**Kratzererkennung mit KI**

Bei der Bilderkennung mittels künstlicher Intelligenz gibt es prinzipiell drei große Verfahren: Bilderkennung, Objekterkennung und Segmentierung. Für die Erkennung von Kratzern und Fehlern eignet sich vor allem die Bildsegmentierung, weil dadurch Klassifikationen auf Pixel-Ebene durchgeführt werden können. Möchte man auch zwischen Objekten unterscheiden können, ist ein Spezialfall der Segmentierung anzuwenden: Instanz Segmentierung. Für Instanz Segmentierung gibt es eine sehr gut dokumentierte und erforschte Architektur: MASK R-CNN. (<https://github.com/matterport/Mask_RCNN>). Instanz Segmentierung msit MASK R-CNN ist für Tensorflow/Keras implementiert (MASK R-CNN Matterport). Der Code der Quox befindet sich hier und kann in Colaboratory ausgeführt werden (<https://colab.research.google.com/drive/1Fabb54dzUddkwJv6dPHYwCxmooHuVKXz>)

MASK R-CNN kann auch über Detectron2 auf PyTorch implementiert werden (<https://github.com/facebookresearch/detectron2>).  Der Code kann in diesem Colab ausgeführt werden (<https://drive.google.com/open?id=1mceucvL7FysEXV61UKpUZV6Ff9f-OxL0>).

# MASK RCNN by matterport

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

sample demo.ipynb

* imgaug is needed -> need to install opencv-python. However  there is no opencv-python (and opencv-python-headless) wheel for ARM on PyPI.
  + Reference: <https://forums.developer.nvidia.com/t/imgaug-on-jetson-nano/79415/5>.
* Other indispensable Packages: imgaug 0.4.0, qt4-default, libgeos-dev, pycocotools. (all packages installed in virtual env ´~/venv´ including opencv-python to avoid conflicting with opencv 4.1.1 in the system)
* Install pycocotools: <https://github.com/matterport/Mask_RCNN/issues/6>
* demo configuration <https://pastebin.com/DpvrrTWa>
* ResourceExhaustedError: OOM when allocating tensor with shape[3,3,256,256] and type float on /job:localhost/replica:0/task:0/device:GPU:0 by allocator GPU\_0\_bfc [Op:Mul] name: res4m\_branch2b/random\_uniform/mul/
  + Use more memory:
  + <https://www.jetsonhacks.com/2019/04/14/jetson-nano-use-more-memory/>
  + <https://forums.developer.nvidia.com/t/tensorflow-always-out-of-memory/72535>
  + <https://www.zaferarican.com/post/how-to-save-1gb-memory-on-jetson-nano-by-installing-lubuntu-desktop>
* tensorflow has no attribute "log": downgrade to tf 1.13.1
* cannot import name 'saving' from keras.engine: saving.py is erst ab keras 2.2.0 available, upgrade to keras 2.3.1
* InternalError: Dst tensor is not initialized. [[{{node \_arg\_Placeholder\_658\_0\_619}}]]
  + full log: <https://pastebin.com/creMtWPV>
  + this error message is generated when there is not enough memory to handle the batch size: <https://stackoverflow.com/questions/37313818/tensorflow-dst-tensor-is-not-initialized>
  + decrease image per GPU from 2 to 1 in coco.py. (2 image per GPU is used for 12GB RAM)
  + OOM caused by operation keras.backend.random\_uniform, add report\_tensor\_allocations\_upon\_oom to RunOptions: <https://stackoverflow.com/questions/49665757/how-to-add-report-tensor-allocations-upon-oom-to-runoptions-in-keras>
* ResourceExhaustedError (see above for traceback): OOM when allocating tensor with shape[1024,324] and type float on /job:localhost/replica:0/task:0/device:GPU:0 by allocator GPU\_0\_bfc     [[node mrcnn\_bbox\_fc/random\_uniform/RandomUniform (defined at /home/dlinano/venv/lib/python3.6/site-packages/keras/backend/tensorflow\_backend.py:4357) ]]
* install jetson-stats for gpu monitoring: sudo jtop
* on jetson platform Memory is shared between CPU and iGPU: [*https://docs.nvidia.com/cuda/cuda-for-tegra-appnote/index.html#memory-management*](https://docs.nvidia.com/cuda/cuda-for-tegra-appnote/index.html#memory-management)
* [*https://stackoverflow.com/questions/36927607/how-can-i-solve-ran-out-of-gpu-memory-in-tensorflow/44849124#44849124*](https://stackoverflow.com/questions/36927607/how-can-i-solve-ran-out-of-gpu-memory-in-tensorflow/44849124#44849124)
* **Mask RCNN Literature**
* <https://www.topbots.com/semantic-segmentation-guide/>
* <https://www.apriorit.com/dev-blog/580-mask-r-cnn-using-tensorrt>
* <https://blog.csdn.net/mengjiexu_cn/article/details/95619272>
* <https://jkjung-avt.github.io/tf-trt-on-nano/>
* **MaskRcnn with MobileNetV1 as backbone**
* <https://github.com/gustavz/Mobile_Mask_RCNN>
* convert detection\_demo.ipynb to .py file and run in bash console to save memory used by Jupyter
* add the following lines at the beginning of /mmrcnn/model.py
* ##FIX OOM  
  config = tf.ConfigProto()  
  config.gpu\_options.allow\_growth = True  
  config.gpu\_options.per\_process\_gpu\_memory\_fraction = 0.10 # 10% of memory  
  session = tf.Session(config=config)