Lucerne University of
Applied Sciences and Arts

HOCHSCHULE
LUZERN

Informatik

Object detection in fine-art photography

Themenbereiche:	Artificial Intelligence, Visual Computing
Studierender:	Fabian Meyer
Betreuungsperson:	Dr. Simone Lionetti
Experte:	Roman Bachmann, Swisscom
Keywords:	Object detection, Machine learning, Computer vision, Photography

1 Task description

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.

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

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

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

2 Results

A dataset consisting of 42 images from four different artists has been gathered. The frameworks TensorFlow, Detectron2 and MMDetection have been tried out on Google Colab on this dataset with each an implementation of an instance segmentation model. MMDetection has been chosen as it has a very high-level API and has some of the best state of the art models avaiable. Mask R-CNN has been chosen as the model to use for the project realisation, as it has the better performance compared to CenterNet. The model chosen has a ResNet-101 backbone and contains a feature-pyramid-network. With the MMDetection framework and this model, a web-application has been developed. The web-application has been deployed on the EnterpriseLab, it can be accessed at: http://bdaf20-iameyer.enterpriselab.ch. In a further step, the existing Mask R-CNN model has been taken and retrained on Google Colab with fine-art photography data from the series "Kunst aufräumen". The performance of the out-of-the-box Mask R-CNN model and the retrained model version had been compared.

In sum the performance of the pretrained on COCO-dataset Mask R-CNN model depends on the following criteria: The class of objects depicted in the image (included in COCO-dataset or not), the size of objects in the image and the degree to which an object is visible and is shown in a natural style.

When applied to images of fine-art photography we can observe that performance of object-detection models is highly variable to some extend because different photographers use different styles to depict their objects in their images. For example when using an image with a lot of smaller objects (like Andreas Gursky does), performance is rather poor compared to an image with fewer and bigger objects.

3 Solution

The web application analyses images which can either be selected from a predefined list or uploaded by the user or fetched via a url. Then inference is run on the chosen image. The chosen model detects potential objects in the image and returns them as a data structure, containing masks, bounding-boxes, name of class and confidence score.
The program reads these four results and cuts out all objects from the original image. The found objects will be depicted on a new image in a grid-like fashion. One can see a high-level view of the program-flow figure 1.

Cutting out the objects is possible with the help of the bounding-box and the mask of one object alone.

As there are a lot of frameworks and libraries for deep-learning and object detection that are using Python as their main programming-language, Python has been chosen as the language to develop the full project. To transform the program into a full working web application, Plotly Dash has been chosen. Dash is using HTML-, CSS-, and JavaScript-technologies under the hood to get a Python app running on a web server.

For the framework, MMDetection has been chosen because of its high level API and vast array of models avaiable. As MMDetection (like most frameworks) needs a GPU for training and inference, SMD (a wrapper around mmdetection models) is used as a way to perform inference on CPU.

A sample input image from "Kunst aufräumen" and the corresponding output is shown in figure 2 and figure 3.

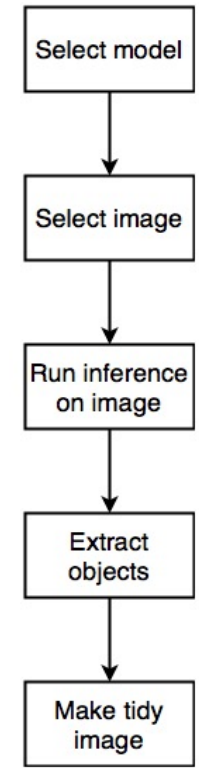

Figure 1: Control-flow diagram

Figure 2: Sample input image

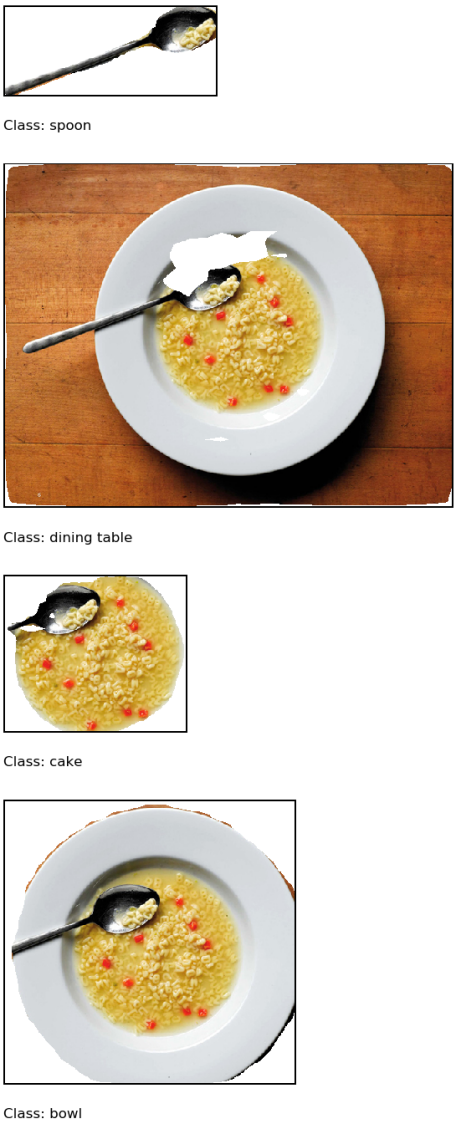

Figure 3: Sample output image

4 Challenges

One big challenge was, that at the time of writing, MMDetection did not support inference performed on CPU. As a consequence, a wrapper around the existing model is used to get a model running on a standard CPU, for development purpose but also for deployment on a server. As most deep-learning networks are trained with GPUs, the most performant (and also most preferable) way to deploy an app would be to use a GPU environment. Using a CPU for inference results in much longer computation time. This is a big drawback, in terms of user experience, as the user has to wait quite some time until the output is shown.

5 Outlook

After the experience of using an existing model for detecting objects in fine-art photography and retraining it with sample data from artistic photography, more models could be retrained with more data to gain a deeper understanding of the detection of objects in images from different photographers and artists.

In a future work one could for example train a model on images from one photographer and test it on images from another photographer that contains the same classes of objects in it. In technical terms it would be interesting to decrease the computation time of the inference as much as possible, either by using a GPU-powered environment for deployment or by using a more recent framework that provids sufficiently fast computation on CPU.