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 Initial task

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 success of object detection models is usually measured with benchmark metrics on standard datasets such as MNIST, PASCAL VOC, ImageNet, COCO and Open Images. These reference scores summarise a complex set of performance parameters and are essential for a meaningful comparison of different approaches. Precisely because they are extremely synthetic, however, it is important to assess the quality of results with other, more ingenuous criteria.

2 Results

During the project cycle the following results have been achieved:

- A dataset from different artists with different styles has been gathered
- Multiple object detection frameworks and models have been tested on these images of fine-art photography
- The most promising framework (mmdetection) and model (Mask R-CNN) have been selected
- A program that takes in an image and puts out a tidied up version with all objects found in it has been developed
- The program has been converted into a web app
- The web app has been deployed on the EnterpriseLab infrastructure
-

The web application can be viewed at: `http://bdaf20-iameyer.enterpriselab.ch`.
In sum the performance of the Mask R-CNN model depends on the following criteria:

- Class of objects in the image (included in COCO-dataset or not)
- Size of objects in the image
- 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 is very variable 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 less number and bigger objects. However with retraining

3 Solution concept

The web application takes in images either from a selection (hosted on the server) or via upload or via a url. Then the model starts inference on the chosen image. The chosen model will detect potential objects in the image and returns them as a data structure, containing masks, bounding boxes, name of class and confidence score.
The program takes in the four results from the model and cuts out all objects from the original image. The found objects will be depicted on a new image in a grid-like fashion.

Cutting out the objects is possible with the bounding box and the object mask alone. One can cut out an object with the bounding box and colour all pixels for example with white colour which do not belong to the mask.

As nowadays most of research and work in the field of deep-learning is done in Python, 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 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 ease to use. As mmdetection (like most frameworks) needs a GPU for training and inference, smd (a wrapper around mmdetection models) has been chosen as a way to do inference on CPU.

One can see a high-level view of the program in image: figure 1 on page 2 and a sample output image in figure 2 on page 3

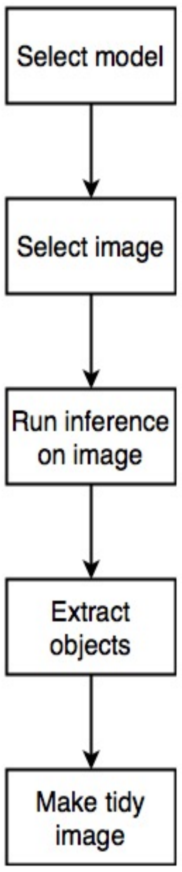

Figure 1: Control-flow diagram

4 Special challenges

The biggest challenge was 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 easiest (and also most performant) way to deploy an app would be to use a GPU environment. However there are some frameworks that do accept GPU-trained networks and can use a CPU for inference. Still, using a CPU for inference results in much longer computation time.

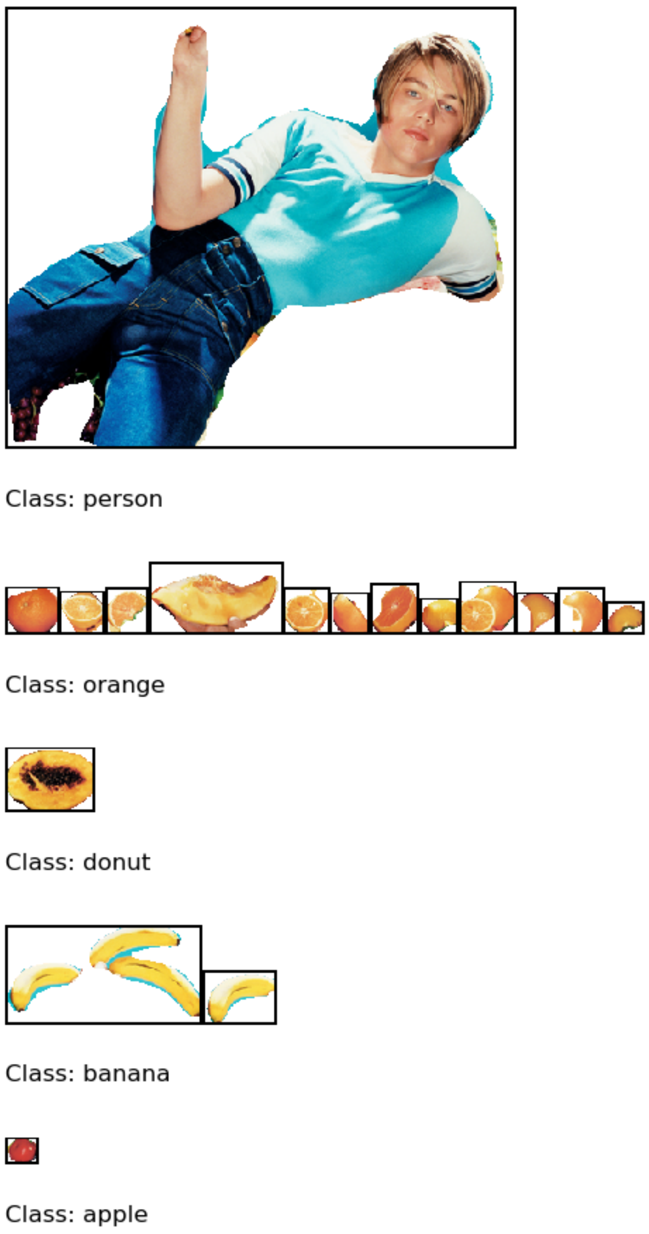

Figure 2: Sample output image

5 Outlook

In a further work, more models could be retrained to gain a deeper understanding of the detection of objects in images from different photographers. 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.