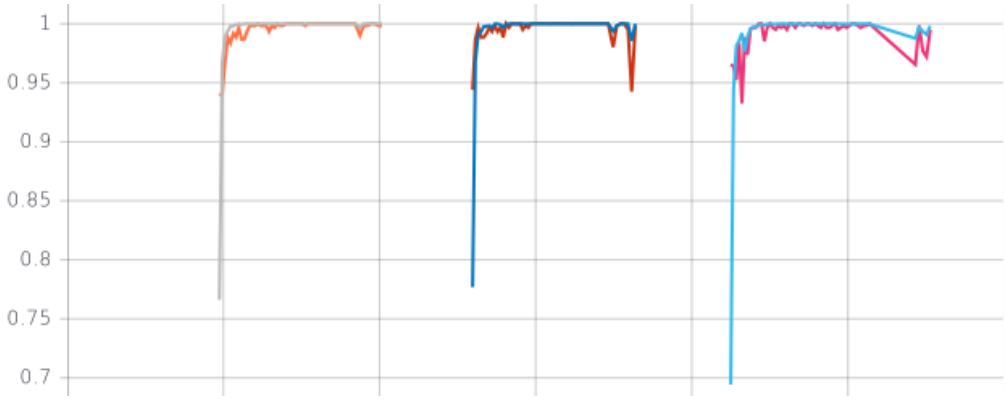


Figura 14 - Valorile acurateței la antrenare VGG16 (simplu, un strat dens, două straturi dense)

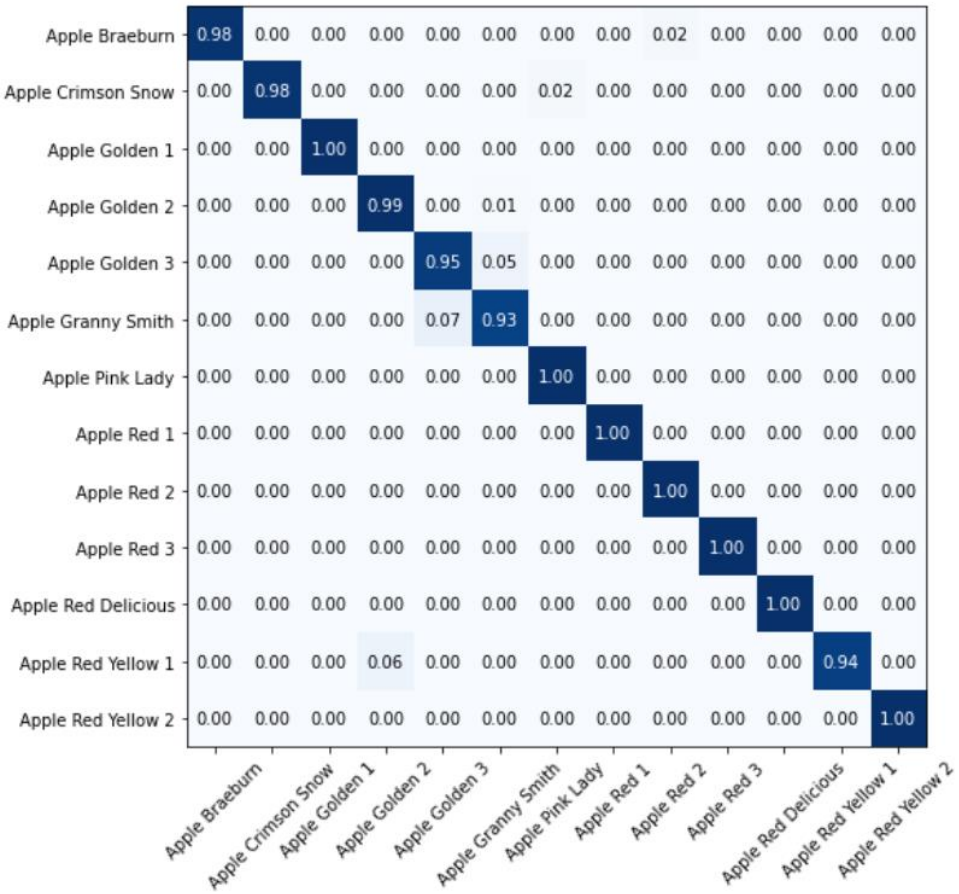
Din figurile de mai sus se poate observa cum modelul VGG16 atinge mult mai rapid valori mai bune decât în modelul anterior. În an partea a doua cu toate straturile dezghețate se observă mici fluctuații la modelele cu un strat dens și cu două straturi dense. Nici un model nu a trecut de 60 de epoci de antrenament, primele două oprindu-se la 56 de epoci, iar ultimul la 54 de epoci.

	VGG16 50 de epoci			VGG16 peste 50 de epoci		
	simplu	un strat	2 straturi	simplu	un strat	2 straturi
Loss	0.0667	0.0274	0.0411	0.0135	0.0388	0.0317
Accuracy	0.9827	0.9902	0.9799	0.9986	0.9836	0.9855

Tabel 3 - Valorile loss și acurateţe VGG-16 (simplu, un strat, două straturi)

Din tabelul de mai sus putem observa că valorile acurateței sunt mult mai mari în orice caz față de modelul fără transfer de învățare, precum și valorile loss care sunt mai mici. Se poate vedea o îmbunătățire a acurateței și a valorii loss în două cazuri din trei atunci când se dezgheață toate straturile și se antrenează la o rată de învățare mică. Cel mai bun model după dezghețarea toate straturilor este cel simplu cu cea mai mare creștere a acurateței de 0.0159 și cea mai mare scădere a valorii loss.

Din figura de mai jos se poate observa că datorită dezghețarea tuturor straturilor și antrenarea acestora la o rată de învățare mică a rezultat la eliminarea clasificării incorecte a merelor golden cu cele granny smith, dar și a merelor golden cu cele roșii galbene. Aceste categorii clasificate incorect sunt rezolvate în modelul antrenat la 56 de epoci, dar acesta începe să clasifice incorect 0.01% din merele granny smith cu cele roșii.



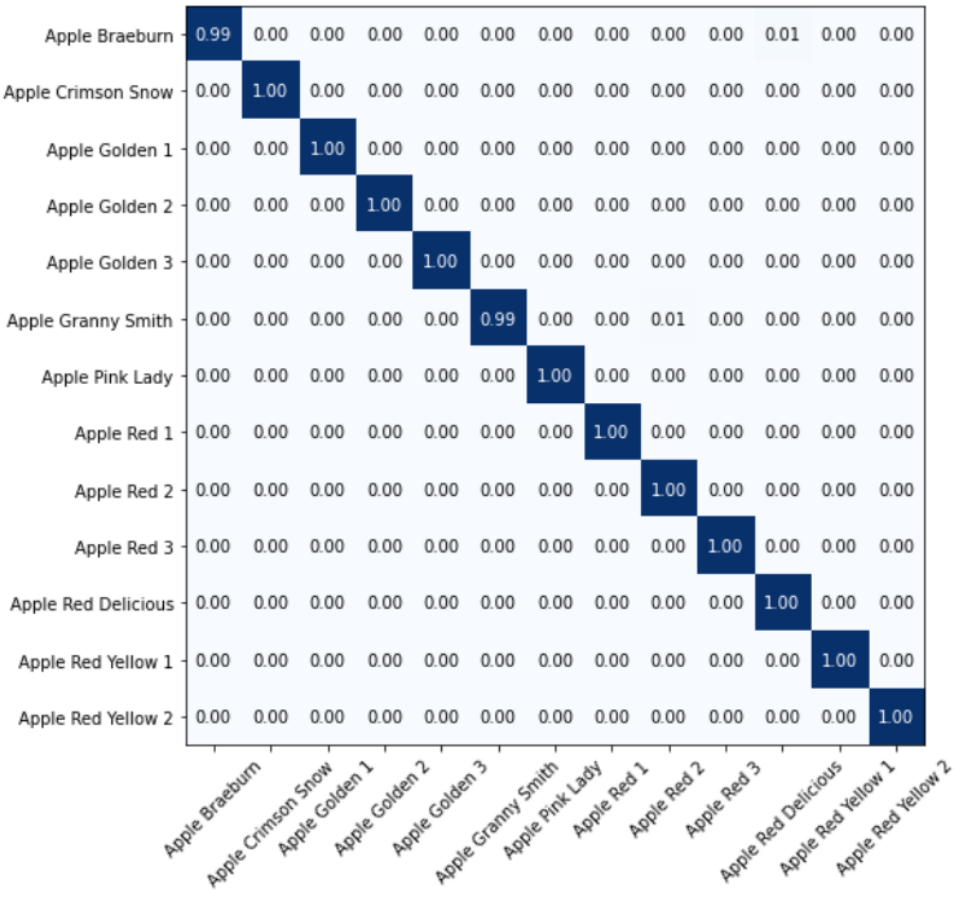


Figura 15 - Matrici de confuzie VGG16 pentru modelul simplu 50 de epoci și cel de 56 de epoci

Arhitectura Xception

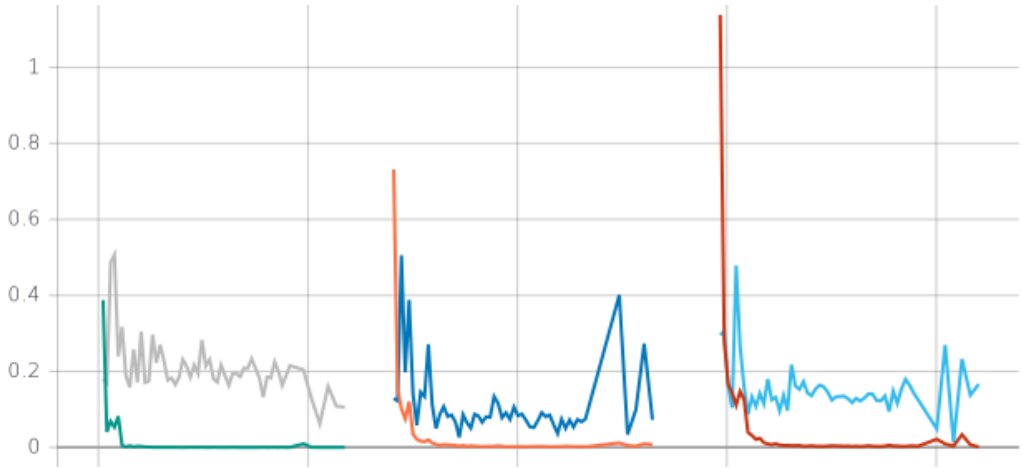


Figura 16 - Valorile loss la antrenare Xception (simplu, un strat dens, două straturi dense)

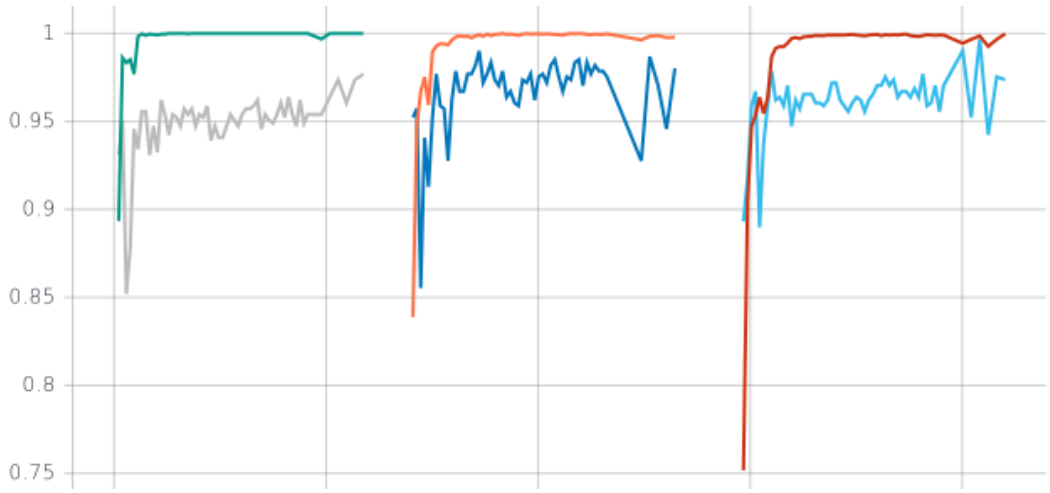


Figura 17 - Valorile acurateței la antrenare Xception (simplu, un strat dens, două straturi dense)

În general, arhitectura Xception are acuratețea cea mai bună, dar în acest caz se poate observa din grafice că este mai mică modele prezentate precedent. Aceste rezultate sunt realizate pe setul de antrenare și cel de validare. În modelul simplu la antrenarea cu straturile dezghețate, fluctuațiile sunt mai mici decât în cealalte modele. În antrenarea cu straturile dezghețate doar modelul cu un strat dens s-a oprit la 54 epoci, iar restul s-au oprit la 55 de straturi.

Pe setul de testare modelele Xception nu prea au reușit să clasifice așa de bine soiurile de mere. Din tabelul de mai jos, cel mai bun model este cu un strat dens, care antrenat cu straturile dezghețate se poate observa că obține acuratețea de 0.9274 și valoare loss de 0.3233. Creșterea de 0.0211 este destul de mare comparativ cu cealalte modele prezentate. În antrenarea cu straturi dezghețate doar modelul cu un strat dens a avut o creștere a acurateței, cealalte scăzând puțin în acuratețe.

	Xception 50 de epoci			Xception peste 50 de epoci		
	simplu	un strat	2 straturi	simplu	un strat	2 straturi
Loss	0.5870	0.4758	0.3764	0.6638	0.3233	0.3785
accuracy	0.9128	0.9063	0.9096	0.9110	0.9274	0.9063

Tabel 4 - Valorile loss și acuratețe Xception (simplu, un strat, două straturi)

În matricele de confuzie de mai jos se observă că modelul Xception cu un strat dens antrenat la 50 de epoci clasifică greșit merele crimson snow cu pink lady și mere roșii. O clasificare incorectă nouă până acum este cea a merelor golden 2 cu cele granny smith, unde culorile lor sunt diferite, iar pe lângă aceasta mai este o clasificare greșită a merelor roșii de tip 1 cu cele de tip 2 și pink lady. Modelul Xception cu straturile dezghețate reduce clasificările incorecte scapând de câteva, dar tot rămân predicții incorecte în clasele golden 3, granny smith și mere roșii.

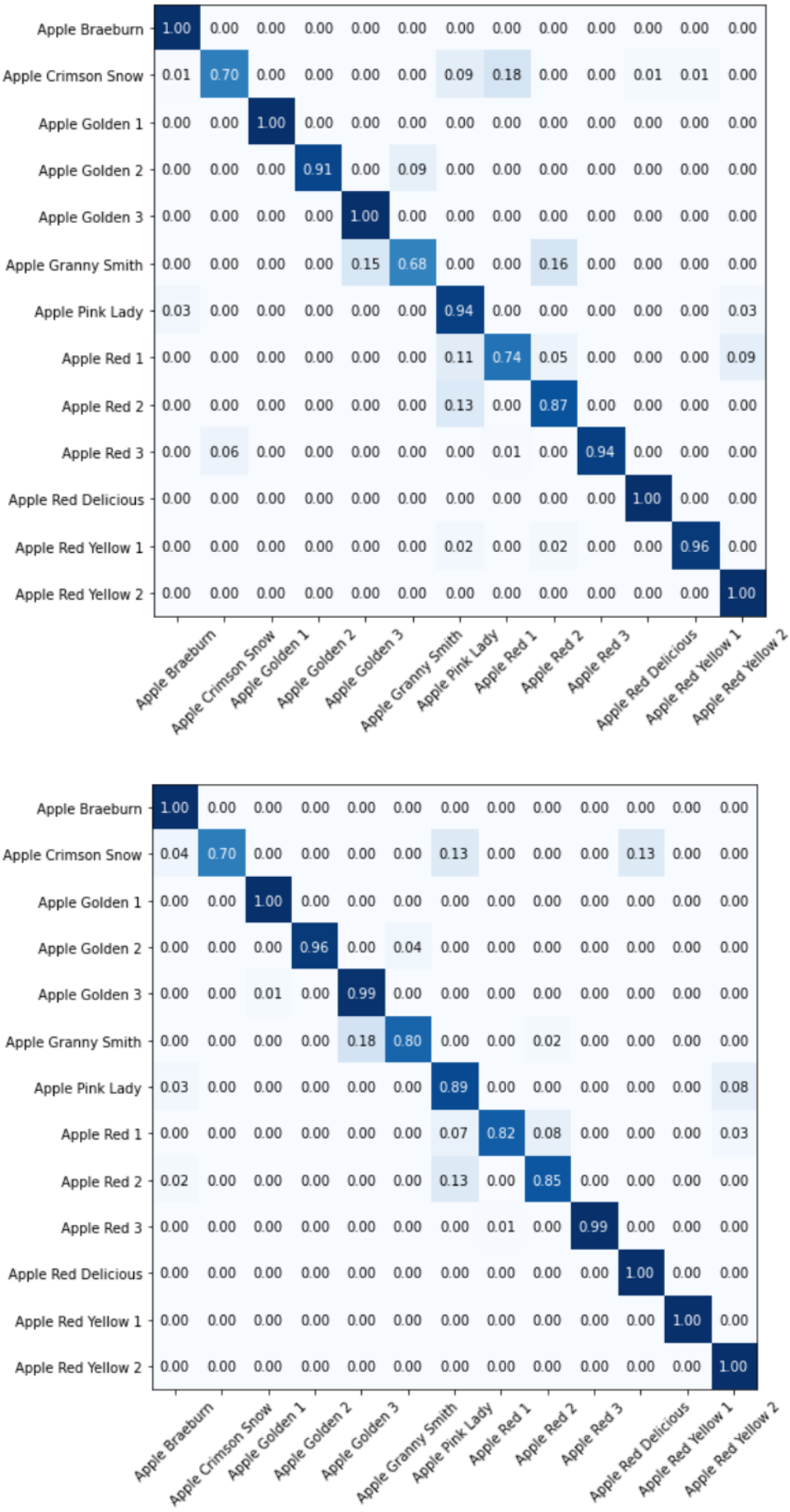


Figura 18 - Matrici de confuzie Xception cu un strat dens 50 de epoci și 55 de epoci

Arhitectura MobileNetV2

Ultimul model este MobileNetV2, care este cel mai mic în dimensiuni. Din graficele de mai jos se observă că modelul simplu ajunge cu acuratețea și acuratețea la validare la valorea de 1, ceea ce semnifică o generalizare proastă a categoriilor. În celelalte cazuri creste treptat acuratețea. Un lucru important de menționat este că doar în primul model după dezghețarea straturilor apare o fluctuație de 0.04 în grafic. Cu straturile dezghețate modelul simplu s-a oprit la 59 de epoci, cel cu un strat simplu la 53 epoci și cel cu 2 straturi dense la 54 epoci.

Figura 19 - Valorile loss la antrenare MobileNetV2 (simplu, un strat dens, două straturi dense)

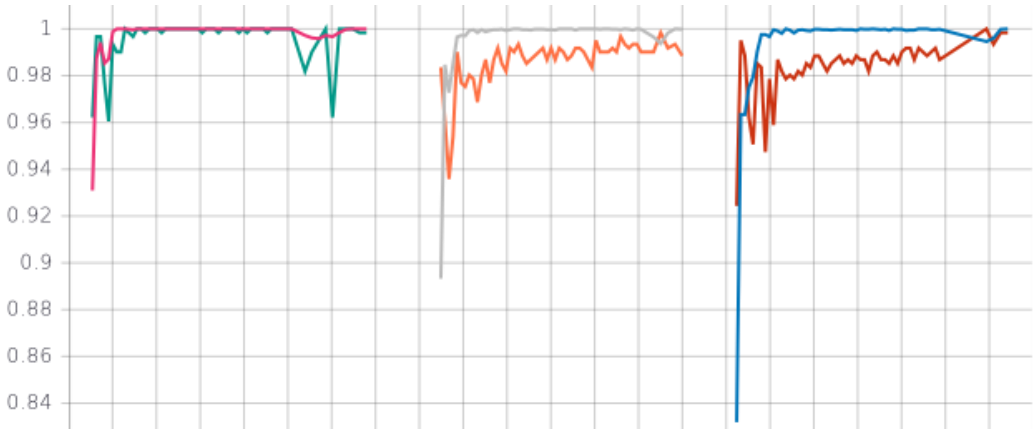


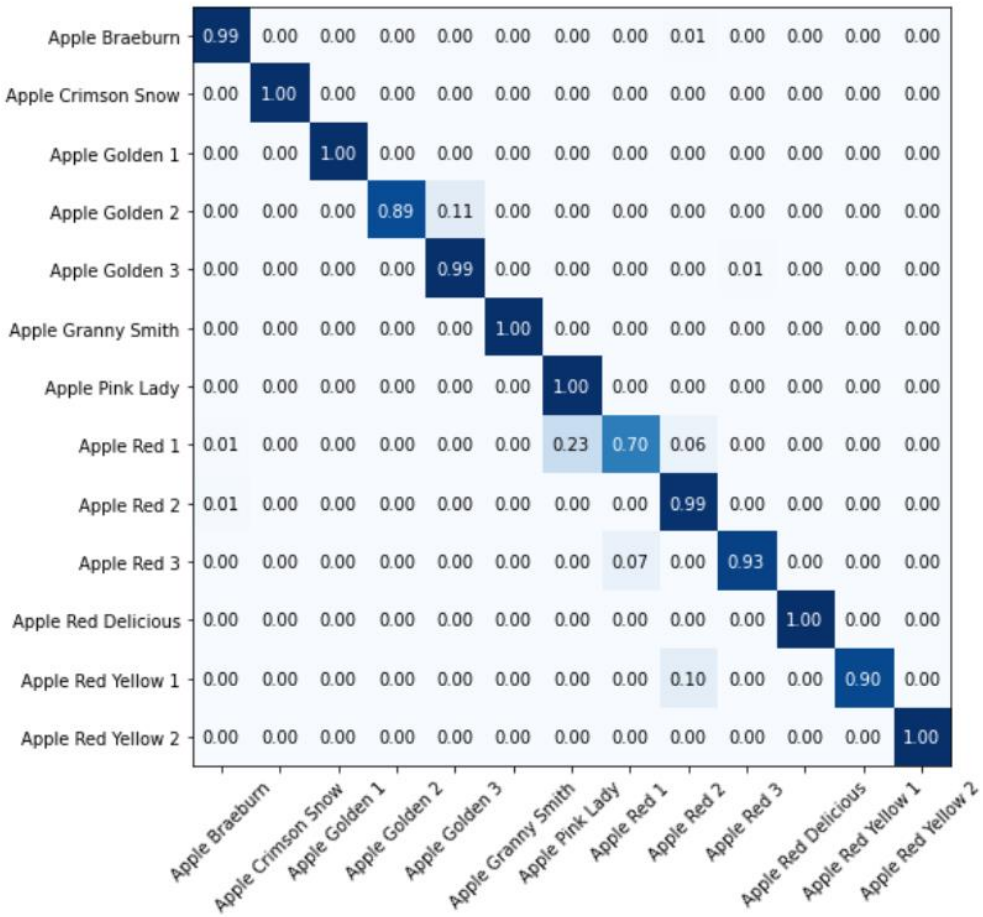
Figura 20 - Valorile acurateței la antrenare MobileNetV2 (simplu, un strat dens, două straturi dense)

Rezultatele modelelor pe setul de testare a fost unul mai bun decât cel al modelelor Xception. Putem observa că în toate cazurile modelele au avut creșteri în acurateț atunci când au fost antrenate cu straturile dezghețate. Chiar dacă nu e cel mai bun model dintre aceste cazuri, modelul fără straturi dense are o creștere de 0.0370 în acurateț. Acest model este și cel care a avut cele mai multe epoci de antrenare cu straturile dezghețate. Cel mai bun model este cel cu un strat dens.

	MobileNetV2 50 de epoci			MobileNetV2 peste 50 de epoci		
	simplu	un strat	2 straturi	simplu	un strat	2 straturi
loss	1.0261	0.2606	0.3126	0.4123	0.1307	0.1739
accuracy	0.9072	0.9545	0.9428	0.9442	0.9700	0.9574

Tabel 5 - Valorile loss și acurateţe MobileNetV2 (simplu, un strat, două straturi)

Din matricele de confuzie reiese că în ambele cazuri clasificări incorecte în categoriile mere roșii de tipul 1, 2 și pink lady, mere golden de tipul 2 cu cele de tipul 3 și mere roșii galbene cu mere roșii. Modelul antrenat cu straturile dezgheţate scade semnificativ clasificările incorecte ale merelor golden de tipul 2 cu tipul 3 , a merelor roșii de tipul 1 cu cele de tipul 2 și mere pink lady, dar creşte procentul de clasificări incorecte ale merelor roșii galbene cu mere roșii de tipul 2.



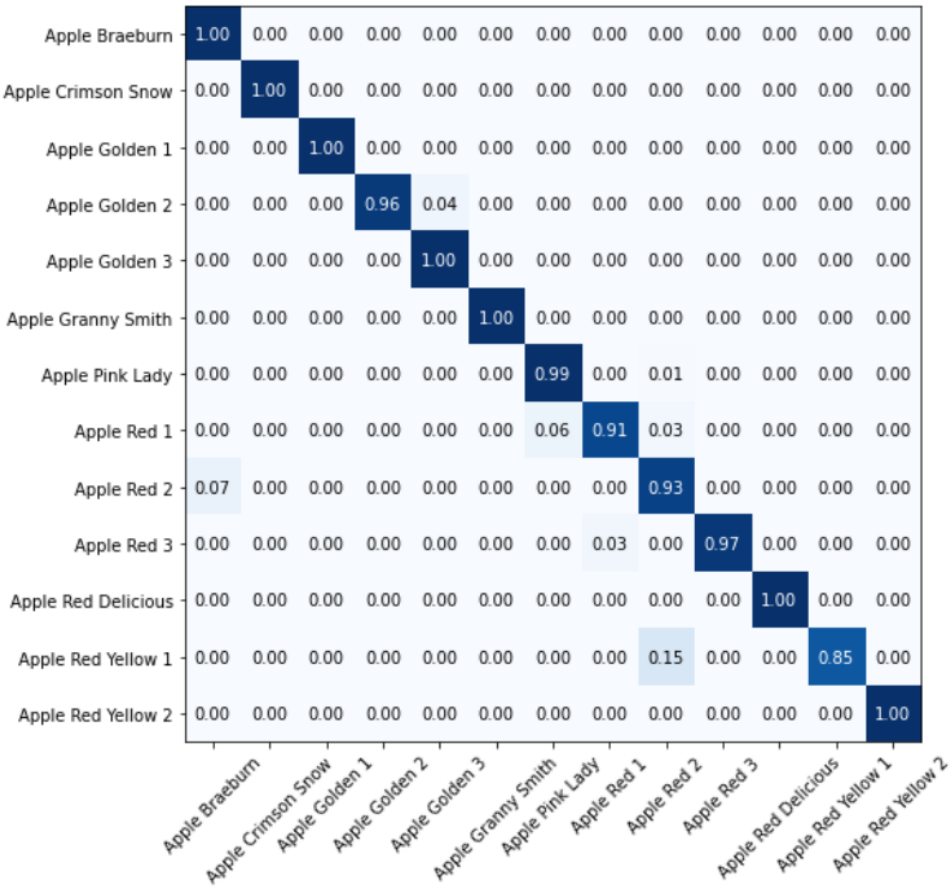


Figura 21 - Matrici de confuzie MobileNetV2 cu un strat dens 50 de epoci și 53 de epoci

În concluzie modelele cele mai bune pentru acest set de date sunt cele cu un strat dens, iar simplu în cazul modelului VGG16. Toate aceste modele au avut creșteri ale acurateței atunci când au fost antrenate cu straturile dezghețate.

Ca un ultim test vom lua cele mai bune patru modele și vor prezice opt poze cu mere. Merele sunt de soiuri diferite. Unele mere au fundalul alb uniform, iar alte mere au fundalul reprezentat de o foaie albă sau o masă.

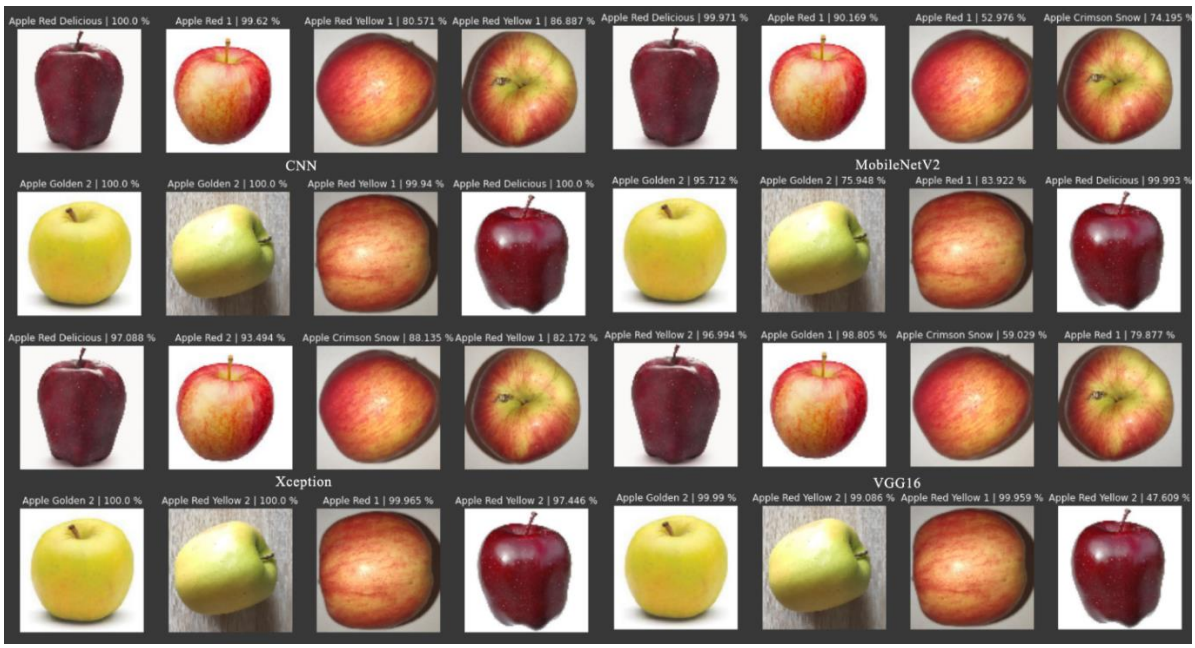


Figura 22 - Comparație ale celor patru modele pe imagini care nu se află în setul de date

Singura imagine clasificată corect de toate modelele este cea cu mărul golden cu fundal alb și cu probabilitatea între 95%-100%. Mărul golden pe masă a fost clasificat corect de CNN și MobileNetV2, iar cealalte două modele l-au clasificat ca un măr roșu galben 2. Cele mai multe mere au fost clasificate corect de către CNN și MobileNetV2. Singurele diferențe dintre aceste două modele sunt probabilitățile categoriilor și că MobileNetV2 a clasificat diferit 1 măr din cele 3 imagini cu același măr în alte poziții. Categoriile prezise la cele 3 imagini cu același măr sunt mere roșii galbene 1, crimson snow și roșii 1, aceste categorii nu sunt așa de rele pentru că se aseamănă mult intre ele. Cele mai greșite preziceri sunt la VGG16 și Xception cu mărul roșu delicios care a fost clasificat ca și măr roșu galben 2, la mărul golden pe masă cu măr roșu galben 2 și la mărul roșu cu mărul golden 2.

Prin urmare, cea mai bună configurație pentru acest set de date o reprezintă cea cu un strat dens. În privința clasificării, setului de testare modelul VGG16 a avut cea mai mare acuratețe, iar la imagini care au fost clasificate pentru prima dată de modele cel mai bun a fost modelul prezentat în Structura arhitecturii rețelei neuronale convoluționale.

3.4.2 Clasificare tipurilor de fructe

Subcapitolul va conține o clasificare a patru categorii extrase din fruits-360 [3] de fructe cu un set de antrenare nebalansat și balansat. Fructele alese sunt: mere, pere, banane și portocale. Setul de antrenare nebalansat va fi format din toate categoriile combinate de soiuri de fructe. Vor mai fi prezentate două seturi de antrenare balansate. Primul set este format din categorii asemănătoare cu soiurile de fructe din Figura 2 - Imagini bine încadrate din setul de testareFigura 2, iar al doilea set este format cu ajutorul tehnicii de undersampling [11]. Vom lua setul nebalansat și vom extrage din fiecare clasă aleatoriu imagini până când numărul de imagini se aproprie de cel al clasei cu cele mai puține imagini. Seturile de testare sunt cel din Figura 2, cel extras din fruits-360 [3] și cel cu fructe neîncadrate bine.

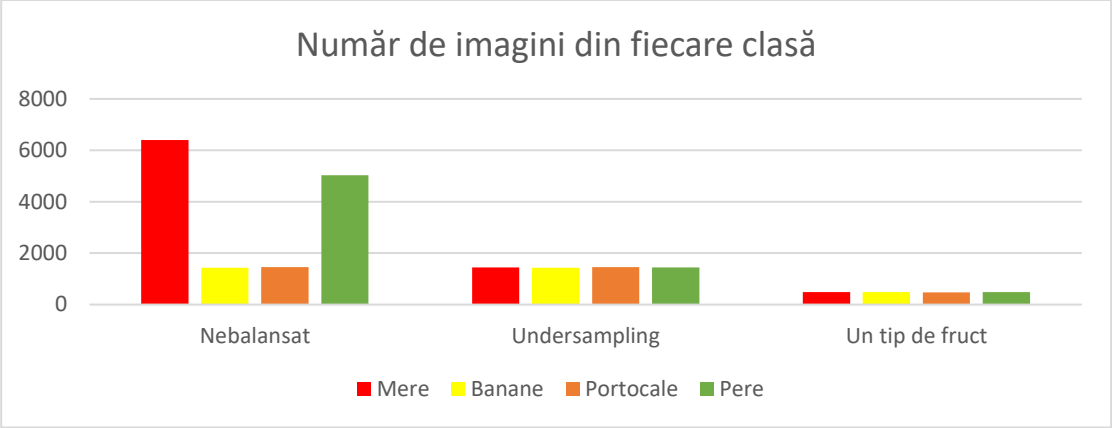


Figura 23 - Setul de date cu fructe neîncadrate

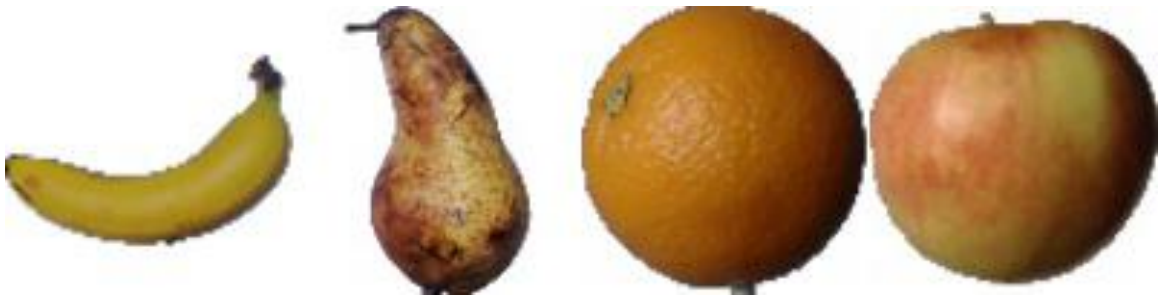


Figura 24 - Exemplu de imagini din setul de antrenament

Arhitecturile folosite pentru acest subcapitol sunt cea prezentată în Structura arhitecturii rețelei neuronale convoluționale și MobileNetV2. Am ales aceste modele deoarece au dimensiuni mici și le face bune în cazul clasificării fructelor în supermarket. La arhitectura de bază MobileNetV2 se va adăuga un strat dens de 256 de unități cu funcția de activare ReLU, o funcție dropout de 20% și un strat dens de 4 unități cu funcția softmax, iar la cealaltă arhitectură se va schimba numărul de unități la 4 în ultimul strat dens. Optimizatorul folosit este Adam, alături de callback-ul ReduceLROnPlateau care înjumătățește rate de învățare. Ambele modele sunt antrenate la 50 de epoci, iar MobileNetV2 se va mai antrena în continuare cu straturile dezghețate la o rată de învățare 1e-5 și folosind callback-ul Earlystopping cu rate de așteptare de 3 epoci.

Rezultate arhitectură rețea neuronală convoluțională

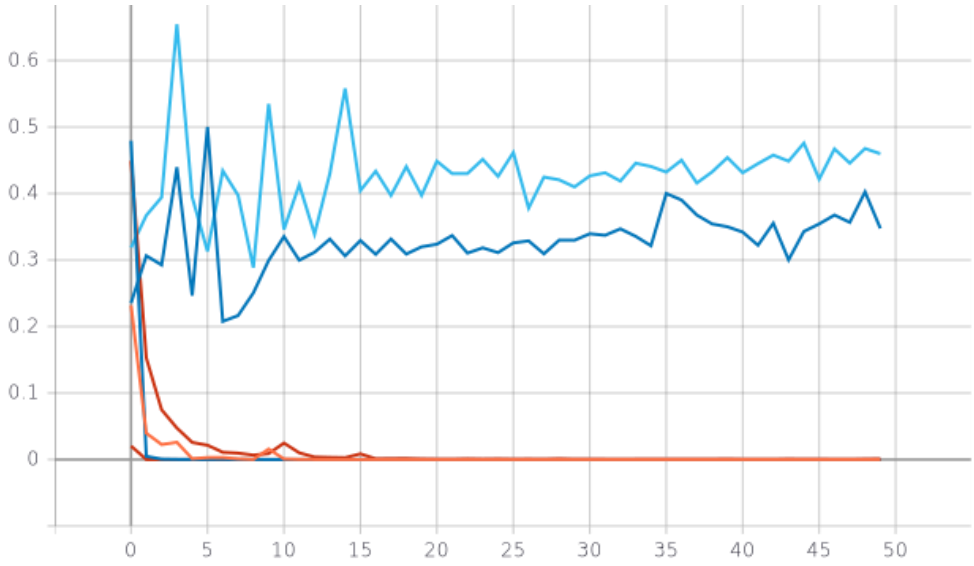


Figura 25 - Grafic cu valorile loss obținute cu cele trei seturi de date

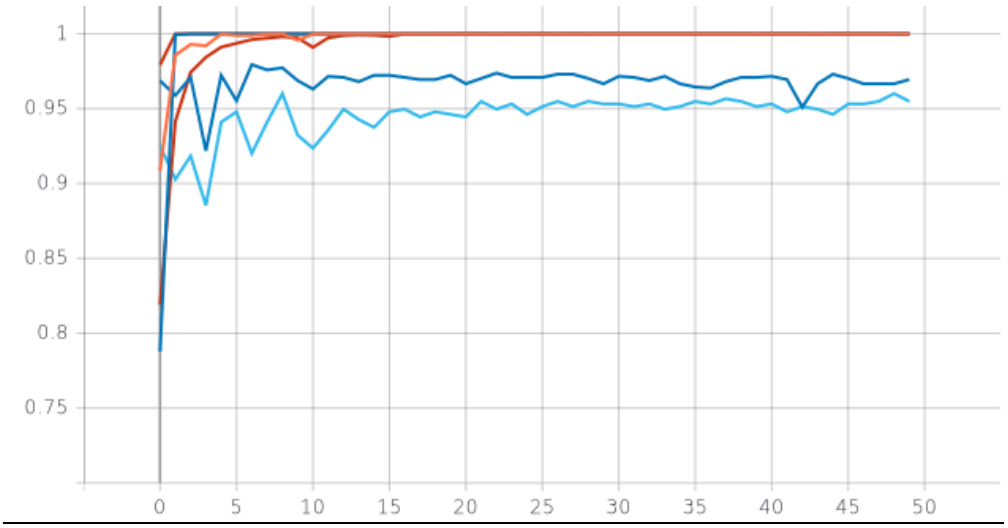


Figura 26 - Grafic cu valorile acurateței obținute cu cele trei seturi de date

Din graficele de mai sus reiese că acuratețea la antrenare atinge în toate cele trei cazuri valoarea 1, iar în cazul acurateței la validare se vede că cea mai mică acuratețe este cea a modelului antrenat pe setul nebalansat, urmat de acuratețea setului pe care s-a aplicat metoda de undersampling și cele mai bună acuratețe este cea a setului cu cele patru categorii extrase din fruits-360 [3].

	Set de testare		Set de testare neîncadrat	Set de testare bine încadrat
	Simplu	Nebalansat		
Simplu	1.000	-	0.2195	0.5122
Undersampling	-	0.9884	0.4390	0.9756
Nebalansat	-	0.9903	0.4878	0.9756

Tabel 6 - Valorile acurateței CNN (set de testare simplu, neîncadrat și încadrat)

Modelul antrenat pe setul nebalansat a avut cea mai bună acuratețe peste toate seturile de testare. Folosind metoda de undersampling am echilibrat setul nebalansat, dar din rezultate se poate observa că acuratețea a scăzut puțin în primele două seturi și în ultimul a rămas neschimbată. Setul simplu are o acuratețe perfectă la setul de testare simplu, dar acuratețea la cealalte seturi este înjumătățită față de cealalte modele. În toate cele trei modele antrenate se observă că dacă fructele sunt bine încadrate atunci acuratețea se dublează față de cele neîncadrate.

Figura 27 - Matricele de confuzie pe setul de testare nebalansat (undersampling și nebalansat)

Metoda de undersampling nu a afectat prea mult acuratețea în setul de testare, chiar dacă setul de antrenare nebalansat are cu 8000 de imagini mai mult. Ambele modele clasifică 0.03% incorect bananele cu perele și cu 0.02%/0.01% perele cu merele.

Figura 28 - Clasificare incorectă (undersampling și nebalansat)

Acestea sunt clasificările incorecte din setul cu imagini bine încadrate și în dreapta am încercat să dau exemple din setul de antrenament cu două imagini cu pere care oarecum seamănă.

Rezultate MobileNetV2

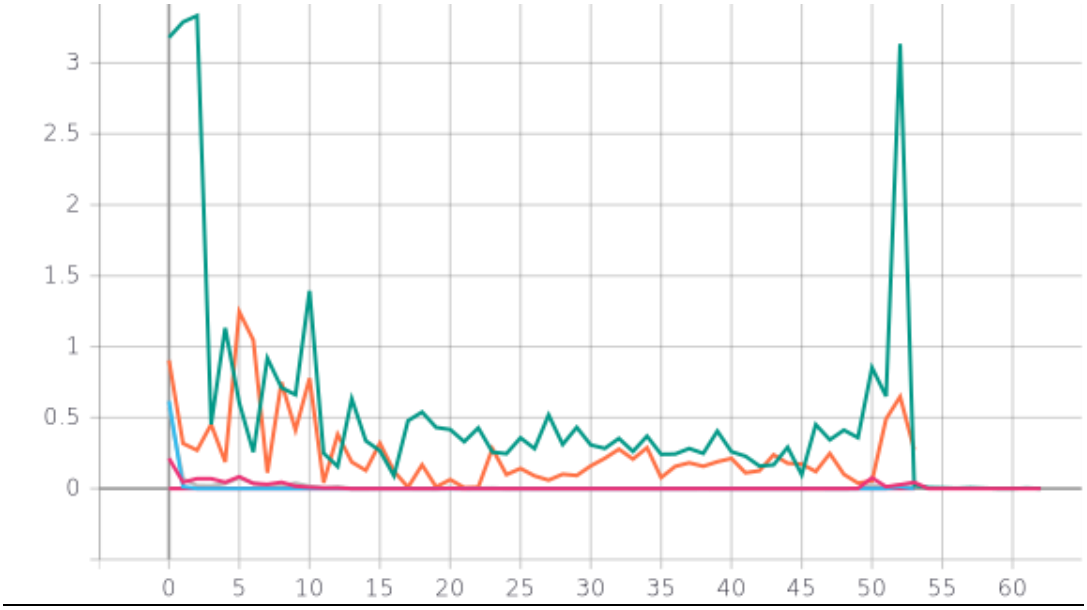


Figura 29 - Grafic cu valorile loss obținute cu cele trei seturi de date

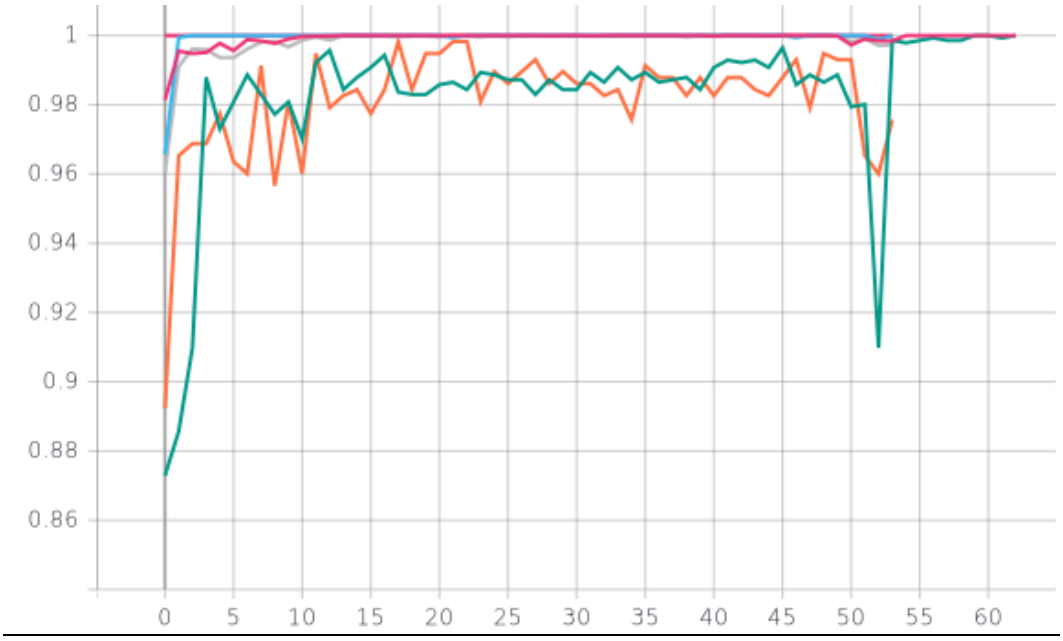


Figura 30 - Grafic cu valorile acurateței obținute cu cele trei seturi de date

Pe arhitectura MobileNetV2 se observă un comportament asemănător cu cel al arhitecturii anterioare. Modelul antrenat pe setul de date simplu are acuratețea cea mai mare. Cealalte două modele au acuratețea asemănătoare, în jur de 0.98, care este mai mare în comparație cu cealaltă arhitectură. Această acuratețe este pe setul de validare. La antrenarea cu straturile dezghețate, cele mai multe epoci le obține modelul antrenat pe setul de date simplu de 12 epoci urmat de cealalte două modele cu 3 epoci. Se observă o fluctuație mare la modelul antrenat pe setul nebalansat, iar pe setul la care s-a aplicat metoda de undersampling, fluctuația este mult mai mică.

MobileNetV2 50 de epoci	Set de testare		Set de testare neîncadrat	Set de testare bine încadrat
	Simplu	Nebalansat		
Simplu	1.000	0.7642	0.8293	0.8049
Undersampling	1.000	0.9919	0.8537	0.9512
Nebalansat	1.000	0.9986	0.9024	0.9756
MobileNetV2 Peste 50 de epoci	Set de testare		Set de testare neîncadrat	Set de testare bine încadrat
	Simplu	Nebalansat		
Simplu	1.000	0.7245	0.7561	0.8293
Undersampling	1.000	0.9909	0.7561	0.8780
Nebalansat	1.000	0.9982	0.8293	0.9512

Tabel 7 - Valorile acurateței MobileNetV2 (set de testare simplu, neîncadrat și încadrat)

Se observă că prin transfer de învățare, toate modelele obțin acuratețe dublă la setul de testare neîncadrat față de arhitectura anterioară și că antrenarea modelelor cu straturile dezghețate scade acuratețea în majoritatea cazurilor. Toate modelele antrenate au acuratețea de 100% la setul de testare simplu. La setul de testare nebalansat se observă o mică diferență de acuratețe între modelele antrenate pe setul nebalansat și setul pe care s-a aplicat undersampling, iar aceste valori sunt mai mari decât cele din arhitectura precedentă și setul simplu are o acuratețe mai mică cu 20%. La setul de testare bine încadrat se poate observa că modelul antrenat pe setul nebalansat are cea mai mare acuratețe, urmat de cel pe care s-a aplicat undersampling, iar cel simplu are o acuratețe cu mult mai mare față de cel din arhitectura precedentă. Antrenarea cu straturile dezghețate scade acuratețea în toate cazurile. La setul de testare neîncadrat cu 50 de epoci acuratețea este mai mică cu 7.5-10% față de setul de testare bine încadrat. În această secțiune cea mai mare acuratețe este obținută de setul nebalansat. Observăm că și în cazul setului bine încadrat acuratețea crește odată cu numărul de imagini, iar metoda de undersampling se pare că nu scade așa de mult din acuratețe la setul de date bine încadrat.

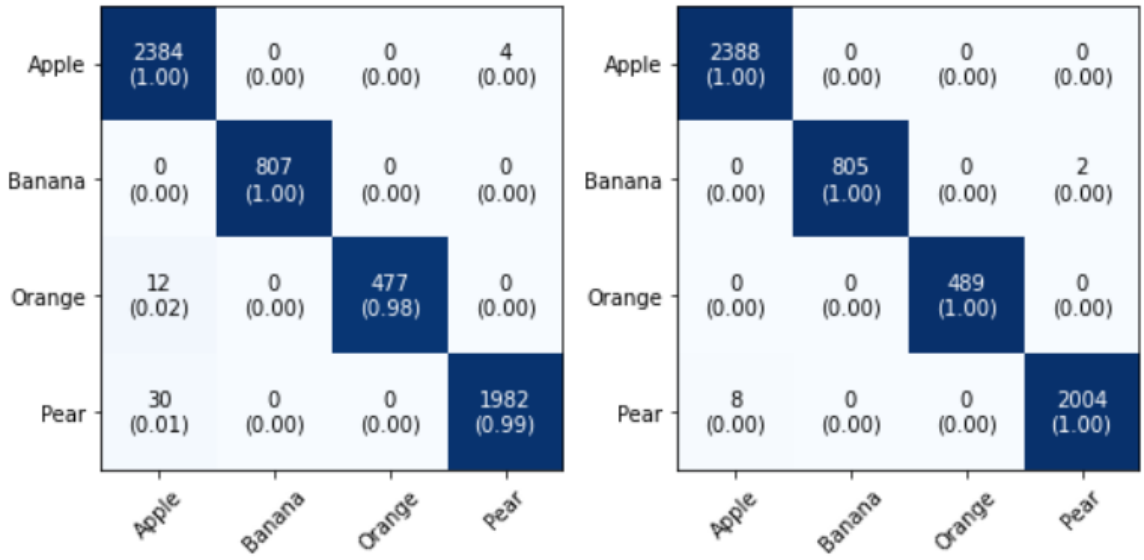


Figura 31 - Matricele de confuzie pe setul de testare nebalansat cu 50 epoci (undersampling și nebalansat)

Modelul antrenat pe setul nebalansat are rezultate clasifică greșit 2 banane ca pere și 8 pere ca mere, iar la celălalt model se observă clasificarea corectă a bananelor, dar clasificare incorectă a 4 mere cu pere și a 30 de pere cu mere.

Figura 32 - Clasificare incorectă (undersampling și nebalansat)

Același fruct clasificat greșit cu o pară a fost mărul în această poziție, iar modelul antrenat pe setul de date pe care s-a folosit undersampling a mai clasificat banana ca o pară.

În concluzie metoda de undersampling nu scade cu mult acuratețea, chiar dacă numărul de imagini care sunt extrase este mare. Setul de date simplu a dat mereu cele mai proaste rezultate atunci când a fost supus pe seturi de testare mai mari. Transferul de învățare ajută mult ca modelele să clasifice corect cât mai multe imagini, chiar dacă acestea nu sunt bine încadrate. Antrenarea cu straturile dezghețate scade acuratețea în toate cazurile. Modelul fără

transfer de învățare se descurcă la fel de bine la clasificarea fructelor bine încadrate, dar acuratețea este mică la setul de teste neîncadrat.

3.4.1 Clasificare fructelor din fruits-360

Setul de date folosit în acest subcapitol este cel din Figura 1 – Imagini din setul de date fruits-360 [3]. Acesta conține 131 de categorii de fructe și legume. Arhitecturile folosite sunt cea din Figura 3 - Arhitectură CNN și arhitecturile de bază de la MobileNetV2, Xception și VGG-16. În cazul transferului de învățare vom adaugă peste arhitecturile de bază un strat dens de 256 de unități cu funcția de activare ReLU, o funcție de dropout de 20% și un strat dens de 131 de unități cu activarea softmax. La arhitectura CNN vom schimba în ultimul strat dens numărul de unități cu 131.

Optimizatorul folosit este Adam, iar callback-urile sunt ReduceLROnPlateau, care reduce rata de învățare cu 0.5 la 3 epoci dacă modelul nu s-a îmbunătățit până la un minim de 0.00001 și EarlyStopping în cazul antrenării cu straturile dezghețate cu o rată de așteptare de 3 epoci. Toate modelele s-au antrenat la 50 de epoci, iar cele cu transfer de învățare până la un maxim de 75 de epoci cu o rată de învățare mică de 1e-5.

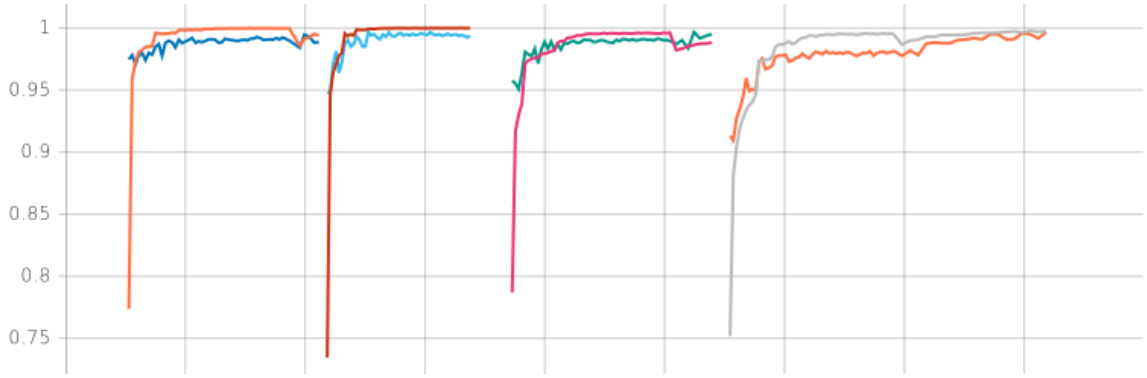


Figura 33 - Valorile loss (VGG-16, CNN, MobileNetV2, Xception)

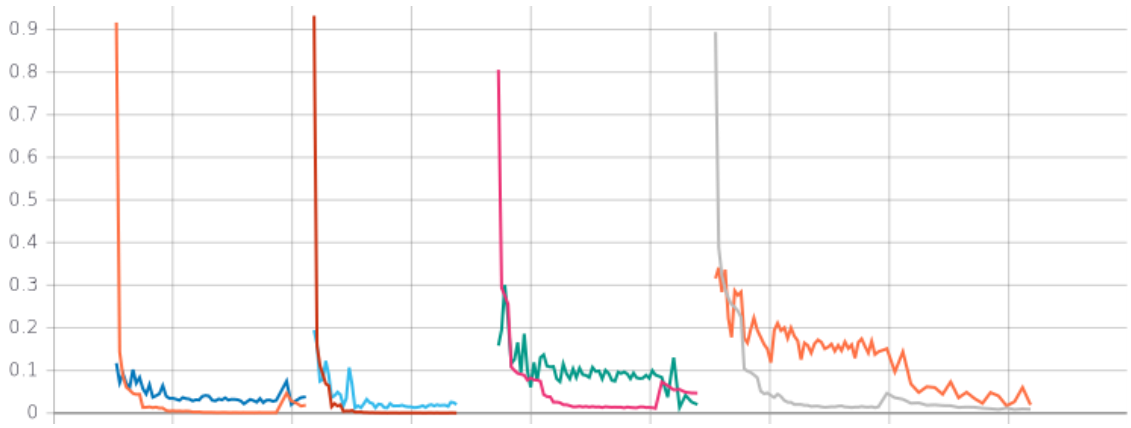


Figura 34 - Valorile acurateței (VGG-16, CNN, MobileNetV2, Xception)

Din graficele de mai sus se poate observa că cea mai bună acuratețe și cel mai bun loss îl are modelul CNN. În cazul celorlalte modele, datorită callback-ului EarlyStopping cu rata de așteptare de 3 epoci, la antrenarea cu straturi dezghețate VGG-16 s-a oprit la 54 de epoci, MobileNetV2 la 56 de epoci și Xception la 68 de epoci, iar în cazul Xception se poate observa o creștere în acuratețe la setul de validare. Toate modelele au acuratețea peste 97% și valoarea loss sub 0.1 în setul de validare.

	loss	accuracy
CNN cu 50 de epoci	0.1985	0.9833
VGG-16 cu 50 de epoci	0.1496	0.9759
VGG-16 cu 54 de epoci	0.1478	0.9721
Xception cu 50 de epoci	0.5435	0.9485
Xception cu 68 de epoci	0.1627	0.9784
MobileNetV2 cu 50 de epoci	0.3572	0.9680
MobileNetV2 cu 56 de epoci	0.2329	0.9726

Tabel 8 - Valorile loss și acurateței (modele antrenate pe 131 de categorii)

Cea mai bună acuratețe la setul de testare o are modelul fără transfer de învățare, iar valoarea loss nu este cea mai mică. În toate modelele care s-au antrenat cu straturile dezghețate se vede o creștere în acuratețe și o scădere în valoare loss, mai puțin modelul VGG-16, căruia i-a scăzut acuratețea de cu 0.0038. Se poate observa că Xception s-a antrenat multe epoci cu straturile dezghețate, rezultând o creștere de 0.0299 în acuratețe și o scădere de 0.3808 în valoare loss. După cele 68 de epoci de antrenare, modelul Xception a ajuns pe locul 2 la acuratețe, dar valoare loss fiind mai mică decât a modelului CNN. Mai jos se poate observa o figură cu imagini prezise de toate modelele pentru prima dată.

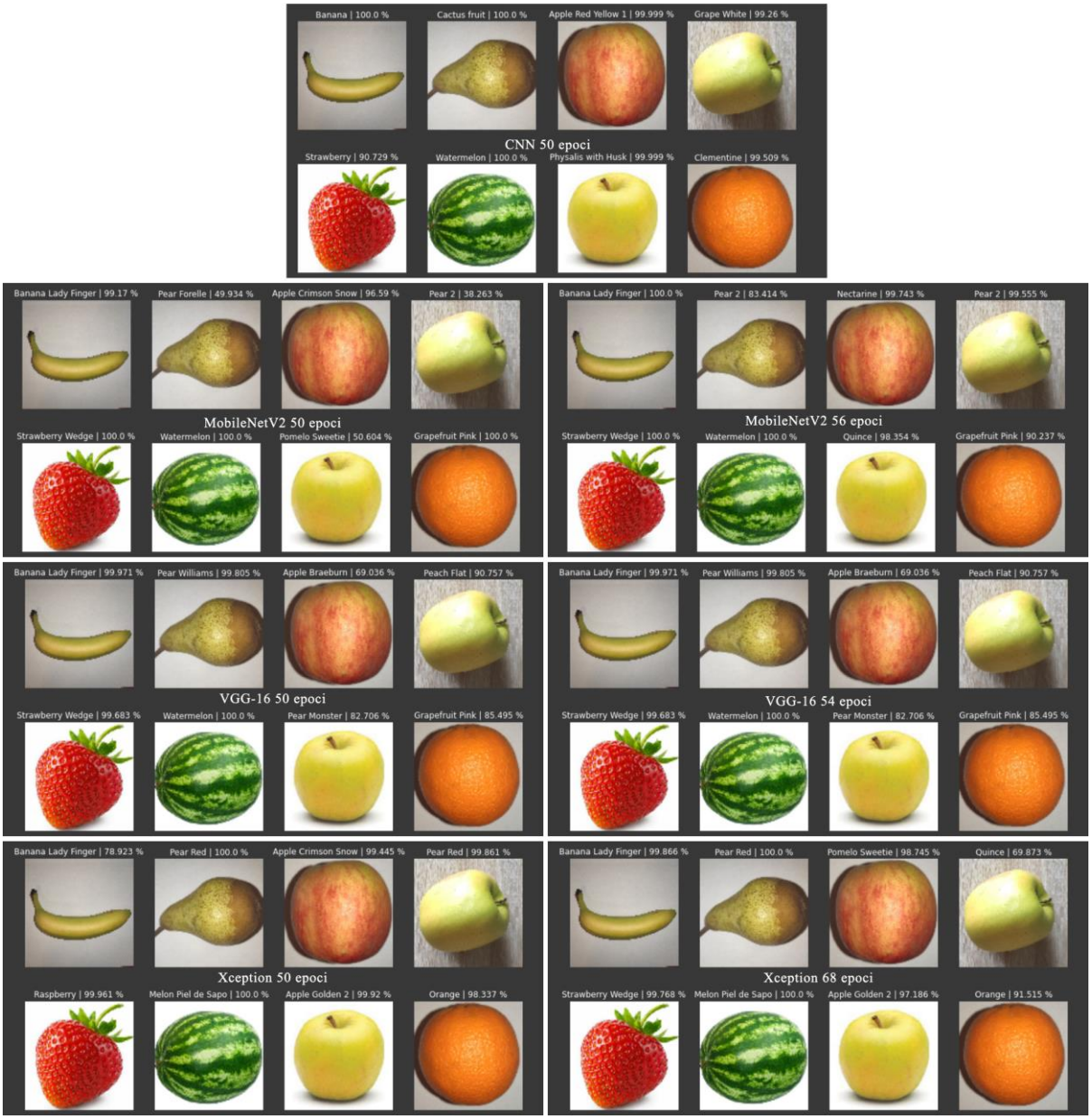


Figura 35 - Clasificare imagini cu modelul de 131 de clase

În toate cazurile se observă că banana cu fundal alb neuniform este clasificată corect. Modelul CNN a clasificat imaginea ca banană, pe când restul de modele a clasificat-o ca banana lady finger. Alta imagine clasificată bine de către toate modelele este cea cu pepene cu fundal uniform, unde modelul Xception o clasifică ca un soi de pepene, iar cealalte modele ca un pepene. Imaginea cu portocală cu fundal alb neuniform a fost clasificată ca și clementină de către un model, grape-fruit roz de către 4 modele și portocală de către 2 modele. Imaginea cu mărul golden așezat pe masă nu a fost clasificată bine de nici un model. Imaginea cu pară a fost clasificată ca și un soi de pară de modelele cu transfer de învățare și modelul CNN a clasificat-o ca și fruct de cactus. Imaginea cu măr golden si fundal alb uniform a fost clasificată corect doar de modelul Xception. Mărul roșu cu fundal alb neuniform este

clasificat corect de 5 din 7 modele. O observație importantă este că prin antrenarea modelelor MobileNetV2 și Xception cu straturi dezghețate clasificarea mărului devine incorectă.

În concluzie, modelul CNN are o acuratețe puțin mai bună pe setul de validare și setul de testare față de modelele cu transfer de învățare, dar are o acuratețe mai mică la prezis imagini pentru prima dată. În majoritate cazurilor, antrenarea cu straturile dezghețate a crescut acuratețea la setul de testare și cel de validare, dar la imagini prezise pentru prima dată există șansa ca imaginile clasificate corect să le clasifice greșit.

Concluzii

În această lucrare au fost prezentate mai multe arhitecturi de clasificare a fructelor. Cu ajutorul setului de date fruits-360 [3] și a setului de testare creat pentru această lucrare s-au putut efectua multe experimente. Primul experiment a fost cel de a clasifica diferite tipuri de soiuri de mere, unde au fost 13 categorii de mere. Pe lângă această clasificare s-au mai testat și ce arhitecturi generalizează mai bine setul de date. S-au făcut diferite teste pe rețeaua neuronală convoluțională creată și pe modelele MobileNetV2, Xception și VGG-16 cu diferite straturi dense la final pentru a observa care configurație merge mai bine, iar această configurație a fost folosită și în următoarele teste. În următorul experiment am clasificat patru tipuri de fructe: mere, pere, portocale și banane cu un set nebalansat, un set pe care s-a folosit metoda de undersampling și un set creat din patru clase din fruits-360 [3]. S-a constatat că metoda undersampling nu scade așa de mult din acuratețe, chiar dacă numărul eliminat de imagini este mare. Modelele folosite au fost cele cu dimensiuni mici. Rețeaua neuronală convoluțională și MobileNetV2 au fost folosite în acest experiment, iar acestea au dat rezultate bune pe setul de testare bine încadrat. Transferul de învățare ajută mult la clasificare fructelor care nu sunt încadrate, iar setul simplu de imagini va avea mereu acuratețe mai mică decât celelalte. În ultimul experiment s-au testat cele patru modele în clasificare setului de date fruits-360 [3]. Transferul de învățare și rețeaua neuronală convoluțională au avut acurateţe asemănătoare la setul de validare și testare, iar la clasificare imaginilor pentru prima dată, cel mai bine s-au descurcat modelele cu transfer de învățare. Câteva direcții viitoare pornind de la această lucrare ar fi:

- Implementarea unui model cu multi-input și cu multi-output, care să primească setul de date și coordonatele fructului în imagine ca și coordonate și ca output să primească predicția și coordonatele.
- Clasificarea fructelor din videoclipuri
- Clasificarea fructelor și legumelor în pungi biodegradabile, care să ajute la simplificarea procesului de identificare în cazul cumpărăturilor din supermarket.
- Clasificarea mai multor fructe dintr-o singură imagine.
- Implementarea unui model cu multi-input, care primește ca parametri setul de imagini și culoarea fructului și returnează categoria fructului.
- Clasificarea fructelor pe diferite fundaluri și pe imagini cu fructe care nu sunt bine încadrate.

Bibliografie

[1] Rojas-Aranda, J. L., Nunez-Varela, J. I., Cuevas-Tello, J. C., & Rangel-Ramirez, G. (2020, June). Fruit Classification for Retail Stores Using Deep Learning. In Mexican Conference on Pattern Recognition (pp. 3-13). Springer, Cham. (https://link.springer.com/chapter/10.1007/978-3-030-49076-8_1)

[2] Onishi, Y., Yoshida, T., Kurita, H., Fukao, T., Arihara, H., & Iwai, A. (2019). An automated fruit harvesting robot by using deep learning. Robomech Journal, 6(1), 1-8. (https://robomechjournal.springeropen.com/articles/10.1186/s40648-019-0141-2)

[3] Fruits-360 dataset (https://www.kaggle.com/moltean/fruits)

[4] Dropout for regularizing (https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/)

[5] Torrey, L., & Shavlik, J. (2010). Transfer learning. In Handbook of research on machine learning applications and trends: algorithms, methods, and techniques (pp. 242-264). IGI global. (https://ftp.cs.wisc.edu/machine-learning/shavlik-group/torrey.handbook09.pdf)

[6] Transfer Learning (https://keras.io/guides/transfer_learning/)

[7] Keras Applications (https://keras.io/api/applications/)

[8] VGG-16 architecture implementation in Keras (https://medium.com/pythoneers/vgg-16-architecture-implementation-and-practical-use-e0fef1d14557)

[9] Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1251-1258).
(https://openaccess.thecvf.com/content_cvpr_2017/papers/Chollet_Xception_Deep_Learning_CVPR_2017_paper.pdf)

[10] MobileNetV2 Architecture (https://arxiv.org/pdf/1801.04381.pdf)

[11] Imbalanced data undersampling method (https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/)