***Content Based Image Search Engine***

Pintilie Ana Maria, Sandoiu Fernando Florin

**Abstract**

Prin realizarea acestui proiect, am demonstrat ca, folosind o retea de detectare de obiecte pre-antrenata, impreuna cu caracteristicile invatate de o retea de clasificare, putem raspunde task-ului actual de cautare de imagini pe baza continutului acestora, renuntand, astfel, la vechile tehnici de indexare a imaginilor pe baza tag-urilor construite in avans.

**Introducere**

Cautarea de imagini pe baza de continut (**Content-Based Image Retrieval** (CBIR)) este ceea ce numim un motor de cautare de imagini, insa cu proprietatea ca aceasta cautare sa fie realizata pe baza continutului imaginii, si nu pe diferite adnotari textuale asociate cu imaginea. In esenta, pentru a realiza cautarea, se utilizeaza algoritmi de extragere a caracteristicilor (“features”) unei imagini. Construindu-se o baza de date de caracteristici pentru fiecare imagine din cele puse la dispozitie pentru cautare, atunci cand utilizatorul introduce o imagine dupa care doreste sa se realizeze cautarea, sunt extrase caracteristici din aceasta pe baza aceluiasi criteriu si sunt comparate cu cele din baza de date creata anterior, returnandu-se apoi imaginile similare.

Pentru definirea unui astfel de sistem de cautare, am luat in considerare urmatorii factori:

**Ce fel de cautare vrem sa efectuam?**

Asemanator rezultatelor de la Pinterest sau Google Images, am dorit sa realizam o cautare la nivelul obiectelor identificate in imagine. Primind o imagine pentru cautare, aplicam un algoritm de detectare de obiecte pentru a le localiza si a le identifica. Ulterior, pentru fiecare obiect astfel rezultat, folosim un model pre-antrenat pentru a extrage trasaturi specifice fiecarei clase, pe baza acestora efectuandu-se o comparare pentru a intoarce imagini ce contin obiecte similare.

**Ce algoritmi de detectare de obiecte folosim?**

Am ales sa testam modul in care functioneaza cautarea cu ajutorul a doi algoritmi, si anume Faster R-CNN si Mask R-CNN. Faster R-CNN este framework-ul care raspunde cel mai bine in momentul actual task-ului de “Object Detection”. Acesta detecteaza in mod eficient un numar variabil de obiecte din clase diferite dintr-o imagine, precizand si locatia acestora printr-un bounding box. Faster R-CNN este impartit in doua etape: in prima parte se scaneaza imaginea si se genereaza regiuni de interes (locatii din imagine in care probabilitatea de a fi un obiect este ridicata), iar in cea de-a doua parte, regiunile astfel gasite sunt propuse pentru clasificare (clasa din care face parte obiectul) si detectare de bounding box (locatia unde au fost identificate obiectele). In plus, la Mask R-CNN este realizata si detectarea unei masti de pixeli (identificarea la nivel de pixeli a obiectului).

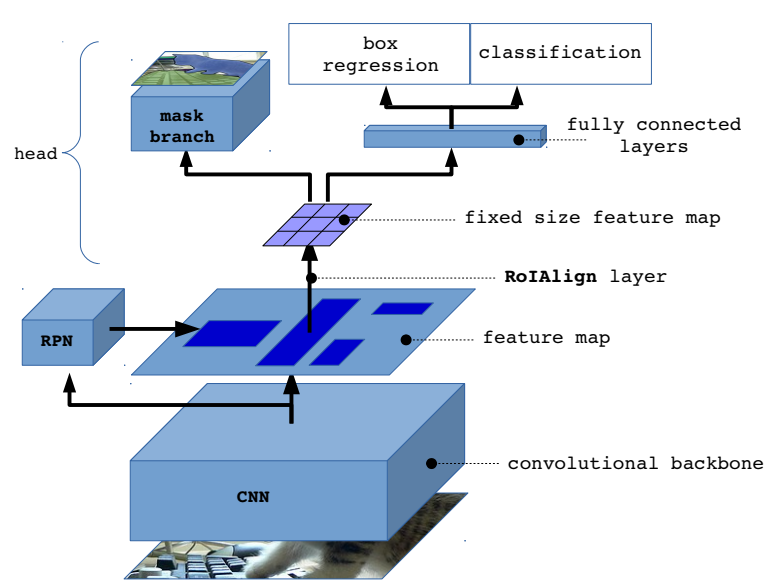
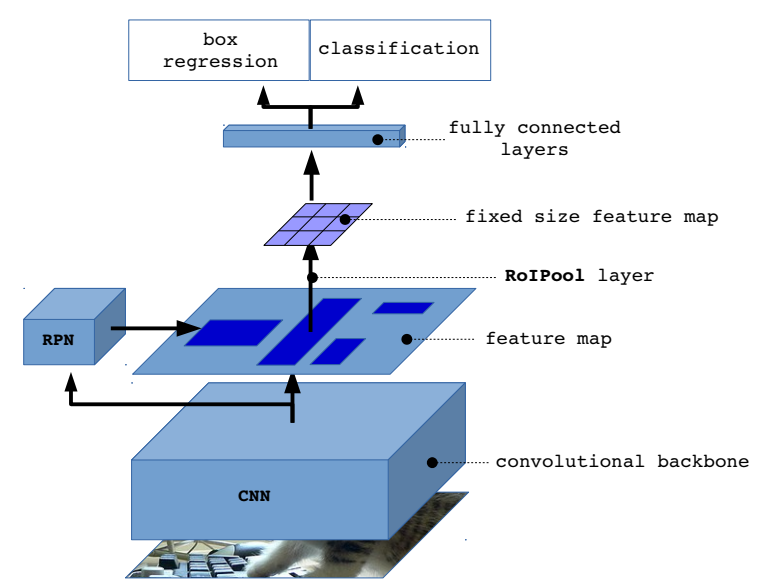


Fig.1. Faster R-CNN architecture Fig.2. Mask R-CNN architecture

**Constructia modelelor Faster R-CNN si Mask R-CNN:**

1. Basic Convolutional Neural Network (Feature extractor)

* De obicei ResNet50 sau ResNet101;
* Feature map-ul rezultat va fi input-ul urmatoarelor etape;

1. Region Proposal Network (RPN)

* Modalitatea prin care acel pas de Region Proposals care in modelele anterioare (R- CNN, Fast R-CNN) era realizat independent de retea, cu ajutorul algoritmilor de tip “Selective Search”, devine mult mai rapid si eficient (Region Proposals sunt extrase din feature map-ul de la pasul anterior). Deci, trebuie sa antrenam o singura retea CNN, care e folosita simultan si pentru a realiza procesul de Region Proposals, dar si pentru clasificare;
* RPN foloseste un sliding window peste feature map-ul rezultat din CNN si, la fiecare window, creeaza k posibilitati de bounding box cu aspect ratio si dimensiuni diferite (numite anchors); Pentru fiecare astfel de anchor, sunt generate doua output-uri: un bounding box si un scor care reprezinta cat de probabil este sa fie in interiorul box-ului un obiect (clasificare binara – Object/NotObject). Dupa acest pas, avem “final proposals”, numite regiuni de interes (ROI), pe care le vom transmite ca input pentru urmatoarea etapa;

1. ROI Pooling

* Input-ul unui clasificator este de obicei fix, insa in urma RPN-ului noi obtinem regiuni de interes de dimensiuni diferite;
* Pentru fiecare regiune de interes, se ia sectiunea corespunzatoare acesteia din feature map si se scaleaza (max-pool) la o dimensiune predefinita (de ex. 7x7).

1. Classifier and Bounding Box Regressor

* Pentru fiecare ROI astfel redimensionat, sunt generate doua output-uri:
  + Class: Clasificarea obiectului din acea regiune de interes (spre deosebire de RPN, care are doar doua clase, aceasta retea este mai deep si are capacitatea sa clasifice clase mai variate si mai specifice);
  + Bounding Box Refinement: Este asemanator cu cea facuta de catre RPN si are rolul de a rafina si mai mult bounding box-ul care incadreaza obiectul gasit;

Mask R-CNN este o extenstie pentru Faster R-CNN, in sensul ca pornind de la framework-ul prezentat anterior, autorii au adaugat urmatoarele modificari:

* Segmentarea instantei la nivel de pixeli (Semantic Segmentation): este realizata in paralel cu task-ul de clasificare si de bounding box;
* Roi Align: inclocuirea etapei de ROI Pooling, tocmai pentru ca atunci cand se realizeaza segmentarea la nivel de pixeli este nevoie sa se pastreze informatiile spatiale dintr-o imagine, realizandu-se astfel mult mai bine maparea ROI-urilor la regiunile din imaginea originala;

In cazul sistemului de cautare prezentat, am folosit framework-ul TensorFlow ObjectDetection API, si anume modelele Faster R-CNN si Mask R-CNN pre-antrenate pe setul de date MS-COCO, cu Resnet101 ca feature extractor.

**Cum indexam dataset-ul?**

Pentru fiecare obiect detectat in imagine, ne vom folosi de modelul Resnet50 pre-antrenat pe setul de date ImageNet, oferit de libraria Keras pentru a extrage caracteristici specifice. Resnet50 este o retea convolutionala cu 50 de straturi pentru clasificarea imaginilor. Pentru a obtine lista de caracteristici calculata de model, trebuie sa luam informatia de pe stratul anterior celui fully-connected folosit pentru clasificare. Astfel, obtinem un vector de caracteristici cu 2048 de valori, pe care il retinem asociat cu obiectul din imagine in baza de date.

**Care este metrica de similaritate folosita?**

Pentru a stabili cat de similare sunt doua imagini, am ales sa folosim “cosine similarity function” aplicata pe vectorii de valori specifici fiecarei imagini rezultati in urma operatiei de extragere de caracteristici.  Pentru doi vectori x si y, aceasta functie este definita astfel:, unde este, de fapt, valoarea cosinus-ului unghiului dintre vectorii si . Functia returneaza valori intre -1 si 1, iar cu cat valoarea este mai apropiata de 1, cu atat similaritatea este mai mare. Pentru a obtine rezultatele dorite, am stabilit un threshold de 0.57.

**Cum este realizata cautarea propriu-zisa?**

Utilizatorul incarca imaginea dupa care doreste sa se realizeze cautarea, iar sistemul va realiza, in primul rand, detectarea de obiecte. Pentru fiecare obiect gasit, se extrag caracteristici si se cauta pe baza functiei de similaritate stabilite mai sus si a clasei obiectului, imagini ce contin obiecte similare cu cel curent.

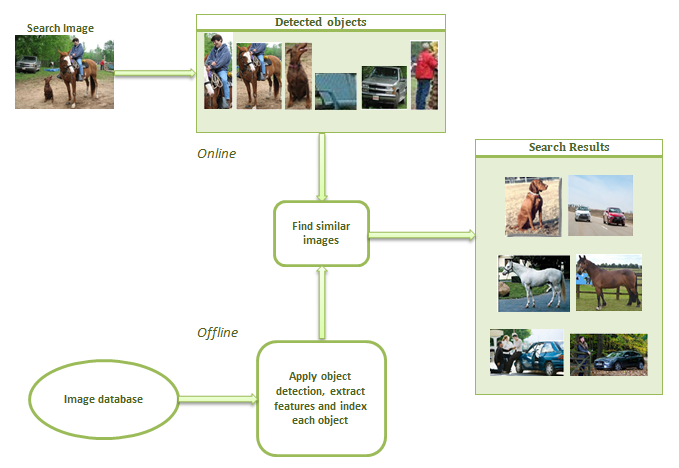


Fig.3. Search Engine schema

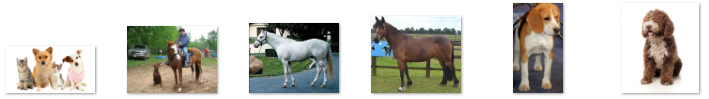
**Rezultate**

Prezentam in continuare un exemplu detaliat cu rezultatele intoarse pentru fiecare obiect detectat (atat imaginile, cat si similaritatea dintre acestea).

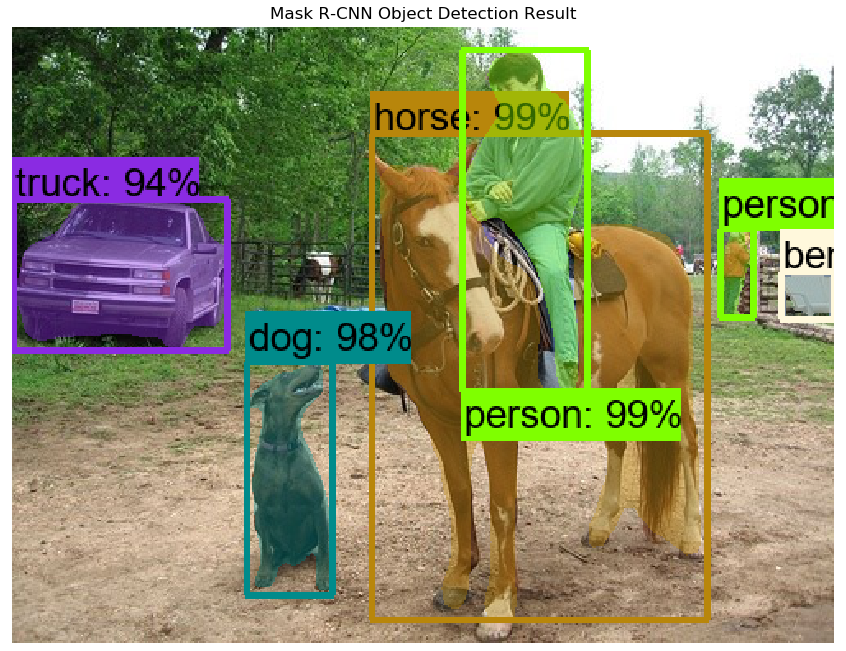
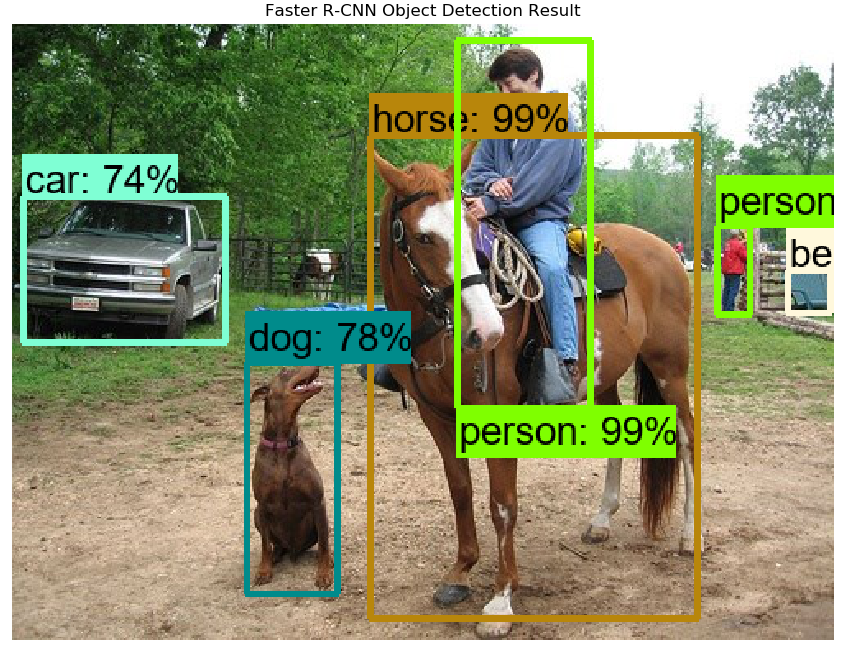
Imaginea dupa care dorim sa facem cautarea:



Baza de date cu imagini pentru cautare:



In urma aplicarii algoritmilor de object detection obtinem:



Pentru fiecare obiect astfel rezultat, sistemul intoarce urmatoarele rezultate:

|  |  |  |
| --- | --- | --- |
| Faster R-CNN | Mask R-CNN | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  | |  |
|  | |  |

**Concluzie**

Diferentele observate pe parcursul testarii sunt in legatura cu performanta, modelul Mask R-CNN consumand mai multe resurse, dar si mai mult timp. In urma rezultatelor, consideram ca de preferat ar fi utilizarea modelului Faster R-CNN, imbunatatirile aduse de Mask R-CNN fiind nesemnificative. Sigura diferenta majora dintre cele doua tehnici este la nivelul detectarii de obiecte, unde pot aparea diferente de clasificare (de exemplu, intr-un test efectuat, Faster R-CNN a detectat un obiect ca apartinand clasei “car”, iar Mask R-CNN a ales clasa “truck”, iar datorita faptului ca noi facem initial o comparare si dupa clasa fiecarui obiect, au aparut rezultate diferite). Asadar, folosindu-ne de modele pre-antrenate, am reusit sa construim un motor de cautare destul de interesant.

**Referinte**

[1] <https://towardsdatascience.com/deep-learning-for-object-detection-a-comprehensive-review-73930816d8d9>

[2] <https://github.com/tensorflow/models/tree/master/research/object_detection>

[3] <https://keras.io/applications/#resnet50>

[4] <https://labs.pinterest.com/user/themes/pinlabs/assets/paper/visual_search_at_pinterest.pdf>

[5] <https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md>