PROIECT - SISTEME DE VEDERE ARTIFICIALĂ

Tema 28 – echipa 20

Studenți:

- **Taradaciuc Nicolae**
- **Vicol Șerban – Ilie**

Grupa 1305B

Îndrumător: Drd. ing. Adrian-Paul Botezatu

Tema SVA – 28

Obiectiv: Realizarea unui sistem de vedere artificială care să permită identificarea unui logo in imagini folosind deep learning.

Etape:

1. Realizarea unui GUI pentru achiziție de imagine in timp real.
2. Antrenare clasificator neuronal.
3. Detectie logo.
4. Afisarea rezultatului in interfata grafică.
5. Concluzie.

Etapa 1: Realizarea unui GUI pentru achiziție de imagine in timp real.

Un GUI în MATLAB este o aplicație cu o interfață vizuală intuitivă, care permite utilizatorilor să interacționeze cu programul prin intermediul componentelor grafice, cum ar fi butoane, meniuri, casete de editare și grafice. Utilizarea unui GUI face mai ușoară și mai accesibilă utilizarea și controlul funcționalităților oferite de codul MATLAB.

Pentru aplicația noastră am folosit mediul de proiectare interactiv GUIDE din MATLAB cu scopul de a dezvolta o interfață GUI ce conține 4 butoane și un graf utilizat pentru afișare.

Butonul “ Antrenează rețeaua “ are rolul de a apela programul de antrenare al rețelei.

Butonul “ Alege imaginea dorită“ are rolul de a apela programul de selectare a imaginii și detectare a logo-ului.

Butonul “ Camera ON “ are rolul de deschide camera și de a detecta logo-ul.

Butonul “ Camera OFF “ are rolul de închide camera.

În partea de jos a GUI-ului avem o porțiune cu text pentru afișarea procentajului de încredere ce este constat modificat în timpul achiziției de imagini în funcție de logo-urile detectate.

Etapa 2: Antrenarea clasificatorului neuronal.

1. Colectarea și pregătirea setului de date;

   Primul pas în antrenarea rețelei noastre neurale este colectarea și pregătirea set-ului de logo-uri. Acest set conține imagini împărțite pe foldere în funcție de numele companiei de la care provin.

```
%% PARAMETERS   // Etapa 1 -> Colectarea si pregatirea setului de date.
% the size of the input images
inputSize = [227 227 3];
% training parameters
MBS=10;% mini-batch size
NEP=30; % number of epochs
% indicate the path to the training and validation images
pathImagesTrain='D:\Facultate\An 3\Semestrul 2\SVA\Proiect SVA\Proiect\images';
% full path of the folder with training images. The folder includes a separate subfolder for each class
```

2. Preprocesarea datelor;

   După colectare, este necesară preprocesarea datelor pentru a le aduce la o formă compatibilă cu rețeaua AlexNet.

   Această etapă implica redimensionarea imaginilor (227x227x3), normalizarea valorilor pixelilor, aplicarea de transformări de augmentare a datelor (translații și flip-uri) și separarea setului de date într-un subset de antrenare și unul de validare în mod aleator.

```
% create the datastore with the training and validation images
imds = imageDatastore(pathImagesTrain,'IncludeSubfolders',true,'LabelSource','foldernames');
% split the dataset into training and validation datasets
[imdsTrain,imdsValidation] = splitEachLabel(imds,0.7,'randomized');
numClasses = numel(categories(imdsTrain.Labels)); %the number of classes
% augment the training and validation dataset
pixelRange = [-30 30];
imageAugmenter = imageDataAugmenter( ...
    'RandXReflection',true, ... // reflexia in oglinda
    'RandXTranslation',pixelRange, ...
    'RandYTranslation',pixelRange);
augimdsTrain = augmentedImageDatastore(inputSize(1:2),imdsTrain, ...
    'DataAugmentation',imageAugmenter,'ColorPreprocessing', 'gray2rgb');
augimdsValidation = augmentedImageDatastore(inputSize(1:2),imdsValidation,'ColorPreprocessing', 'gray2rgb');
```

3. Definirea arhitecturii rețelei;

În integrarea rețelei AlexNet, fiind o rețea preantrenată, am exclus ultimele trei straturi deoarece erau specializate pentru setul de date original pe care a fost antrenată rețeaua. Pentru o integrare corectă, am adaptat rețeaua la cerințele noastre prin adaugarea a trei noi straturi ce vor fi discutate la inițializarea parametrilor rețelei.

```
%% DESIGN THE ARCHITECTURE  //  Etapa 3 din antrenare -> Definirea arhitecturii retelei
% load the pretrained model
net = alexnet;
% take the layers for transfer of learning
layersTransfer = net.Layers(1:end-3);
```

4. Inițializarea parametrilor rețelei;

Cele 3 noi straturi adaugate sunt fullyConnectedLayer, softmaxLayer și classificationLayer.

“fullyConnectedLayer” este un strat complet ce este determinat de numarul de logo-uri pe care dorim să le detectăm. Parametrii acestui strat, “WeightLearnRateFactor” și “ BiasLearnRateFactor” controlează rata de învățare și termenii de decalare. După testele noastre, am observat faptul că 20 este valoarea cea mai convenabilă pentru ambii parametri.

“softmaxLayer” este un strat ce normalizează rezultatele și oferă o interpretare probabilistică pentru detecția logo-urilor.

“classificationLayer” este un strat de clasificare ce este responsabil pentru atribuirea unei etichete imaginii analizate, în funcție de distribuția de probabilitate rezultată din stratul softmax.

```
% create the new architecture: the last fully connected layer is configured for the necessary number of classes

layersNew = [ %       // Etapa 4 de initializare a parametrilor retelei
    layersTransfer
    fullyConnectedLayer(numClasses,'WeightLearnRateFactor',20,'BiasLearnRateFactor',20)
    softmaxLayer
    classificationLayer];
```

5. Propagarea înainte (forward propagation);

Această etapă implică aplicarea imaginilor de antrenare prin rețea, de la stratul de intrare la stratul de ieșire.

Informațiile sunt propagate prin straturi utilizând operații matematice, cum ar fi convoluții, agregări și activări non-liniare.

În final, rețeaua produce o predicție a logo-ului prezent în imaginea de intrare.

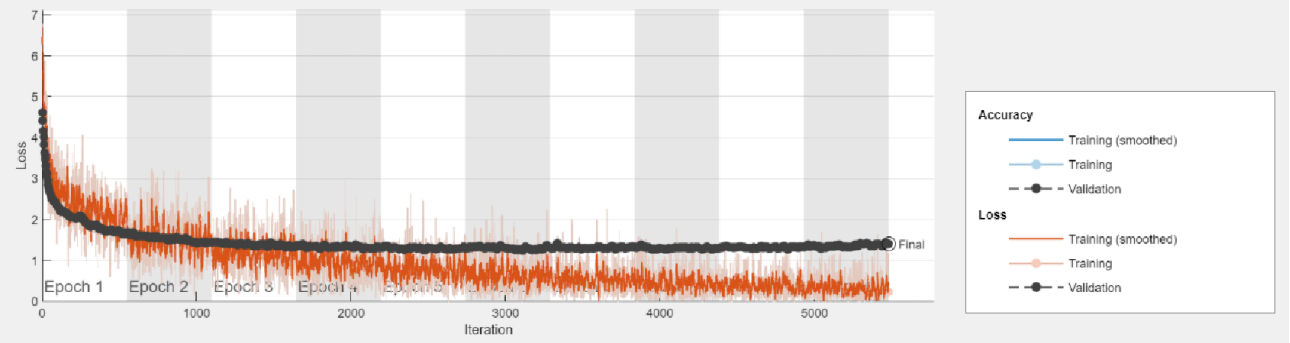
```
%% TRAIN THE CNN // Etapa 5 din antrenarea -> Propagarea inainte ( Antrenarea retelei )
% //implică aplicarea imaginilor de antrenare prin rețea, de la stratul de intrare la stratul de ieșire
% indicate the training parameters
options = trainingOptions('sgdm', ...
    'MiniBatchSize',MBS,...
    'MaxEpochs',NEP, ...
    'InitialLearnRate',1e-4, ...
    'ValidationData',augimdsValidation, ...
    'ValidationFrequency',3, ...
    'ValidationPatience',Inf, ...
    'Verbose',false, ...
    'Plots','training-progress');


% train the model
netTransfer = trainNetwork(augimdsTrain,layersNew,options);
```

6. Calculul funcției de pierdere (loss function).

Pentru a evalua cât de bine performează rețeaua, este necesară definirea unei funcții de pierdere care măsoară diferența dintre predicția rețelei și eticheta așteptată.

Acest calcul este vizibil în meniul de antrenare, fiind un parametru calculat implicit de funcția “trainNetwork”.



Etapa 3: Detecție logo.

1. Propagarea înainte (forward propagation)

Această etapa implică prelucrarea imaginii selectate pentru detectarea logo-ului prin toate straturile rețelei, informațiile fiind propagate de la intrare până la ieșire.

Stratul de ieșire produce o hartă a caracteristicilor (feature map) care evidențiază regiunile cu potențial logo.

2. Calculul scorului de detectare

După propagarea înainte, utilizând harta caracteristicilor, se aplică o operație de evaluare a scorului de detectare pentru a determina probabilitatea ca o regiune să conțină un logo. Scorul reflectă încrederea rețelei în detectarea unui logo într-o anumită regiune.

3. Suprapunerea

Pentru a evidenția regiunile cu scoruri înalte (regiunile care pot conține logo-uri), este adesea aplicată o tehnică de suprapunere (overlapping).

Aceasta implică împărțirea imaginii în regiuni mai mici și evaluarea scorurilor în fiecare regiune.

Etapele menționate anterior sunt conectate prin intermediul unei singure funcții, „classify”.

```
%Incarcarea retelei antrenate de noi
load('D:\Facultate\An 3\Semestrul 2\SVA\Proiect SVA\Proiect\fileREZ\REZtrainedAll.mat', 'netTransfer');

%Propagarea inainte se face prin reteaua netTransfer
%Se realizeaza calculul scorului si stocarea lui in scores
%Functia va utiliza imagine imag si reteaua netTransfer si va realiza suprapunerea si clasificarea
[label,scores]= classify(netTransfer, imag);
```

4. Etichetarea și identificarea logo-urilor.

Folosind funcția menționată anterior (“classify”), vom eticheta logo-ul nostru și îl vom identifica, urmând realizarea unui un top 3, în funcție de procentajul de încredere, stocat în variabila “confidencePercentage”.

```
% Get the top 3 labels and their corresponding scores
[~, sortedIndices] = sort(scores, 'descend');
top3Labels = netTransfer.Layers(end).ClassNames(sortedIndices(1:3));
top3Scores = scores(sortedIndices(1:3));

% Construct the string to display in text10
labelString = '';
for i = 1:numel(top3Labels)
    confidencePercentage = top3Scores(i) * 100;
    labelString = [labelString, sprintf('%s: %.2f%%', top3Labels{i}, confidencePercentage)];

    if i < numel(top3Labels)
        labelString = [labelString, newline];
    end
end

set(handles.text7,'String',label)
set(handles.text10, 'String', labelString);
```

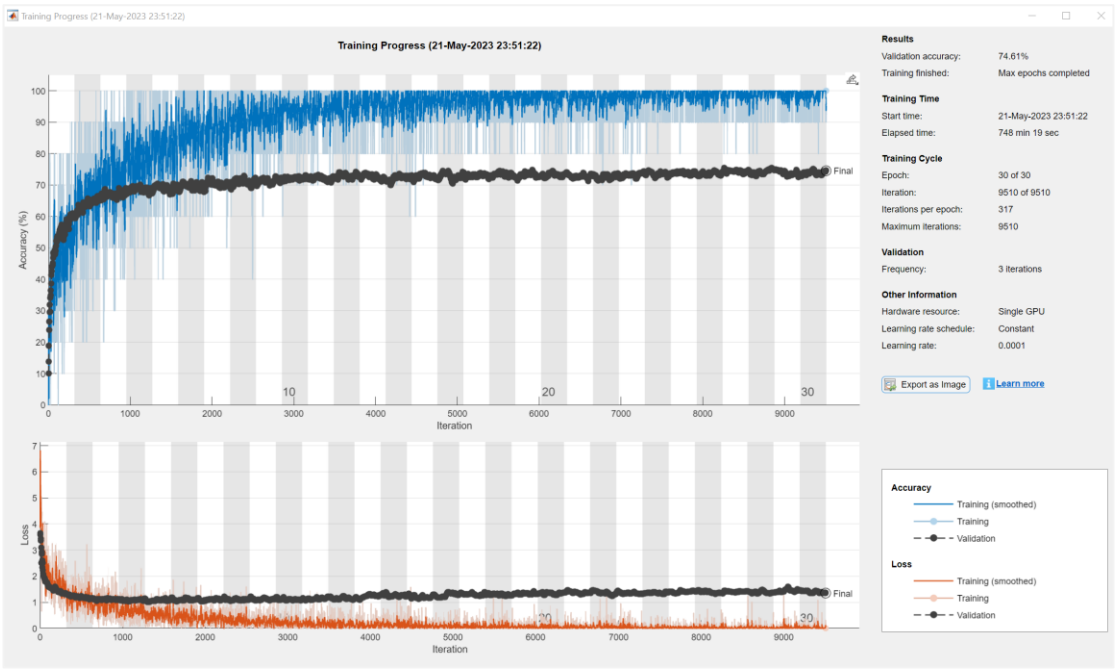
Etapa 4: Afișarea rezultatului în interfața grafică.

1. Testare a aplicației.

Aplicatia poate realiza detecție de logo-uri atât prin **selectarea unei imagini** separate cât și prin **captare de imagini în timp real**, prin intermediul unei camere video.

La apăsarea butonului “ Antrenează rețeaua ”, aplicația va activa funcția de antrenare.

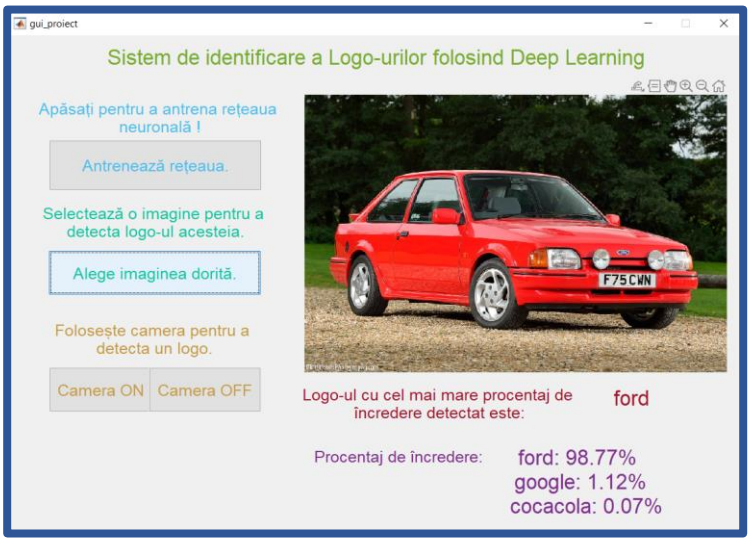
În partea de mai jos se poate observa cum arată o antrenare cu succes a rețelei.

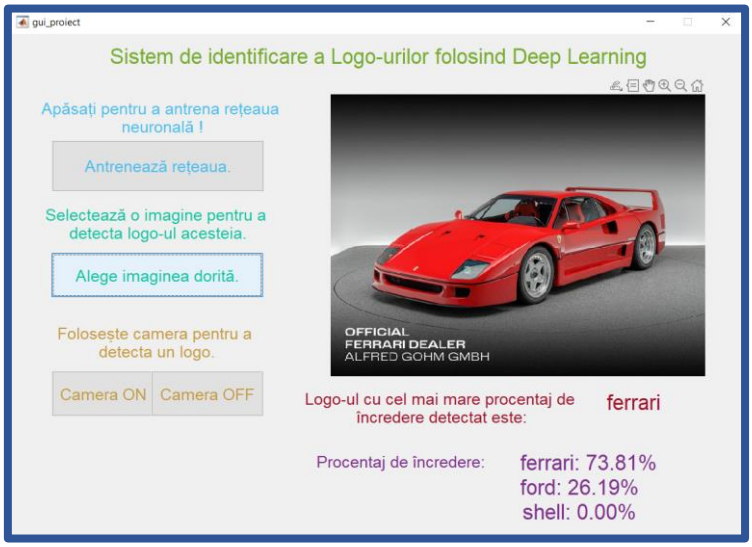


La apăsarea butonului “Alege imaginea dorită”, aplicația ne va permite să selectăm o imagine din calculator.

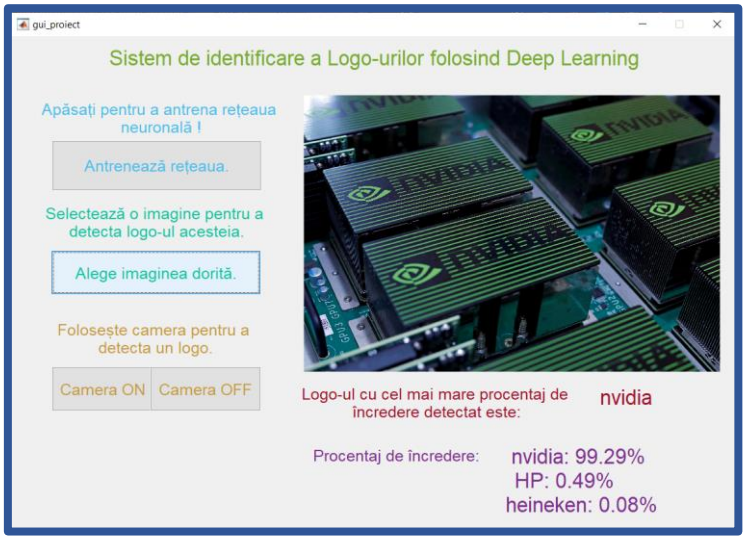
În partea de mai jos se poate observa un experiement în care am testat acuratețea rețelei antrenate de noi, cu două imagini asemănătoare din punct de vedere al culorilor.

După cum se poate observa, logo-urile au fost detectate corect.

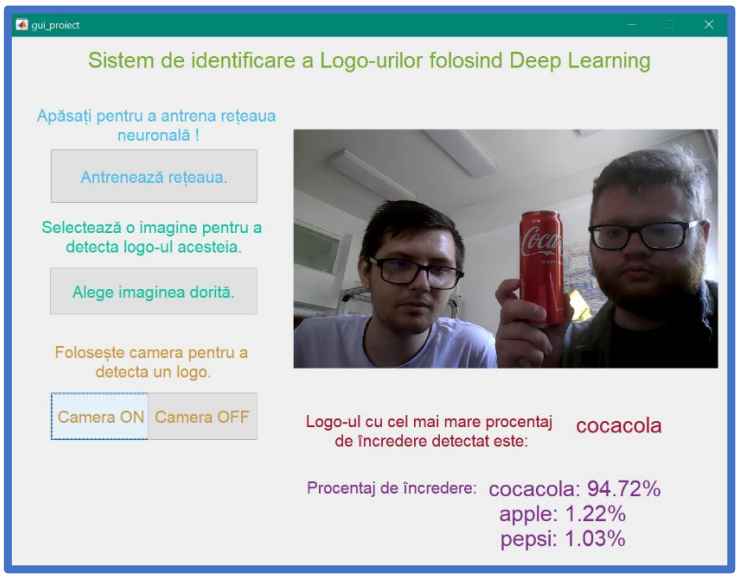


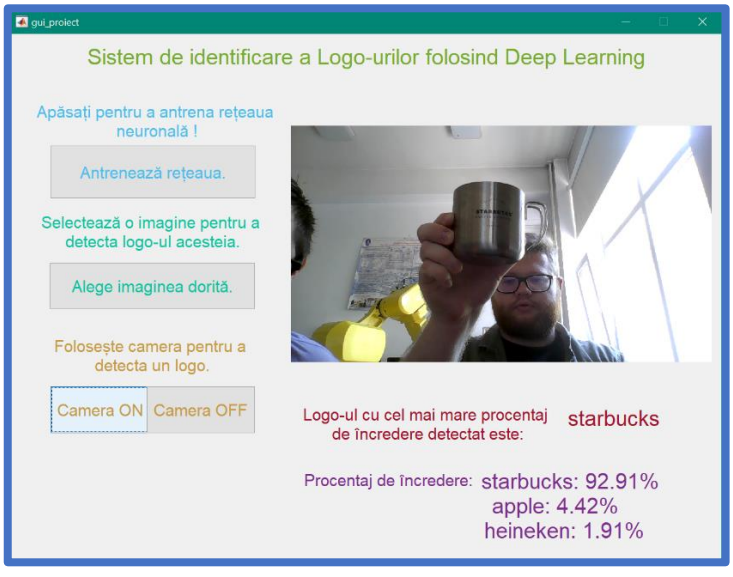


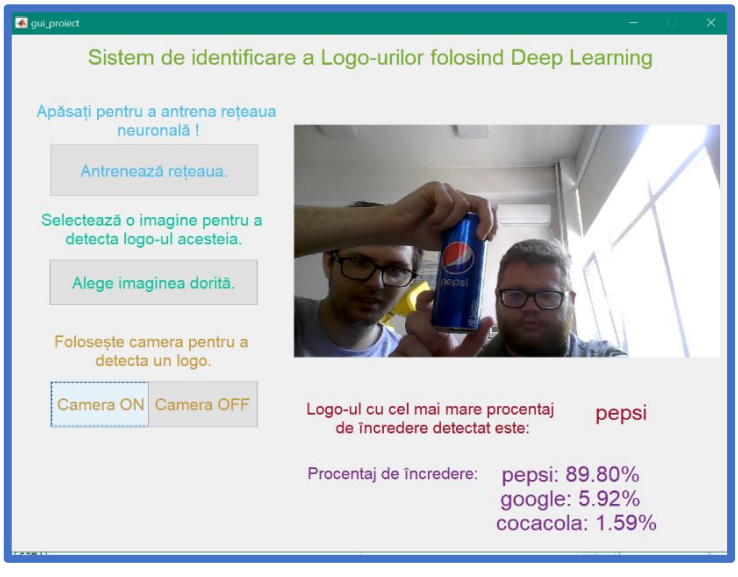
Mai jos avem si alte cazuri în care logo-ul este clar și vizibil.



În continuare vă vom prezenta o serie de experiemente realizate cu ajutorul camerei video.

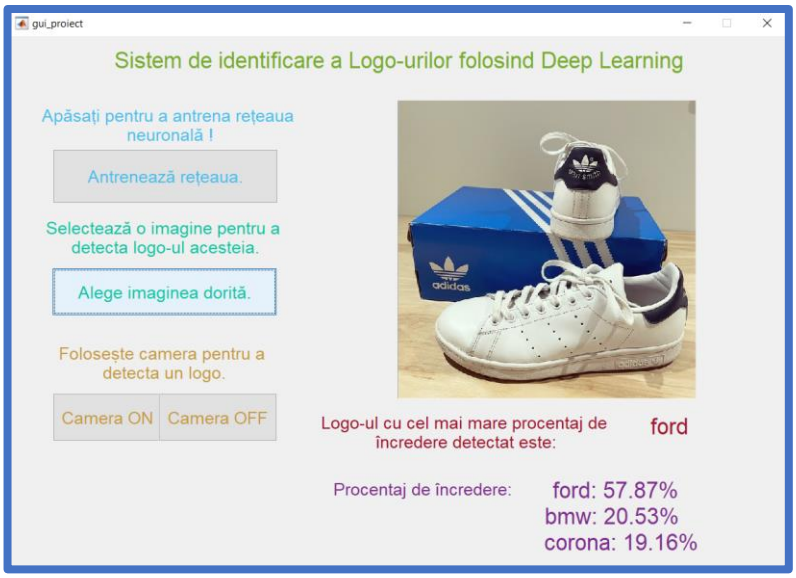






2. Obstacole întâmpinate

Prima noastră rețea antrenată cu succes a fost realizată cu 10 epoci, observând ulterior, în cadrul testelor, faptul că deseori, detecția logo-ului era eronată chiar și în cazurile în care logo-ul ocupa o mare parte din imagine.



După ce am observat faptul că rețeaua antrenată in **10 epoci** nu funcționează conform așteptărilor noastre, am decis să antrenăm o nouă rețea neuronală cu **30 de epoci.**

În timp ce rețeaua era în curs de antrenare, am estimat că aceasta ar finaliza procesul în aproximativ **20 de ore**, fapt ce era o problemă pentru noi și acesta fiind si motivul pentru care am decis să eliminăm o serie de logo-uri ale căror companii păreau a fi irelevante pentru noi.

După această modificare, timpul de antrenare a scăzut la aproximativ **13 ore.**

În momentul actual, rețeaua detectează corect un logo, în **majoritatea** cazurilor în care a fost testat.

Prin teste, am observat faptul că pozele ce conțin o diversitate mare de obiecte sunt dificil de detectat de către rețeaua noastră, chiar și în cazurile în care un sigur logo este prezent în imagine.

Etapa 5: Concluzii.

În concluzie, rețeaua antrenată de noi funcționează conform așteptărilor noastre, reușind să detecteze aproape toate logo-urile din testele realizate.

De asemenea, după cum s-a putut observa, proiectul nostru permite o mulțime de noi îmbunătățiri precum schimbarea ratei de învățare și/sau modificarea numărului de epoci.