**Курсов проект по Обработка на изображения**

**Тема 5. Отделяне на обект от фон в цветни изображения**

Тервел Вълков, ФН 9MI3400058

**1. Описание на задачата**

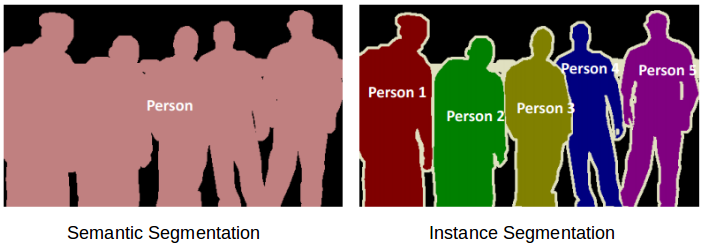
Отделянето на обект от фон е задача в областта на Image Segmentation – клон на обработката на изображения, който се фокусира върху разделянето на изображение на различни части според техните характеристики. Целта тук е да опростим или променим представянето на изображение в нещо по-лесно за анализиране, и следователно това често е първа стъпка при задачи изискващи по-дълбок анализ на дадено изображение.

Различните задачи в тази област включват разделяне на пикселите на изображението на групи (които ще наричаме обекти), присвояването на етикети на тези групи, класифициране на други пиксели според етикетите, намиране и посочване на граници между обектите, и отделянето на даден обект или обекти (които считаме за важни) от останалите – това е нашата конкретна задача в този случай. Като резултат ще получим маски на обектите които сме намерили, и чрез тях ще може от оригиналното изображение да извлечем частите (сегментите) които ни интересуват.

Тези техники намират приложение в много области - ползват се за откриване на обекти (лица, пешеходци, обекти в сателитни снимки, и др.), за разпознаване на обекти (лица, пръстови отпечатъци, ириси на очите при биометрична идентификация, и др.), при образната диагностика в медицината (за намиране на тумори, патологии, планиране и симулиране на хирургии, и др.), при търсене на изображения по съдържание, както и при видеонаблюдение, в системи за контрол на трафика, в автономни автомобили, и много други ситуации.

**2. Подходи към Image Segmentation**

За отделяне на обекти се използват както класически Computer Vision подходи, така и такива, базирани на изкуствен интелект. Има няколко различни по-общи класификации на методите:

- semantic и instance segmentation – при семантичната сегментация за всеки пиксел намираме класа (обект или фон), към който той принадлежи, докато при instance segmentation освен клас добавяме и конкретна инстанция, което ни позволява да отделим различните инстанции на един и същ клас.

- според подхода към идентифициране на обекти

- *region-based* (similarity detection) подходи, при които търсим подобни пиксели, ползвайки техники като thresholding, разделяне/сливане на региони, разрастване на региони, клъстеризация.

- *boundary-based* (discontinuity detection) подходи, при които обратно на region-based подходите търсим различаващи се пиксели, които ще формират границите между обекти. За тази цел ползваме техники като point detection, edge detection, line detection и др.

- според техниката – ползват се структурни техники (разчитащи на информация за структурата на търсения регион), стохастични техники (работещи директно със стойностите на пикселите на региона), както и хибридни техники (комбиниращи характеристики на другите два типа)

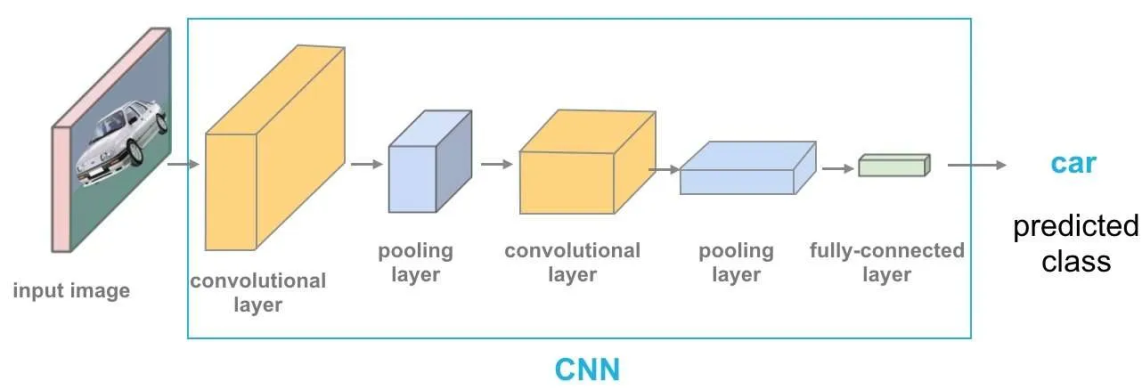
Някои конкретни подходи и алгоритми са:

1. **Thresholding** – пикселите се разделят според нивата им на интензитет, като се проверява дали тези нива са над или под дадена граница (*threshold*). Това е подходящ подход при изображения със светъл обект на тъмен фон, тъй като тогава лесно може чрез интензитета да се определи коя част от изображението е обекта. Има различни варианти – с една единствена глобална граница, с променлива граница (локална или адаптивна), с няколко различни граници. Стойност на една граница обикновено може да получим от хистограмите на изображението. Тези методи са най-простите, но съответно и най-ограничените.
2. **Edge-based segmentation** – тук се търсят места в изображението, където се наблюдава промяна на дадена характеристика (интензитет, текстура, цвят, наситеност, и др.), като така целим да намерим къде има ръбове. След като определим ръбовете, може да ги свържем и така да сформираме пълните граници на обектите. Отново има различни варианти, базиращи се на алгоритми като този на *Roberts*, на *Sobel*, на *Robinson*, *Canny* алгоритъмът, и др. Тези методи са неподходящи при наличие на много шум (тъй като той ще попречи на откриването на ръбовете) или когато има прекалено много ръбове в изображението.
3. **Region-based segmentation** – в този случай изображението се разделя на региони – множества от пиксели с подобни характеристики. Един начин за това е чрез *region growing*, при което се започва с начални пиксели (seeds), съответстващи на регионите и избрани ръчно или автоматично. Избираме даден регион и го сравняваме със съседните му точни, след което го разрастваме така че да включва подобните на него точки. Алтернативен метод е *region splitting и merging*, при който итеративно разделяме изображението на региони с подобни характеристики, и сливаме съседни региони, които са подобни. Този клас методи по-добре се справят с шума, но са по-скъпи откъм време и памет. Подходящи са когато е лесно да се дефинират критерии за подобие.
4. **Clustering-based** – при тези методи сегментираме изображението на клъстери от подобни пиксели. Клъстеризацията може да бъде *hard*, при която всеки пиксел принадлежи на точно един клъстер, или *soft*, при която се позволява пиксел да принадлежи на няколко клъстера, с различна степен на принадлежност към всеки. Основният алгоритъм тук е *k-means*, който има много различни варианти и подобрения (като *k-means++*, *fuzzy c-means*, и др.). Fuzzy клъстеризацията може да бъде особено полезна при реални проблеми, в които принадлежността не е толкова лесна за определяне и съответно е полезно да работим с вероятности вместо простия двоичен вариант, в който пиксел или принадлежи, или не.
5. **Watershed-based** – при този интересен вид методи ние гледаме на изображението като топографична карта, като обикновено използваме интензитета на всеки пиксел за да посочим неговата височина. Така, в топографичната интерпретация на изображението, локалните минимуми представляват долини. В самите алгоритми, ние започваме да „пълним“ долините с вода (с различен цвят за различни етикети), и когато води с различен цвят започват да се смесват, „строим“ бариери. Този процес продължава докато не потопим всичко, след което от построените бариери получаваме търсената сегментация. Тези техники се прилагат често при сегментирането в образната диагностика, но изчисленията в тях могат да бъдат доста сложни.
6. **Partial differential equation-based** – тук изображението се сегментира чрез решаване на частни диференциални уравнения, като се използват различни техники – разпространение на кривата, параметрични методи, level-set методът, fast marching методът, и др. Това са бързи методи, подходящи за time-critical приложения.
7. **Graph partitioning** - тези методи моделират изображението като претеглен неориентиран граф, в който възлите са индивидуални пиксели или групи от пиксели на изображението, ребрата свързват съседни пиксели/групи, а теглата им измерват подобието между тях. След построяването на графа, той бива разделен според даден критерий, при което получаваме търсените сегменти. Критерият трябва да е избран така, че да води до разделянето на добри (по някаква метрика) клъстери. Има много различни техники, като normalized cuts, random walker, minimum cut, isoperimetric partitioning, minimum spanning tree-based segmentation (Felzenszwalb’s Algorithm), segmentation-based object categorization, Markov random fields. Този клас методи са ефективни, и могат да бъдат паралелизирани.
8. **ANN-based (Artificial Neural Network)** – използват невронни мрежи. За разлика от повечето предишни методи, които разчитат само на информация за стойностите на пикселите, невронните мрежи може да бъдат обучени да имат допълнителни знания за областта, което прави ANN-базираните методи потенциално много мощни, но също налага допълнително изискване за първоначално трениране на модела, което отнема време. Те имат две основни стъпки – първо се извличат характеристиките на изображението, след което се извършва самото сегментиране от невронната мрежа. Конкретни при изображенията се използва един специален клас мрежи – Convolutional Neural Networks (CNNs), като някои конкретни такива типове са PCNNs, U-Net, Mask R-CNN, и др.
9. **Други методи** - compression-based, model-based, histogram-based, motion-based, variational methods, multi-scale segmentation, background subtraction methods, background-foreground segmentation (при видео или други последователности от изображения)

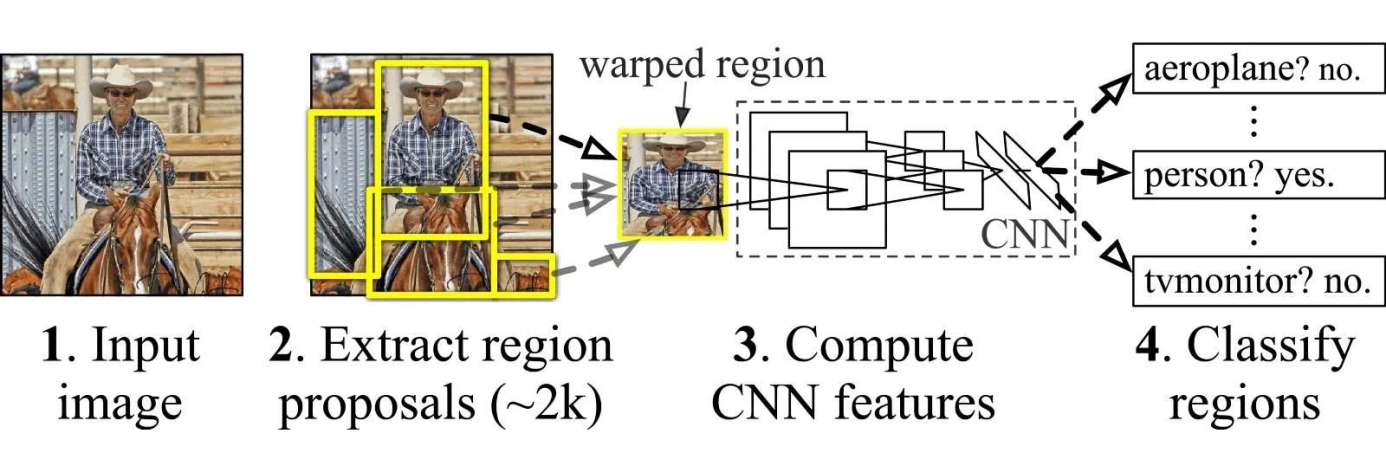
**3. Алгоритми, избрани за курсовия проект**

Специфично за курсовата работа е избран ANN-базиран метод - *Mask R-CNN*. Това е deep learning архитектура, създадена от експертите от Facebook AI Research, и представлява разширена версия на *Faster R-CNN* – архитектура за откриване на обекти, която намира клас и очертаващ правоъгълник за всеки обект в даден изображение. Само правоъгълник не е достатъчен за целите на сегментацията, затова *Mask R-CNN* допълва резултатите на *Faster R-CNN* с пикселна маска за всеки обект, чрез която може да определим точно кои пиксели принадлежат на обекта, и така да изрежем само частта от изображението която го съдържа.

Тези мрежи са *конволюционни невронни мрежи* (CNNs) – специален вид невронни мрежи, оптимизирани за обработката на пикселни данни. Този клас невронни мрежи имат 3 основни слоя – първо има *convolutional* слой, който абстрахира входното изображение като карта на характеристики (*feature map*), чрез филтри и ядра, след което тази карта се предава на *pooling* слоя, който извършва редуциране (*downsampling*) на картата чрез обобщаване на присъстващи характеристики в *patch*-ове. Накрая има *fully connected* слой, който свързва всеки неврон в един слой с всеки неврон от друг. На практика, дадена мрежа може да има няколко convolutional и pooling слоеве, и комбинирането им позволява на мрежата да се научи как да идентифицира и разпознава търсения обект.

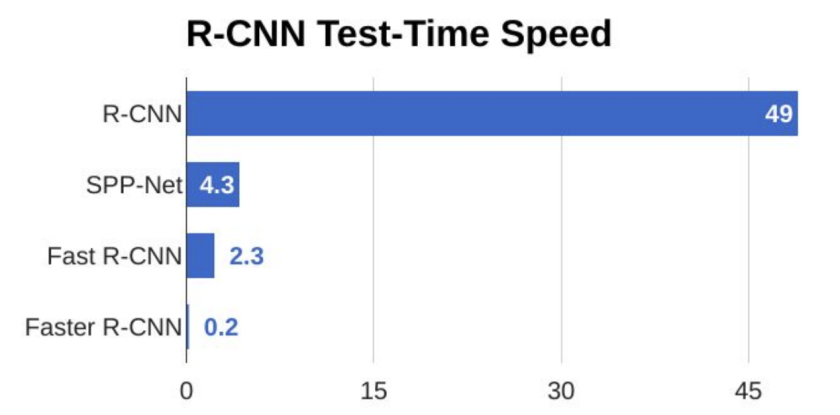


Простите CNNs са създадени за класификация на изображения и откриване на обекти, в случаи когато има само по един обект в изображението. Те не са подходящи за по-сложни ситуации с повече обекти, поради което е разработена *R-CNN* (Region-based CNN). Тя намира кандидат-региони (region proposals), които може да съдържат обект, използва CNN за да извлече характеристики от тях, и според тези характеристики използва SVM за да класифицира присъствието на обекти в кандидат-регионите.

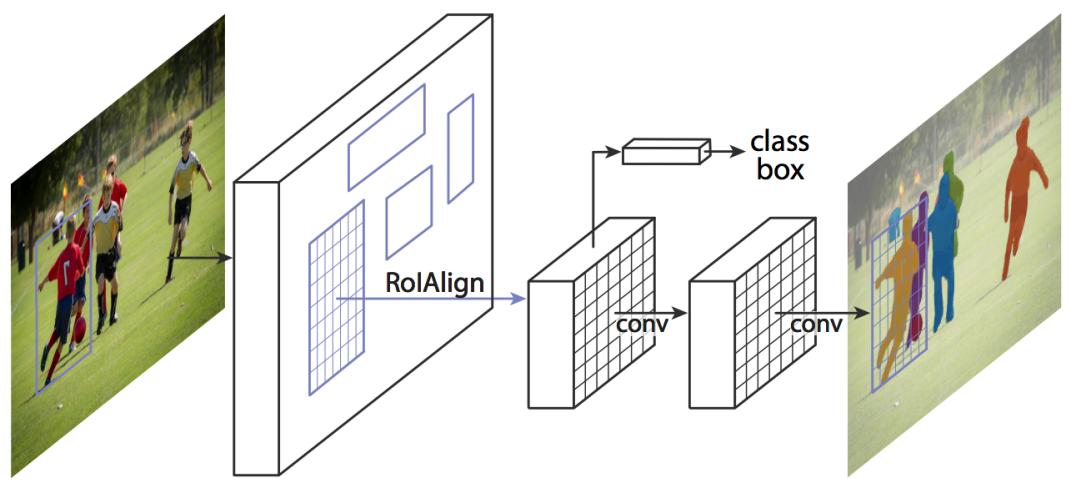


За съжаление, поради стъпката с генерирането на региони, R-CNN се оказва прекалено бавна. Този проблем води до разработването на *Fast R-CNN*, която се опитва да ускори процеса като първо подава изображението на CNN за да получи feature map, след което чрез map-а идентифицира кандидат-регионите, изкривява ги до квадрати и чрез RoI pooling слой ги преоформя до фиксиран размер и ги подава на fully-connected слоя. Накрая, от вектора с RoI характеристики се предсказва класа на кандидат-регион чрез softmax слой.

*Fast R-CNN* е значително по-бърза от обикновената *R-CNN*, но кандидат-регионите все пак водят до забавяне, поради което е разработена и следващо подобрение – *Faster R-CNN*, която отново първо подава изображението на CNN, но вече вместо да търси кандидат-региони, ги предсказва с отделна невронна мрежа (*Region Proposal Network*), и с вече предсказаните региони борави по подобен начин като *Fast R-CNN*. Така се постига още по-добра скорост.



Всичките тези мрежи са предназначени за откриване на обекти, и за тази цел намират само класове и очертаващи правоъгълници за различните обекти в изображението. Това е достатъчно за да установим присъствието на обекта и да получим преценка за разположението му, но не и за да можем да го отделим от остатъка на изображението. Затова *Mask R-CNN* надгражда *Faster R-CNN*, като към резултата добавя и маска. Допълнително изчисление на маските всъщност добавя само малки изчислителни разходи, така че методът продължава да бъде бърз.



Освен самия *Mask R-CNN*, към проекта е добавена и възможността за прилагането на допълнителен алгоритъм – *GrabCut*. Той е базиран на графи и ползва Markov random fields (специални графи, дефиниращи вероятностни разпределения), като започва с определен от потребителя очертаващ правоъгълник около обектът който трябва да се сегментира и пресмята приближение на разпределението на цветове на обекта и на фона (чрез Gaussian mixture model). След това построява Markov random field върху етикетите на пикселите (обект и фон) и прилага оптимизация, базирана на графови изрязвания, за да получи финалната сегментация.

Конкретно в проекта, *GrabCut* не се ползва сам по себе си, а се прилага върху резултатите на *Mask R-CNN* – идеята е, че невронната мрежа може да произвежда маски които не са чисто изрязани (включват части от фона по ръбовете), и следователно може да опитаме да подобрим резултата като приложим *GrabCut* върху него за допълнително изрязване.

**4. Имплементация**

Проектът е написан на *Python*.

Има няколко различни имплементации на *Mask R-CNN* – в проекта е ползвана имплементацията на [Matterport](https://github.com/matterport/Mask_RCNN), разпространявана по MIT лиценза, но има и алтернативни опции, като [Mask R-CNN Inception V2](https://docs.openvino.ai/latest/omz_models_model_mask_rcnn_inception_v2_coco.html), [Mask R-CNN ResNet101 Atrous](https://docs.openvino.ai/latest/omz_models_model_mask_rcnn_resnet101_atrous_coco.html), и други. Освен това, *Mask R-CNN* е една от частите на системата [Detectron](https://github.com/facebookresearch/Detectron) на Facebook AI Research.

Повечето имплементации са тренирани върху dataset-а [MS COCO](https://cocodataset.org/#home), съдържащ над 330 хиляди изображения. Matterport предоставят предварително трениран (върху COCO) модел, но мрежата им също така има и възможности за трениране върху други данни, или за допълнително трениране на съществуващия модел. За проекта е използван само обикновения трениран модел.

Имплементацията на Matterport изисква *Keras* и *TensorFlow*, но за съжаление не работи с най-новите им версии. Изглежда има fork-ове които се опитват да я обновят до новите версии на библиотеките, като <https://github.com/ahmedfgad/Mask-RCNN-TF2>, но за целите на проекта просто използвах по-стари версии на библиотеките – *Keras* 2.2.4 и *TensorFlow* 1.5.0.

За алгоритъма *GrabCut* е ползвана имплементацията му в *OpenCV*. От библиотеката *OpenCV* също така се ползват функциите за прочитане и записване на изображения, както и за конвертиране между цветови пространства.

За тестване са ползвани снимки на животни от *The Oxford-IIIT Pet Dataset*, разпространявана по лиценза *Creative Commons Attribution-ShareAlike 4.0 International License*. Целият dataset съдържа над 7000 изображения, но за целите на тестване е избрано само едно малко подмножество от тях, от 62 изображения.

Конкретно в кода на проекта, имаме два класа. В *Mask-RCNN* има клас *Config*, предназначен за настройване на модела чрез наследяването му и променянето на нужните член-данни. В този случай създаваме клас *ProjectConfig*, в който единствено указваме, че моделът ще работи с единствено GPU, че ще обработва само по едно изображение (тъй като по-големи batch-ове от изображения са предназначени за по-мощни процесори), и че ще борави с 81 класа (80-те класа на обучителните данни от COCO, плюс едно за фона).

Другият клас *Model* ще съдържа като член-данни самият *MaskRCNN* модел, както и някои допълнителни данни, като path-ове до някои директории, и имената на класовете в COCO (които се изискват от функцията за визуализация на резултата). При инициализацията се създава обект от класа MaskRCNN (който ще съдържа самия модел) в режим на inference (тоест с него само ще предсказваме, няма да го тренираме), с конфигурация зададена от *ProjectConfig*, и в него се зареждат теглата на мрежата от данните за предварително тренирания модел, предоставени от Matterport.

Класът съдържа няколко функции - *open\_image* просто прочита едно изображение със съответните OpenCV функции и го връща като резултат, а *infer* приема едно такова изображение и просто извиква функцията *detect* на модела и връща резултата от нея. Той представлява списък от асоциативни списъци, по един за всяко изображение, но тъй като ние обработваме само по едно изображение, списъкът ще съдържа само един елемент, който е всъщност резултата върнат от *infer*. Асоциативният списък съдържа очертаващи правоъгълници, маски и класове (плюс вероятностите им, указващи доколко сигурен е модела в класификацията си) за всеки открит обект. С този резултат по принцип може да извикаме Mask-RCNN функцията *display\_instances*, която показва оригиналното изображение, с правоъгълниците и маските изобразени отгоре му, но тук ползваме модифицирана нейна версия *save\_instances* (модификацията е взета от [issue](https://github.com/matterport/Mask_RCNN/issues/1282#issuecomment-508756219) в github repository-то), която записва това изображение на диска вместо да го показва. Това се случва в член-функцията на Model *save\_visualization.* [Детайлна демонстрация](https://github.com/matterport/Mask_RCNN/blob/master/samples/coco/inspect_model.ipynb) на процеса на работа на модела е предложена в github repository-то на Matterport.

В *save\_segments* използваме маските за да извършим финалната стъпка на сегментацията – генерирането на самите сегменти. Маските върнати от модела са булеви, но ние може лесно да ги конвертираме към числови такива (от 0 и 1), и да ги приложим към изображението – пикселите съответстващи на нули в маската се броят към фона и ще останат без цвят, а пикселите съответстващи на единици се броят към обекта и ще си останат непроменени. Тъй като може да имаме няколко маски (ако моделът е открил няколко обекта), то повтаряме това за всяка отделна маска. Действието на функцията е различно според това дали искаме да използваме GrabCut или не (което подаваме като команден аргумент) – ако не, то сегментите директно се генерират, но ако да, то трябва да извикаме cv2.grabCut. Тя има два режима на работа – инициализация чрез очертаващи правоъгълници (bounding boxes) и инициализация чрез маски. В случая, тъй като получаваме готови маски от мрежата, се ползва втория вариант, и като резултат се получава нова маска, която след това прилагаме по същия начин както правим и в първия случай.

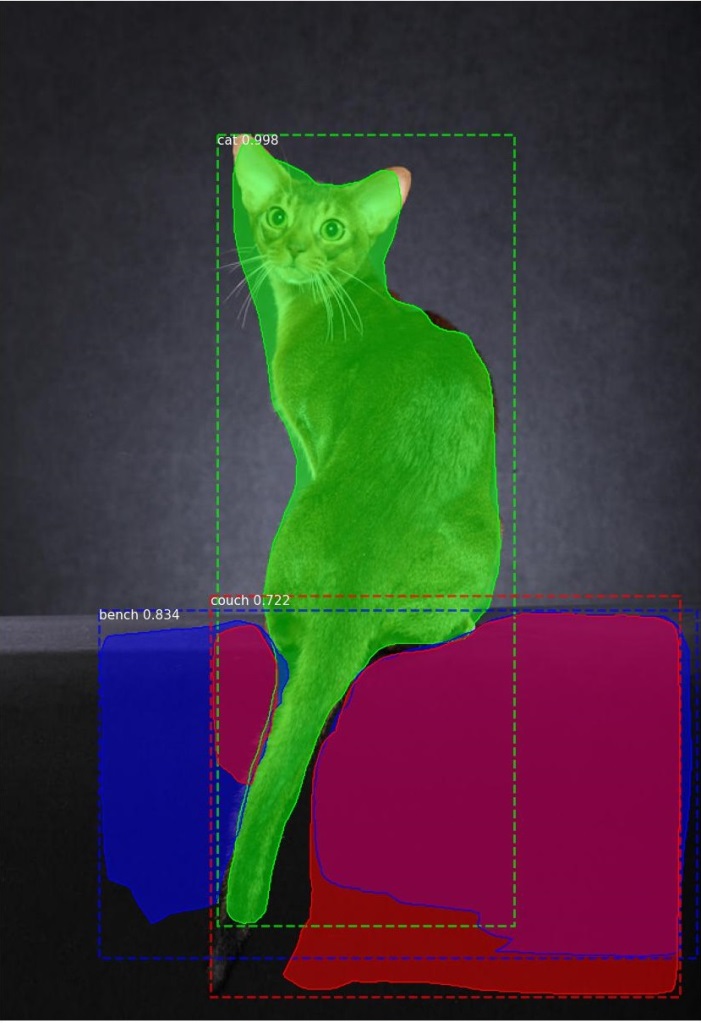
Целият процес на обработване на изображение – прочитането му, подаването му на модела, и записването на резултатите (визуализация на маските, плюс самите сегменти) става чрез функцията *process\_image*. Има и допълнителни функции за обработване на цялата папка от изображения или някакви части от нея, но те са ползвани само при тестването за по-удобно генериране на резултати от много изображения.

Крайната програма пита за имена на изображения от командния ред, и продължава да пита докато не бъде въведено “EXIT”, тъй като първоначалната инициализация на модела е доста бавна, така че е по-удобно той да се инициализира веднъж и след това да му се подават последователно изображения, вместо всеки път програмата да се пуска с различно изображение като параметър.

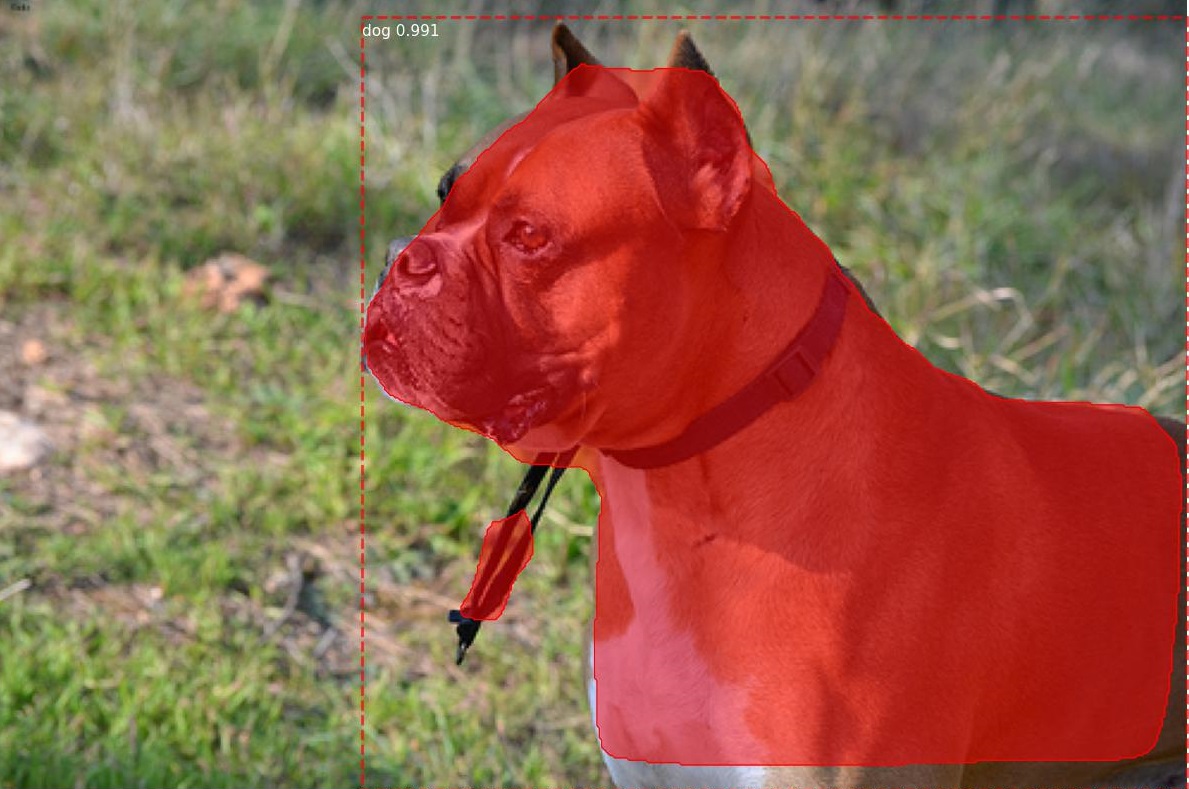
**5. Резултати**

Маски бяха генерирани за всичките 62 тестови изображения, сегменти по маските на *Mask-RCNN* за почти всички (60), а сегменти по допълнително-обработените маски от GrabCut за 18 от изображенията. В много от случаите маските са доста добри, но като цяло имат проблем около ръбовете – често или включват малка част от фона, или изпускат малка част от обекта.

Допълнителното прилагане на *GrabCut* в някои случаи наистина подобрява резултата, но в повечето го влошава, и то доста. Резултатите на GrabCut зависят от маската с която е инициализиран, като идеята зад прилагането му беше да подобрим маските в случаи когато те включват излишни части от фона, като с него ние се опитваме да изрежем тези части. Доста от маските на Mask-RCNN обаче всъщност изпускат части от самите обекти – така че е логично че при допълнително изрязване ще изгубим още повече от обекта, и така ще влошим резултата. Така че комбинирането на двата алгоритъма със сигурност не винаги е уместно, но все пак има някои случаи в които може да има позитивен ефект.















**Източници:**

1. Image Segmentation – Wikipedia

[<https://en.wikipedia.org/wiki/Image_segmentation>]

2. Image Segmentation Techniques – Pavan Vadapalli, upgrad.com

[<https://www.upgrad.com/blog/image-segmentation-techniques>]

3. Computer Vision Tutorial: A Step-by-Step Introduction to Image Segmentation Techniques – Pulkit Sharma, analyticsvidhya.com

[<https://www.analyticsvidhya.com/blog/2019/04/introduction-image-segmentation-techniques-python/>]

4. Image Segmentation with Watershed Algorithm – OpenCV documentation

[<https://docs.opencv.org/4.x/d3/db4/tutorial_py_watershed.html>]

5. Various Image Segmentation Techniques: A Review – Dilpreet Kaur & Yadwinder Kaur, International Journal of Computer Science and Mobile Computing (IJCSMC, Vol. 3, Issue. 5, May 2014, pg.809 – 814)

[<https://ijcsmc.com/docs/papers/May2014/V3I5201499a84.pdf>]

6. Image Segmentation with Classical Computer Vision-Based Approaches – Yağmur Çiğdem Aktaş, Towards Data Science

[<https://towardsdatascience.com/image-segmentation-with-classical-computer-vision-based-approaches-80c75d6d995f>]

7. Mask R-CNN: A Beginner’s Guide – Elisha Odemakinde, viso.ai

[<https://viso.ai/deep-learning/mask-r-cnn/>]

8. Getting Started with R-CNN, Fast R-CNN, and Faster R-CNN – MathWorks

[<https://www.mathworks.com/help/vision/ug/getting-started-with-r-cnn-fast-r-cnn-and-faster-r-cnn.html>]

9. R-CNN, Fast R-CNN, Faster R-CNN, YOLO – Object Detection Algorithms

- Rohith Gandhi , Towards Data Science

[<https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e>]

10. Mask R-CNN for Object Detection and Segmentation – Matterport Inc.

[<https://github.com/matterport/Mask_RCNN>]

11. Computer Vision Tutorial: Implementing Mask R-CNN for Image Segmentation (with Python Code) – Pulkit Sharma, analyticsvidhya.com

[<https://www.analyticsvidhya.com/blog/2019/07/computer-vision-implementing-mask-r-cnn-image-segmentation/>]

12. How to Use Mask R-CNN in Keras for Object Detection in Photographs – Jason Brownlee, Machine Learning Mastery

[<https://machinelearningmastery.com/how-to-perform-object-detection-in-photographs-with-mask-r-cnn-in-keras/>]

13. Interactive Foreground Extraction using GrabCut Algorithm, OpenCV-Python Tutorials

[<https://docs.opencv.org/3.4/d8/d83/tutorial_py_grabcut.html>]

14. OpenCV GrabCut: Foreground Segmentation and Extraction – Adrian Rosebrock, PyImageSearch

[<https://www.pyimagesearch.com/2020/07/27/opencv-grabcut-foreground-segmentation-and-extraction/>]

15. Image Segmentation with Mask R-CNN, GrabCut, and OpenCV – Adrian Rosebrock, PyImageSearch

[<https://www.pyimagesearch.com/2020/09/28/image-segmentation-with-mask-r-cnn-grabcut-and-opencv/>]