

Object Detection in Fine Art Photography

Bachelor thesis, spring semester, 2020

Fabian Meyer

Supervisor: Dr. Simone Lionetti

Lucerne University of Applied Sciences and Arts
Bachelor in Information Technology
May 30, 2020

Bachelorarbeit an der Hochschule Luzern - Informatik

Titel: Object Detection in Fine Art Photography

Student: Fabian Meyer

Studiengang: Bachelor Informatik

Abschlussjahr: 2020

Betreuungsperson: Dr. Simone Lionetti

Expertin / Experte: Roman Bachmann, Swisscom

Eidesstattliche Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig und ohne unerlaubte fremde Hilfe angefertigt habe, alle verwendeten Quellen, Literatur und andere Hilfsmittel angegeben haben, wörtlich oder inhaltlich entnommene Stellen als solche kenntlich gemacht haben, das Vertraulichkeitsinteresse des Auftraggebers wahre und die Urheberrechtsbestimmungen der Fachhochschule Zentralschweiz (siehe Markblatt "Studentische Arbeiten" auf MyCampus) respektieren werde.

Ort / Datum, Unterschrift: _____

Abgabe der Arbeit auf der Portfolio Datenbank

Bestätigungsvisum Studentin / Student

Ich bestätige, dass ich die Bachelorarbeit korrekt gemäss Merkblatt auf der Portfolio Datenbank abgelegt habe. Die Verantwortlichkeit sowie die Berechtigungen habe ich abgegeben, so dass ich keine Änderungen mehr vornehmen kann oder weitere Dateien hochladen kann.

Ort / Datum, Unterschrift: _____

Verdankung

Danke liebe Freunde

Eingangsvisum (durch das Sekretariat auszufüllen):

Rotkreuz, den _____ Visum: _____

Hinweis: Die Bachelorarbeit wurde von keinem Dozierenden nachbearbeitet. Veröffentlichungen (auch auszugsweise) sind ohne das Einverständnis der Studiengangleitung der Hochschule Luzern Informatik nicht erlaubt.

Copyright © 2019 Hochschule Luzern - Informatik

Alle Rechte vorbehalten. Kein Teil dieser Arbeit darf ohne die schriftliche Genehmigung der Studiengangleitung der Hochschule Luzern - Informatik in irgendeiner Form reproduziert oder in eine von Maschinen verwendete Sprache übertragen werden.

Abstract

Contents

1	Introduction	1
1.1	Task description	1
1.2	Why fine art photography?	1
2	Current state	3
2.1	About convolutional neural networks	3
2.2	From CNNs to Mask R-CNN	3
3	Methode	6
3.0.1	quantitative/qualitative Querschnittsanalyse	6
3.0.2	Fallstudie	6
3.0.3	Labor- / Feldexperiment	6
3.1	Projektinformationen	6
3.1.1	Vorgehensmodell	6
3.1.2	Ermittlung offener Projektrahmenbedingungen	6
3.1.3	Projektanforderungen / Anforderungsanalyse	6
3.1.4	Einschränkungen und Abgrenzungen	6
3.1.5	Systemarchitektur	6
3.1.6	Komponentendesign	6
3.1.7	Umsetzung / Programmierung	6
3.1.8	Testing	6
4	Ideen und Konzepte	7
4.1	Grundidee	7
5	Realisation	8
5.1	Data collection	8
5.2	Model selection	9
5.3	Framework selection	9
5.4	Choosing the programming language	9
5.5	Creating the tidied up image	9
5.6	Building the application	11
5.7	Deployment	12
5.8	Refining the model	12
5.9	Results	12

6	Evaluation und Validation	13
6.1	Vergleich mit Anforderungen	13
6.2	Technische Aspekte	13
6.3	Vorgehen	13

1 Introduction

Object detection is an important computer vision and machine learning task that consists in the localisation and classification of items within digital images. The business and industry applications of this technology are growing in number and relevance, especially because of the tremendous progress, largely driven by deep learning, that has been made in recent years.

Uses of object detection technologies include applications in security measures for example images from surveillance cameras, search engines, object tracking or counting in traffic, autonomous driving cars, robotics in general and the field of medical- and bioinformatics.

1.1 Task description

The aim of this work is to develop a web-application that is built around an object-detection model. This web-application takes a picture, looks for objects in it and returns a "tidied up" image with the objects found in it.

The "tidied up" image is inspired by "Kunst aufräumen", a series of fine-art photographs by Swiss artist Urs Wehrli, that consists of two images, a "messy" one and a "tidy" one.

In a further step, an existing model was refined by retraining it on a dataset with images from "Kunst aufräumen" and the performance of the two models were examined

1.2 Why fine art photography?

Models for object detection are usually trained on datasets, containing pictures that show objects in a very clear manner. These images usually contain only a few objects and are shot in natural lighting with natural colours. Composition-wise they are simple as it is the goal to depict the object in most clear way.



Figure 1.1: Sample image from COCO dataset

Fine art photography pictures on the other hand is in strong contrast to these images, as they are depicting objects who do not belong together usually. They are often shot in studios with artificial lighting and heavy post-processing or even digital manipulation. And they often contain a lot more objects in it which even can overlap each other.



Figure 1.2: Sample image from artist David LaChapelle

The question that now arises is, how will object detection models that are trained on traditional datasets perform on given images of artistic photography?

2 Current state

2.1 About convolutional neural networks

The model that is used for this project is from the family of convolutional neural networks or CNNs. These are models that are especially useful to perform operations on digital images. CNNs contain at least one convolutional layer that computes a mathematical operation that is called discrete convolution. Discrete convolution operation has been proved useful when applied on digital images to detect structural features like edges and corners.

The advantage of CNNs compared to traditional multi layer perceptrons (MLPs) is that discrete convolutional operation is translational invariant. This means it is irrelevant where in a given image a specific object is located: The model will not learn the position of the specific object. Another huge advantage of CNNs over MLPs is that CNNs usually have sparse interactions. This means that it is possible to yield a good performance even if some adjacent layers are not fully connected with each other. Another way to speed up the training process is to use parameter sharing: Some weights will be used at multiple places at the same time. These two techniques, sparse interactions and parameter sharing does make it possible to train very deep neural networks in a feasible time.

When applying discrete convolution to an image, a filter with a given kernel is used to perform convolution on the image. To detect edges and corners there exist specific kernels especially for that kind of purpose. But often it is also possible to learn the optimal kernel in the process of machine learning.

Another important kind of layer that is usually employed in CNNs are pooling layers. Also called subsampling or downsampling layers sometimes, they reduce the size of a given image. Some frequently used pooling layers are max- and average-pooling. The advantages of pooling layers are smaller input, parameter reduction and more invariance to scaling and transformations.

The third important kind of layer is ReLU (rectified linear unit). The mere use of this layer is to preprocess the image before the next convolutional operation. This just adjusts the image brightness in a way that all negative values (values that are darker than the middle grey of an image) got adjusted to middle grey.

2.2 From CNNs to Mask R-CNN

The very first CNN appeared in the year 1994, it was named LeNet5, by Yann LeCun. This fundamental work was the first network that used convolutional and pooling layers to process images. In a time long before consumer graphic processing units (GPUs), where even

CPUs were slow, it was important to reduce the number of parameters to a bare minimum (accomplished with sparse connect layers).

Starting with AlexNet by Alex Krizhevsky, Ilya Sutskever and Geoffrey Hinton in 2012, CNNs gained significantly performance gains when applied on image classification tasks. AlexNet took the idea of a CNN from LeNet5 and added more layers to it and was the first to include ReLU layers. It also used dropout techniques to avoid overfitting during the training process. AlexNet was still an image detection model, trained to label images. This means a single label (class) is given to a whole input image.

ResNet (residual neural network) in 2015 was the first CNN that included residual blocks or identity shortcut connections. This are connections in the network that skip one or more layers and just use the input value as an output (mathematical identity). This was another step to reduce computation cost and lead to even deeper networks. ResNet is still used today as part of the backbone in a lot of object detection and instance segmentation models.

With R-CNN (regions with CNN features) in 2013, the rise of object detection models began. These researcher asked themself, how can one use the technique from CNNs to not only classify an image but to classify objects in that image? These family of models are capable of labelling multiple objects depicted in the same image with each a bounding box and a class prediction. R-CNN models do that by proposing regions in that potential objects may lie. R-CNN is thus called a two stage model: The first stage scans the image and generates region proposals, whereas the second stage uses these regions to extract CNN features from it and to ultimately classify objects in it. For the classification step in the last layer, R-CNN uses a support vector machine (SVM) as a classifier. In the very last step, a regression is used to further tighten the coordinates of the bounding boxes of each object. [1]

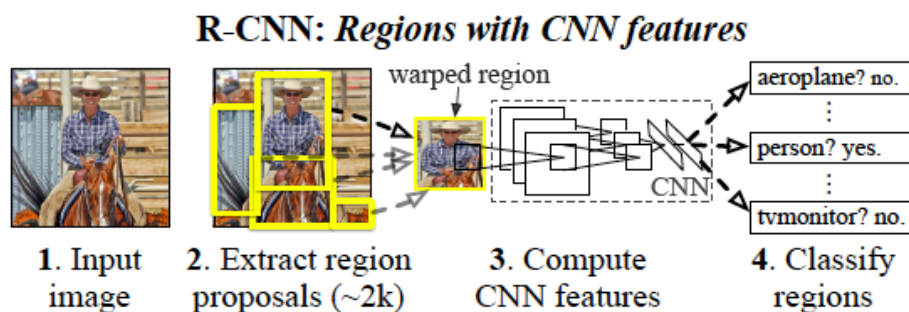


Figure 2.1: Architecture of R-CNN

In 2017, the team from Ross Girshick, one of the creators of R-CNN, delivered Fast R-CNN an improved, faster version of it. In R-CNN there are a lot of overlapping region proposals and every computation gets calculated again, even if the regions are very similar to each other. To circumvent this he invented region of interest pooling (RoIPool). RoIPool shares these computation across the regions of an image and can speed up computation a lot. The other improvement was to put all computation in a single network (compared to R-CNN where for example classification ran in a single network).

Also in 2017, also by the team from Ross Girshik, the second iteration of R-CNN got released, called Faster R-CNN. The main improvement was to use just one CNN that produces a feature map for both region proposal and classification.

With Mask R-CNN from 2017 (also by Ross Girshik et. al) it is also possible to not only predict the bounding box of an object but also to predict a mask that shows the exact shape of the object (called pixel level segmentation). These models are called instance segmentation models. This is accomplished by adding a branch to the model that computes a binary mask for a given object in the image. Mask R-CNN model is of the family of object or instance segmentation models.

An overview over the different tasks and approaches and their models in the field of object recognition can be seen here:

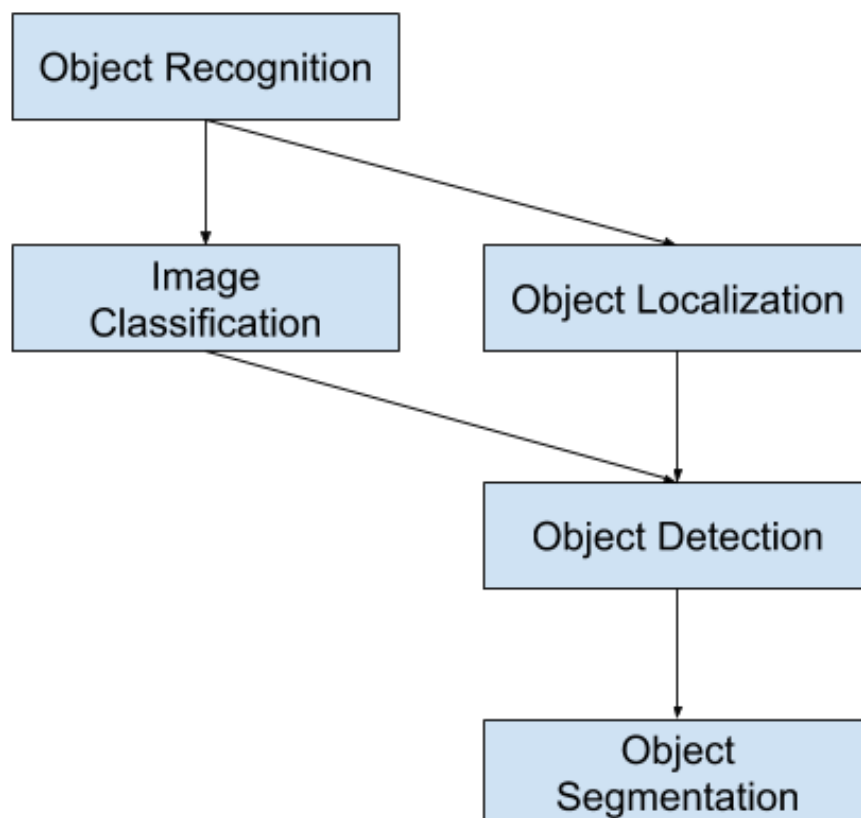


Figure 2.2: Tasks and approaches in object recognition

3 Methode

3.0.1 quantitative/qualitative Querschnittsanalyse

3.0.2 Fallstudie

3.0.3 Labor- / Feldexperiment

3.1 Projektinformationen

3.1.1 Vorgehensmodell

3.1.2 Ermittlung offener Projektrahmenbedingungen

3.1.3 Projektanforderungen / Anforderungsanalyse

3.1.4 Einschränkungen und Abgrenzungen

3.1.5 Systemarchitektur

3.1.6 Komponentendesign

3.1.7 Umsetzung / Programmierung

3.1.8 Testing

4 Ideen und Konzepte

4.1 Grundidee

5 Realisation

5.1 Data collection

The first step in this project was to collect some data. Two pictures of fine art photography were already shown in the project description: An image by german photographer Andreas Gursky and one by Swiss artist Urs Wehrli. The main motivation behind data collection was to obtain images from different artists that each show a lot of different objects. This let us examine the performance of an object detection model applied to fine art photography, by using images that contain a lot of objects that optimally are also included in the COCO dataset.

The COCO (Common Objects in COntext) dataset is among the most popular image datasets created for object recognition tasks. It was lanced by Microsoft in 2014 and the most recent version was published in 2017. It uses 80 different classes for object detection tasks that can be seen here:



Figure 5.1: All 80 classes from COCO dataset 2017

The COCO dataset not only contains bounding boxes and binary masks for object detection and segmentation, it also includes informations for keypoint estimation, panoptic segmentation and image captioning tasks.

In the end, 42 different images from the four artists Urs Wehrli, Andreas Gursky, David LaChapelle and Jeff Wall has been gathered. Urs Wehrli, the swiss artist behind "Kunst aufräumen" was asked to give his permission to use his images for this project. Thankfully he gave permission to do so. The dataset has then been examined with different object segmentation models on Google Colab.

Google Colab is a free to use infrastructure, powered by Google, that offers a zero-configuration Python and Jupyter environment. Running on Linux Ubuntu with the latest Nvidia GPUs this is a good option to get started with deep learning, as there are also a lot of notebooks about every single kind of neural network tasks available.

5.2 Model selection

With the gathered dataset, different object segmentation models from different frameworks have been tried out on the dataset. The tried out models were: Mask R-CNN, CenterNet, Detectron2 and ShapeMask. In general, Mask R-CNN outperformed the other models, when inference is run on the dataset. Detectron2 also delivered a good performance but it threw an error when running on many images in a loop on Google Colab.

5.3 Framework selection

At the same time different frameworks have been tested out. These were Tensorflow from Google, Detectron from Facebook and MMDetection, which is a part of the OpenMMLab project developed by Multimedia Laboratory by the Chinese University of Hong Kong. All these frameworks do at least contain one Mask R-CNN model. MMDetection has been chosen as the framework to develop the project in, because it worked out-of-the box when tried out on Google Colab. MMDetection has a vast number of state-of-the-art models for computer vision tasks available and offers a high-level API. It is built on top of PyTorch (primarily developed by Facebook) and delivers a good performance.

5.4 Choosing the programming language

Because most of the deep-learning frameworks are using Python and of its ease of use, Python has been used as the sole programming language to develop this project. Python does also offer a lot of visual computing libraries that were used to create the tidied up image.

5.5 Creating the tidied up image

After the dataset has been collected, and the model and the framework has been chosen, the next step was to extract all single objects from the output of the model when running in inference mode on an image. Mask R-CNN model outputs for every input image a list with with each a list of bounding boxes, masks, confidence score and predicted class of all objects found in the image. This is per default limited to 100 objects per image. The confidence score was used to filter all objects to get the objects with the highest quality as predicted by the model. The binary mask of the found objects was then applied to the image, creating binary images of the object. An example can be seen here:

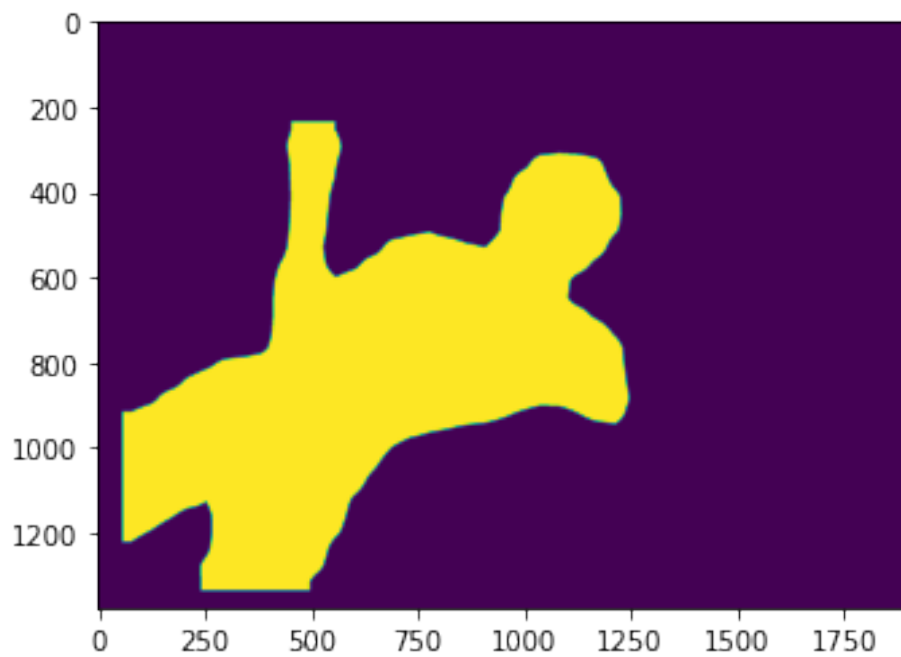


Figure 5.2: Sample binary mask image

With help of the binary masks, the background of the found object was coloured white whereas the foreground stayed as is. Subsequently the corresponding bounding boxes were taken to cut out the objects from the image. The last step is to create a grid with all the found objects. This was done with the help of Python's Matplotlib. The idea is to create a grid with the number of rows according to the number of classes and insert every object as a new column. The problem with using Matplotlib's `plt.subplots` is that it creates a grid where every image has the same size. To circumvent this, a more complicated approach was taken, by creating a grid manually with the help of the width and the height of all the found objects. A sample input of a tidied up image together with its output can be seen here:

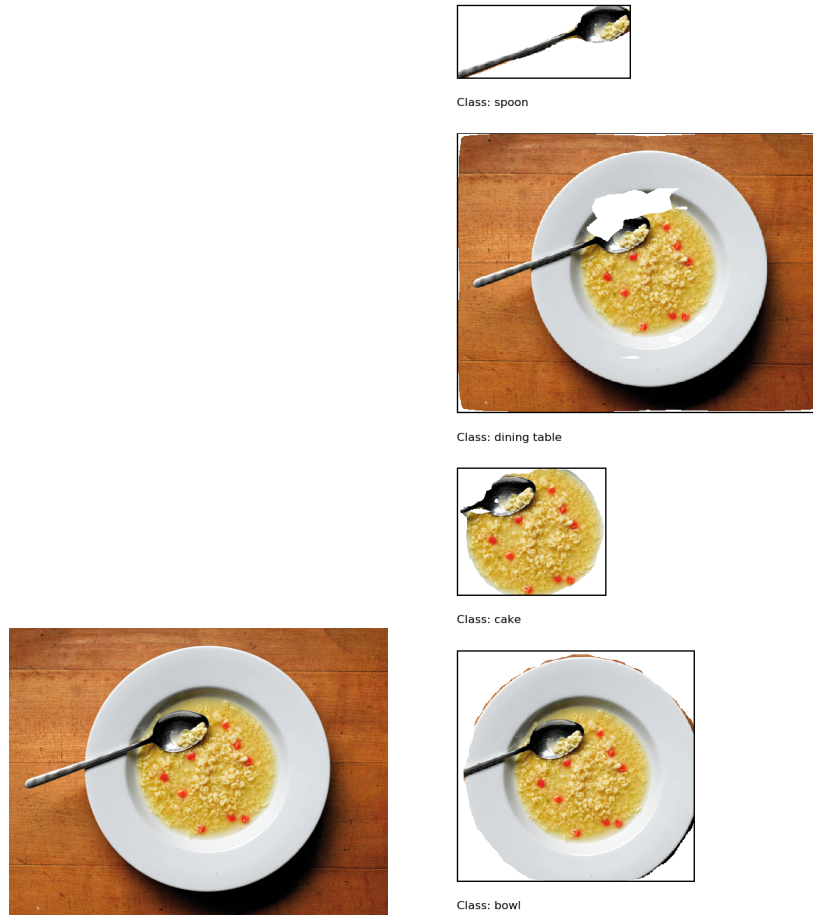


Figure 5.3: Sample input and output image

5.6 Building the application

After completing the tidied up image, the next step was to build an application that can load an image, run the model in inference mode and return the tidied up image from the input image. A difficulty was that MMDetection, like most deep learning frameworks, needs a Nvidia GPU to run even in inference mode. To solve this problem, a wrapper called SMD was used, that takes in MMDetection models and can run the inference on an arbitrary CPU. [2]

To convert the program into a full-blown web application, a framework, called Plotly-Dash has been used. Plotly-Dash is using Flask to create a webserver and is using HTML-, CSS- and JavaScript-technologies under the hood to create a running web application from a Python program. It offers out-of-the-box user interface (UI) components, that are useful to rapid prototype a web application. With the help of UI-elements like buttons and dropdown-lists, the user is given the possibility to upload and select images and models and to start the inference by himself.

5.7 Deployment

To deploy the application, server space on the EnterpriseLab has been allocated. The web application can be accessed at: `http://bdaf20-iameyer.enterpriselab.ch`.

5.8 Refining the model

Hello

5.9 Results

Hello

6 Evaluation und Validation

6.1 Vergleich mit Anforderungen

6.2 Technische Aspekte

6.3 Vorgehen

List of Figures

1.1	Sample image from COCO dataset	2
1.2	Sample image from artist David LaChapelle	2
2.1	Architecture of R-CNN	4
2.2	Tasks and approaches in object recognition	5
5.1	All 80 classes from COCO dataset 2017	8
5.2	Sample binary mask image	10
5.3	Sample input and output image	11

List of Tables

Formelverzeichnis

Bibliography

- [1] Girshick, R.B., Donahue, J., Darrell, T., & Malik, J. (2014). Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. 2014 IEEE Conference on Computer Vision and Pattern Recognition, 580-587.
- [2] Karaniewicz, A. (n.d.). SMD (simple mmdetection) . Retrieved May 30, 2020, from <https://github.com/akarazniewicz/smd>