**Anleitung zu Mask R-CNN**

Version: 10.01.2022

Editor: Stephan Sibirtsev

**Inhaltsverzeichnis**

[1 Anlernen des Mask R-CNN 4](#_Toc92719102)

[1.1 Daten zum Anlernen mit VGG Image Annotator erstellen 4](#_Toc92719103)

[1.2 Anlernen auf dem Cluster (Linux, GPU) 5](#_Toc92719104)

[1.2.1 Der Hochleistungsrechner (HPC) 5](#_Toc92719105)

[1.2.2 Installation von Anaconda 6](#_Toc92719106)

[1.2.3 Anaconda Umgebung erstellen 6](#_Toc92719107)

[1.2.4 Daten zum Anlernen vorbereiten 7](#_Toc92719108)

[1.2.5 Testen von Jobs auf dem GPU node 7](#_Toc92719109)

[1.2.6 Job erstellen 8](#_Toc92719110)

[1.2.7 Job einreichen 10](#_Toc92719111)

[1.3 Anlernen auf dem Rechner (Windows, CPU/GPU) 10](#_Toc92719112)

[1.3.1 Installation von Anaconda 10](#_Toc92719113)

[1.3.2 Anaconda Umgebung erstellen 11](#_Toc92719114)

[1.3.3 Anaconda Umgebung zuordnen 11](#_Toc92719115)

[1.3.4 Daten zum Anlernen vorbereiten 11](#_Toc92719116)

[1.3.5 Mask R-CNN anlernen 12](#_Toc92719117)

[2 Auswerten mit dem Mask R-CNN 14](#_Toc92719118)

[2.1 Auswerten auf dem Cluster (Linux, CPU) 14](#_Toc92719119)

[2.1.1 Der Hochleistungsrechner (HPC) 14](#_Toc92719120)

[2.1.2 Installation von Anaconda 14](#_Toc92719121)

[2.1.3 Anaconda Umgebung erstellen 14](#_Toc92719122)

[2.1.4 Daten zur Auswertung vorbereiten 14](#_Toc92719123)

[2.1.5 Testen von Jobs auf dem CPU node 15](#_Toc92719124)

[2.1.6 Job erstellen 16](#_Toc92719125)

[2.1.7 Job einreichen 17](#_Toc92719126)

[2.2 Auswerten auf dem Rechner (Windows, CPU/GPU) 18](#_Toc92719127)

[2.2.1 Installation von Anaconda 18](#_Toc92719128)

[2.2.2 Anaconda Umgebung erstellen 18](#_Toc92719129)

[2.2.3 Anaconda Umgebung zuordnen 18](#_Toc92719130)

[2.2.4 Die Daten zur Auswertung vorbereiten 18](#_Toc92719131)

[2.2.5 Bilder Auswerten 18](#_Toc92719132)

[3 Anhang 20](#_Toc92719133)

[3.1 Liste der Terminal Befehle 20](#_Toc92719134)

[3.2 Liste der SLURM Job Parameter 21](#_Toc92719135)

[3.3 Liste der MASK R-CNN Job Parameter 22](#_Toc92719136)

[3.4 Liste der GPU nodes 24](#_Toc92719137)

[3.5 Weitere Infos 24](#_Toc92719138)

# Anlernen des Mask R-CNN

## Daten zum Anlernen mit VGG Image Annotator erstellen

1. Die Erstellung der Daten mit dem Programm VGG Image Annotator (VIA) erfolgt auf dem eigenen Rechner bzw. AVT Rechner / Server
2. Bilder für das Training aufteilen in z.B. 90% Trainings-/Validierungbilder und 10% Testbilder
3. Trainings- / Validierungbilder im folgenden Ordner ablegen:  
   ...\VGG Image Annotator\data\train\_val
4. Testbilder im folgenden Ordner ablegen:  
   ...\VGG Image Annotator\data\test
5. Trainings- Validierungbilder und Testbilder separat mit dem Programm VIA manuell auswerten, dazu:
6. Das Programm ***VIA.html*** im Internet-Browser öffnen und über den Reiter „Image“ → „Load or Add Images“ die Bilder reinladen
7. Bilder nacheinander manuell auswerten, dazu mit der geeigneten „Region Shape“ das auszuwertende Objekt (z.B. Tropfen) markieren
8. Die manuelle Auswertung unter dem Reiter „Annotation“ → „Save as JSON“ als ***train.json*** bzw. ***test.json*** im entsprechenden Ordner (train\_val oder test) speichern
9. Bei unterschiedlichen Bildqualitäten, z.B. aufgrund der Bildaufnahme an unterschiedlichen Betriebspunkten (z.B. Hold-up Variation), für die Trainings- /´Validierungbilder jeder Bildqualität die Schritte 2 bis 8 separat durchführen und die Trainings- / Validierungbilder sowie die entsprechende ***<FileName>.json*** Datei in folgender Unterordnerstruktur ablegen (i ist dabei die Nummer der Bildqualität):

..\VGG Image Annotator\data\train\_val\Quality\_i

1. Die Auswertung der Testbilder aller Bildqualitäten kann auch in einem durchgeführt werden.

## Anlernen auf dem Cluster (Linux, GPU)

### Der Hochleistungsrechner (HPC)

**Allgemeine Informationen:**

<https://help.itc.rwth-aachen.de/service/rhr4fjjutttf>

**HPC-Account anlegen:**

<https://help.itc.rwth-aachen.de/service/rhr4fjjutttf/article/994489e7ce8749b69e5615d513e159c2>

**Zugangsmöglichkeiten zum HPC:**

1. Über Desktop client FastX (graphische Oberfläche, Linux):

* Name: **beliebiger Name**
* Host: **login18-x-1.hpc.itc.rwth-aachen.de**
* Port: **22**
* User: **Tim-Kennung**
* Launch Session  XFCE
* [Downloadlink & Anleitung](https://help.itc.rwth-aachen.de/service/rhr4fjjutttf/article/25f576374f984c888bb2a01487fef193)

1. Über WinSCP (remote file transfer):

* Rechnername: **login18-x-1.hpc.itc.rwth-aachen.de**
* Port: **22**
* Benutzername: **Tim-Kennung**
* [Downloadlink & Anleitung](https://help.itc.rwth-aachen.de/service/sl0p0u7tdi5s/article/9229ea232b5f4a1c99ddd57c07a929cb)

1. Remotedesktopverbindung (graphische Oberfläche, Windows):

Computer: **cluster-win.rz.rwth-aachen.de**

Benutzername: **WIN-HPC\Tim-Kennung**

1. Über SSH (text-mode remote access):

https://help.itc.rwth-aachen.de/service/rhr4fjjutttf/article/10c8f0d9b0064013aa439f0b504cc806

**Voreinstellungen bei Zugang über FastX**

1. In Desktop client FastX anmelden (Launch Session  XFCE)
2. Terminal aufrufen
3. Tastaturlayout auf qwertz dauerhaft umstellen:  
   $ echo “setxkbmap de“ >> ~/.bashrc

### Installation von Anaconda

1. In Desktop client FastX anmelden (Launch Session  XFCE)
2. Auf Anaconda [Downloadseite](Anaconda.com/downloads) gehen
3. Auf der Downloadseite die Linux Version wählen und Downloadlink kopieren
4. Terminal aufrufen
5. Ins Home Verzeichnis navigieren, einen tmp Ordner erstellen und in diesen navigieren:  
   $ cd ~  
   $ mkdir tmp  
   $ cd tmp
6. Anaconda bash Skript in den tmp Ordner mit Hilfe des kopierten Links runterladen  
   $ wget https://repo.continuum.io/archive/Anaconda3<Release>.sh
7. In den Ordner, wo das bash Skript gespeichert wurde navigieren und Anaconda über das heruntergeladene bash Skript installieren:  
   $ ls Anaconda3<Release>.sh  
   $ bash Anaconda3<Release>.sh
8. Die base Umgebung von Anaconda mit dem PATH verbinden, damit beim Start des Terminals Anaconda mit aktiviert wird:  
   $ /home/<UserId>/anaconda3/bin/conda init
9. Terminal neustarten
10. Installation verifizieren:  
    $ cd ~  
    $ source .bashrc  
    $ python
11. Es sollte folgendes im Terminal angezeigt werden wenn alles geklappt hat:  
    Python 3.6.5 |**Anaconda, Inc.**|...
12. Python beenden:  
    $ exit()
13. Quelle der [Anleitung](https://problemsolvingwithpython.com/01-Orientation/01.05-Installing-Anaconda-on-Linux/)

### Anaconda Umgebung erstellen

1. Über WinSCP mit dem Cluster verbinden und die ***env\_mrcnn\_linux\_gpu.yml*** Datei in einen beliebigen Ordner auf den HPC übertragen
2. In Desktop client FastX anmelden (Launch Session  XFCE)
3. In den Ordner, wo die ***env\_mrcnn\_linux\_gpu.yml*** Datei hinterlegt ist navigieren und das Terminal dort öffnen (Rechtsklick --> Terminal hier öffnen)
4. Mit Anaconda basierend auf der ***env\_mrcnn\_linux\_gpu.yml*** Datei die Umgebung für Mask R‑CNN erstellen.

$ conda env create -f env\_mrcnn\_linux\_gpu.yml

1. Prüfen ob Anaconda Umgebung **env\_mrcnn\_gpu** installiert wurde:  
   $ conda env list
2. Quelle der [Anleitung](https://docs.conda.io/projects/conda/en/4.6.1/user-guide/tasks/manage-environments.html#create-env-file-manually)

### Daten zum Anlernen vorbereiten

1. In den Pfad "...\Mask\_R\_CNN\samples\droplet“ navigieren und das ***train\_droplet.py*** Skript öffnen. Die Variable cluster auf 1 setzen, cluster=0, und das Skript speichern und schließen.
2. Über WinSCP mit dem Cluster verbinden und das Mask R-CNN in einen gewünschten Ordner auf den HPC übertragen.
3. Die zum Anlernen benötigten Files (Trainings-/Validierungbilder sowie ***<FileName>.json*** Datei der Trainings-/Validierungbilder, s. Kapitel 1.1) in den Mask R-CNN Ordner unter folgenden Pfad hinterlegen:  
   .../Mask\_R\_CNN/datasets/input/<TrainValidationFolderName>/Quality\_1
4. Bei unterschiedlichen Bildqualitäten, z.B. aufgrund der Bildaufnahme an unterschiedlichen Betriebspunkten (z.B. Hold-up Variation), für jede Bildqualität einen entsprechenden Ordner anlegen und die benötigten Files wie in Schritt 1 bis 2 beschrieben ablegen (i ist dabei die Nummer der Bildqualität):   
   .../Mask\_R\_CNN/datasets/input/<TrainValidationFolderName>/Quality\_i

### Testen von Jobs auf dem GPU node

1. In Desktop client FastX anmelden (Launch Session  XFCE)
2. Terminal aufrufen
3. Mit dem GPU node verbinden (s. Tabelle 3):

$ ssh -l <UserId> <NodeName>.hpc.itc.rwth-aachen.de

1. Cuda Modul laden:  
   $ module load cuda/10.0
2. Pfad in welchem sich Anaconda befindet exportieren:  
   $ export PATH=$PATH:/home/<UserID>/anaconda3/bin
3. Anaconda Umgebung aktivieren:  
   $ source activate env\_mrcnn\_gpu
4. In den Pfad in welchem sich das ***train\_droplet.py*** Skript befindet navigieren:  
   $ cd /home/<UserID>/.../Mask\_R\_CNN/samples/droplet/
5. Das ***train\_droplet.py*** Skript mit Memorytracker ausführen (am Stück eingeben, Parameter ohne Deteitypendungen, Beschreibung / Standardeinstellungen der Parameter s. Tabelle 3):  
   $ r\_memusage python train\_droplet.py --dataset\_path=<TrainValidationFolderName> --name\_result\_file=<ExcelFileName> --new\_weights\_path=<WeightsFolderName> --base\_weights=coco --image\_max=<MaxImageSize> --masks=0 --device=1 --images\_gpu=<NumberImages> --early\_stopping=0 --epochs=5 --dataset\_quantity=100 --k\_fold=<NumberFolds> --k\_fold\_val=0
6. Nach 10 Minuten abbrechen mit Strg + Shift + C
7. Eintrag für “peak usage: physical memory” auf Gigabyte aufrunden und notieren
8. Rechendauer pro Epoche auf Minuten aufrunden und notieren

### Job erstellen

1. Auf den eigenen Rechner ***<FileName>.job*** Datei erstellen und mit Notepad (oder alternativem Texteditor) öffnen. Beispiel s. ***Vorlage\_GPU\_Job\_Datei.job***
2. Die ***<FileName>.job*** Datei auf Unix Konvertierung umstellen.  
   Bei Notepad auf Edit --> EOL Conversion --> Unix (LF)
3. SBATCH Einträge erstellen (Beschreibung / Standardeinstellungen der Parameter s. Tabelle 2):
   1. Muss am Anfang jedes Jobs stehen  
      #!/usr/bin/zsh
   2. Job Name:  
      #SBATCH --job-name=<JobName>
   3. Pfad und Name der Outputdatei der Job Durchführung:  
      #SBATCH --output=/home/<UserID>/.../<JobOutputFolderName>/%x\_%J\_output.txt
   4. Laufdauer des Jobs (wird durch das Testen von Jobs auf dem GPU node ermittelt, Rechendauer pro Epoche mit Anzahl der zu trainierenden Epochen und ggf. Sicherheitsfaktor multiplizieren):  
      #SBATCH --time=d-hh:mm:ss
   5. Arbeitsspeicherbedarf pro CPU (wird durch das Testen von Jobs auf dem GPU node ermittelt, ggf. Sicherheitsfaktor draufpacken. Arbeitsspeicherbedarf hier: 5G)  
      #SBATCH --mem-per-cpu=<AmountMemory>
   6. Arbeitsspeicherbedarf pro GPU (wird durch das Testen von Jobs auf dem GPU node ermittelt, ggf. Sicherheitsfaktor draufpacken. Arbeitsspeicherbedarf hier: 5G)  
      #SBATCH --mem-per-gpu=<AmountMemory>
   7. Email Adresse:  
      #SBATCH --mail-user=<EmailAdress>
   8. Emails die erhalten werden sollen:  
      #SBATCH --mail-type=ALL
   9. Anzahl an tasks, die durchgeführt werden sollen:  
      #SBATCH --ntasks=1
   10. Anzahl der benötigten GPUs pro node.  
       #SBATCH --gres=gpu:1
4. Terminal Einträge erstellen:
   1. Das Cuda Modul laden:  
      module load cuda/10.0
   2. Pfad in welchem sich Anaconda befindet exportieren:  
      export PATH=$PATH:/home/<UserID>/anaconda3/bin
   3. Umgebung aktivieren:  
      source activate env\_mrcnn\_gpu
   4. Sich in den Pfad, wo sich das ***train\_droplet.py*** Skript befindet navigieren:  
      cd /home/<UserID>/.../ Mask\_R\_CNN/samples/droplet/
   5. Das ***train\_droplet.py*** Skript ausführen (als eine Zeile eingeben, Parameter ohne Deteitypendungen, Beschreibung / Standardeinstellungen der Parameter s. Tabelle 3):  
      python train\_droplet.py --dataset\_path=<TrainValidationFolderName> --name\_result\_file=<ExcelFileName> --new\_weights\_path=<WeightsFolderName> --base\_weights=coco --image\_max=<MaxImageSize> --masks=0 --device=1 --images\_gpu=<NumberImages> --early\_stopping=0 --epochs=<NumberEpochs> --dataset\_quantity=<QuantityDataset> --k\_fold=<NumberFolds> --k\_fold\_val=<FoldNumber>
5. ***<FileName>.job*** Datei speichern

### Job einreichen

1. Über WinSCP mit dem Cluster verbinden und die ***<FileName>.job*** Datei in einen gewünschten Ordner auf dem Cluster übertragen.
2. In Desktop client FastX anmelden (Launch Session  XFCE)
3. In den Ordner, wo die ***<FileName>.job*** Datei hinterlegt ist navigieren und das Terminal dort öffnen (Rechtsklick --> Terminal hier öffnen)
4. Job einreichen:  
   $ sbatch <FileName>.job
5. Prüfen ob der Job in der Warteschleife angekommen ist:  
   $ squeue -u <UserID>
6. Sobald der Job startet, bekommt man eine Benachrichtigung per Email
7. Sobald der Job Fertig ist oder aufgrund eines Fehlers beendet wird, bekommt man eine Benachrichtigung per Email. Die Job-Output Datei mit dem dokumentierten Job-Verlauf (Terminalausgabe des Mask R-CNN) ist unter folgenden Ordner gespeichert: /home/<UserID>/.../<JobOutputFolderName>/
8. Sobald der Job fehlerfrei durchgelaufen ist, werden die Outputfiles des Mask R‑CNN (Gewichte der einzelnen Epochen) im folgenden Ordner gespeichert:  
   .../Mask\_R\_CNN/droplet\_logs/<WeightsFolderName>/

## Anlernen auf dem Rechner (Windows, CPU/GPU)

### Installation von Anaconda

1. Auf Anaconda [Downloadseite](Anaconda.com/downloads) gehen:
2. Auf der Downloadseite die Windows Version runterladen und installieren. Bei Installation unbedingt „**Install for: Just Me**“ (So funktioniert die Installation auch auf dem AVT Rechner/Server ohne Adminrechte) und „**Add Anaconda to my PATH environment variable”** auswählen.
3. Quelle der [Anleitung](https://www.datacamp.com/community/tutorials/installing-anaconda-windows)

### Anaconda Umgebung erstellen

1. Bevorzugten Quellcode Editor aufrufen (z.B. Visual Studio Code)
2. Geeignete Umgebung (CPU, GPU AMD oder GPU NVIDIA) wie folgt über die entsprechende ***<FileName>.yml*** Datei (***env\_mrcnn\_windows\_cpu.yml***, ***env\_mrcnn\_windows\_gpu\_amd.yml*** oder ***env\_mrcnn\_windows\_gpu\_nvidia.yml***) installieren:
3. In den Ordner, wo die ***<FileName>.yml*** Datei hinterlegt ist navigieren (hier sind die '...' sehr wichtig):  
   cd '...\<FolderName>\'
4. Mit Anaconda auf Basis der ***<FileName>.yml*** Datei die Umgebung für Mask R-CNN erstellen.

conda env create -f <FileName>.yml

1. Liste der Umgebungen nach der installierten Umgebung checken, um zu prüfen, ob alles geklappt hat:  
   conda env list
2. Quelle der [Anleitung](https://docs.conda.io/projects/conda/en/4.6.1/user-guide/tasks/manage-environments.html#create-env-file-manually)

### Anaconda Umgebung zuordnen

1. In den Ordner Mask\_R\_CNN navigieren und die ***Mask\_RCNN.code-workspace*** Datei mit einem Quellcode Editor (z.B. Visual Studio Code) öffnen.
2. Im geöffneten Workspace in den Pfad "...\Mask\_R\_CNN\samples\droplet“ navigieren und das ***train\_droplet.py*** Skript öffnen.
3. Python Pfad spezifizieren:  
   „Strg + Shift +P“ drücken → „Preferences: Open User Settings“ eingeben → „Python: Python Path“ suchen → Pfad der Anaconda Umgebung angeben (z.B.: „C:\Anaconda3\envs\env\_mrcnn\_windows\_cpu\python.exe“).
4. Dem ***evaluate\_droplet.py*** Skript die Anaconda Umgebung zuordnen:  
   „Strg + Shift + P“ drücken → „Python: Select Interpreter“ eingeben → „Mask\_R\_CNN“ auswählen → Anaconda Umgebung auswählen (z.B.: „C:\Anaconda3\envs\env\_mrcnn\_windows\_cpu\python.exe“.

### Daten zum Anlernen vorbereiten

1. Die zum Anlernen benötigten Files (Trainings-/Validierungbilder sowie ***<FileName>.json*** Datei der Trainings-/Validierungbilder, s. Kapitel 1.1) in den Mask R-CNN Ordner unter folgenden Pfad hinterlegen:  
   .../Mask\_R\_CNN/datasets/input/<TrainValidationFolderName>/Quality\_1
2. Bei unterschiedlichen Bildqualitäten, z.B. aufgrund der Bildaufnahme an unterschiedlichen Betriebspunkten (z.B. Hold-up Variation), für jede Bildqualität einen entsprechenden Ordner anlegen und die benötigten Files wie in Schritt 1 bis 2 beschrieben ablegen (i ist dabei die Nummer der Bildqualität):   
   .../Mask\_R\_CNN/datasets/input/<TrainValidationFolderName>/Quality\_i

### Mask R-CNN anlernen

1. In den Ordner Mask\_R\_CNN navigieren und die ***Mask\_RCNN.code-workspace*** Datei mit einem Quellcode Editor (z.B. Visual Studio Code) öffnen.
2. Alle Einträge unter „Necessary Parameters and Data Names“ ausfüllen (Parameter ohne Deteitypendungen, Beschreibung / Standardeinstellungen der Parameter s. Tabelle 3):

* Auswertung findet nicht auf dem Cluster statt:  
  cluster = 0
* Sollen zu der Erkennung der Bounding Boxen auch die Erkennung der Masken gelernt werden? 1 = ja, 0 = nein

masks = 0

* Wird die Auswertung auf CPU oder GPU durchgeführt? 1=GPU, 0=CPU

device = 1

* Anzahl der Bilder, mit denen auf jeder GPU trainiert wird:

images\_gpu = 1

* Basisgewichte:

base\_weights = "coco"

* max. Bildgröße, z.B. 64, 128, 256, 512, 1024, 2048, ...  
  Den nächstgelegenen Wert wählen, der der größten Seite des Bildes entspricht.

image\_max = <MaxImageSize>

* Soll early stopping vorgenommen werden? 1 = ja, 0 = nein

early\_stopping = 0

* Anzahl der Epochen zu trainieren:

epochs = 50

* Menge des Trainings- / Validierungsdatensatzes in [%]

dataset\_quantity = 100

* Anzahl der folds für die k-fold Kreuzvalidierung

k\_fold = 5

* fold Nummer, die für die Validierung verwendet wird. Beginnend mit 0

k\_fold\_val = 0

* Pfad in welchem die Bilder für die Auswertung hinterlegt sind:  
  dataset\_path = r“<InputFolderName>“
* Pfad in welchem die **h5-Dateien** (Gewichte) gespeichert werden:

new\_weights\_path= r“<WeightsFolderName>“

* Name der Excel-Datei:  
  name\_result\_file = "<ExcelFileName>"

1. Das Skript starten und abwarten bis es durchgelaufen ist.
2. Sobald der Job fehlerfrei durchgelaufen ist, werden die Outputfiles des Mask R‑CNN (Gewichte der einzelnen Epochen) im folgenden Ordner gespeichert:  
   .../Mask\_R\_CNN/droplet\_logs/<WeightsFolderName>/

# Auswerten mit dem Mask R-CNN

## Auswerten auf dem Cluster (Linux, CPU)

### Der Hochleistungsrechner (HPC)

1. s. Kapitel 1.2.1

### Installation von Anaconda

1. s. Kapitel 1.2.2

### Anaconda Umgebung erstellen

1. Über WinSCP mit dem Cluster verbinden und die ***env\_mrcnn\_linux\_cpu.yml*** Datei in einen beliebigen Ordner auf den HPC übertragen
2. In Desktop client FastX anmelden (Launch Session  XFCE)
3. In den Ordner, wo die ***env\_mrcnn\_linux\_cpu.yml*** Datei hinterlegt ist navigieren und das Terminal dort öffnen (Rechtsklick --> Terminal hier öffnen)
4. Mit Anaconda basierend auf der ***env\_mrcnn\_linux\_cpu.yml*** Datei die Umgebung für Mask R‑CNN erstellen.

$ conda env create -f env\_mrcnn\_linux\_cpu.yml

1. Prüfen ob Anaconda Umgebung **env\_mrcnn\_cpu** installiert wurde:  
   $ conda env list
2. Quelle der [Anleitung](https://docs.conda.io/projects/conda/en/4.6.1/user-guide/tasks/manage-environments.html#create-env-file-manually)

### Daten zur Auswertung vorbereiten

1. Auf dem eigenen Rechner in den Pfad „...\Mask\_R\_CNN\samples\droplet“ navigieren und das ***evaluate\_droplet.py*** Skript öffnen.
2. Alle Einträge unter „Necessary Parameters and Data Names“ ausfüllen:

* Auswertung findet auf dem Cluster statt:  
  cluster = 1
* Angabe welches jedes Bild gespeichert werden soll. Hier jedes 2te:

save\_xth\_image = 2

* Grenzwert für Kugelform, Verhältnis zwischen Höhe und Breite des detektierten Tropfens. Hier 0,8:

min\_aspect\_ratio = 0.8

* Pixelgröße in [px/µm] (s. Kameraspezifikation). Für den Sonderfall, dass die Bilder mit einer Sopat-Sonde aufgenommen wurden und mit dem Spoat Programm eine entsprechende ***<FileName>.json*** Datei generiert wurde „pixelsize = 0“ eingeben. Hier 1,00:

pixelsize = 1.00

1. Das ***evaluate\_droplet.py*** Skript speichern und schließen.
2. Über WinSCP mit dem Cluster verbinden
3. Das ***evaluate\_droplet.py*** Skript auf dem Cluster unter folgendem Pfad ablegen:

.../Mask\_R\_CNN/samples/droplet/

1. Die Bilder für die Auswertung auf dem Cluster im folgenden Ordner ablegen:

.../Mask\_R\_CNN/datasets/input/<EvaluationFolderName>

1. Die ***<FileName>.h5*** Dateien mit den Gewichten auf dem Cluster im folgenden Ordner ablegen:

.../Mask\_R\_CNN\droplet\_logs\<WeightsFolderName>

### Testen von Jobs auf dem CPU node

1. In Desktop client FastX anmelden (Launch Session  XFCE)
2. Terminal aufrufen
3. Anaconda Umgebung aktivieren:  
   $ source activate env\_mrcnn\_cpu
4. In den Pfad in welchem sich das ***evaluate\_droplet.py*** Skript befindet navigieren:  
   $ cd /home/<UserID>/.../ Mask\_R\_CNN/samples/droplet/
5. Das ***evaluate\_droplet.py*** Skript mit Memorytracker ausführen (am Stück eingeben, Parameter ohne Deteitypendungen, Beschreibung / Standardeinstellungen der Parameter s. Tabelle 3):  
   $ r\_memusage python evaluate\_droplet.py --dataset\_path=<InputFolderName> --save\_path=<OutputFolderName> --name\_result\_file=<ExcelFileName> --weights\_path=<WeightsFolderName> --weights\_name=<WeightsFileName> --masks=0 --device=0 --image\_max=<MaxImageSize>
6. Nach 10 Minuten abbrechen mit Strg + Shift + C
7. Eintrag für “peak usage: physical memory” auf Gigabyte aufrunden und notieren
8. Rechendauer pro ausgewertetes Bild ausrechnen, auf Minuten aufrunden und notieren

### Job erstellen

1. Auf den eigenen Rechner ***<FileName>.job*** Datei erstellen und mit Notepad (oder alternativem Texteditor) öffnen. Beispiel s. ***Vorlage\_CPU\_Job\_Datei.job***
2. Die ***<FileName>.job*** Datei auf Unix Konvertierung umstellen.  
   Bei Notepad auf Edit --> EOL Conversion --> Unix (LF)
3. SBATCH Einträge erstellen (s. Tabelle 2):
   1. Muss am Anfang jedes Jobs stehen  
      #!/usr/bin/zsh
   2. Job Name:  
      #SBATCH --job-name=<FileName>
   3. Pfad und Name der Outputdatei der Job Durchführung:  
      #SBATCH --output=/home/<UserID>/.../<JobOutputFolderName>/%x\_%J\_output.txt
   4. Laufdauer des Jobs (wird durch das Testen von Jobs auf dem CPU node ermittelt, Rechendauer pro ausgewertetes Bild mit Anzahl der auszuwertenden Bilder und ggf. Sicherheitsfaktor multiplizieren):  
      #SBATCH --time=d-hh:mm:ss
   5. Arbeitsspeicherbedarf pro CPU (wird durch das Testen von Jobs auf dem GPU node ermittelt, ggf. Sicherheitsfaktor draufpacken. Empfehlung sind mind. 15G)  
      #SBATCH --mem-per-cpu=15G
   6. Email Adresse:  
      #SBATCH --mail-user=<EmailAdress>
   7. Emails die erhalten werden sollen:  
      #SBATCH --mail-type=ALL
   8. Anzahl an tasks, die durchgeführt werden sollen:  
      #SBATCH --ntasks=1
4. Terminal Einträge erstellen:
   1. Pfad in welchem sich Anaconda befindet exportieren:  
      export PATH=$PATH:/home/<UserID>/anaconda3/bin
   2. Umgebung aktivieren:  
      source activate env\_mrcnn\_cpu
   3. Sich in den Pfad, wo sich das ***evaluate\_droplet.py*** Skript befindet navigieren:  
      cd /home/<UserID>/.../Mask\_R\_CNN/samples/droplet/
   4. Das ***evaluate\_droplet.py*** Skript ausführen (als eine Zeile eingeben):  
      python evaluate\_droplet.py --dataset\_path=<InputFolderName> --save\_path=<OutputFolderName> --name\_result\_file=<ExcelFileName> --weights\_path=<WeightsFolderName> --weights\_name=<WeightsFileName> --masks=0 --device=0 --image\_max=<MaxImageSize>
5. ***<FileName>.job*** Datei speichern

### Job einreichen

1. Über WinSCP mit dem Cluster verbinden und die ***<FileName>.job*** Datei in einen gewünschten Ordner auf dem Cluster übertragen.
2. In Desktop client FastX anmelden (Launch Session  XFCE)
3. In den Ordner, wo die ***<FileName>.job*** Datei hinterlegt ist navigieren und das Terminal dort öffnen (Rechtsklick --> Terminal hier öffnen)
4. Job einreichen:  
   $ sbatch <FileName>.job
5. Prüfen ob der Job in der Warteschleife angekommen ist:  
   $ squeue -u <UserID>
6. Sobald der Job startet, bekommt man eine Benachrichtigung per Email
7. Sobald der Job Fertig ist oder aufgrund eines Fehlers beendet wird, bekommt man eine Benachrichtigung per Email
8. Sobald der Job fehlerfrei durchgelaufen ist, werden die Outputfiles des Mask R‑CNN im folgenden Ordner gespeichert:  
   /home/<UserID>/.../Mask\_R\_CNN/datasets/output/<OutputFolderName>/

## Auswerten auf dem Rechner (Windows, CPU/GPU)

### Installation von Anaconda

1. s. Kapitel 1.3.1

### Anaconda Umgebung erstellen

1. s. Kapitel 1.3.2

### Anaconda Umgebung zuordnen

1. s. Kapitel 1.3.3

### Die Daten zur Auswertung vorbereiten

1. Die ***<FileName>.h5*** Dateien mit den Gewichten auf den Windowsrechner im Mask R-CNN Ordner unter folgendem Pfad ablegen:

...\Mask\_R\_CNN\droplet\_logs\<WeightsFolderName>

1. Die Bilder für die Auswertung auf dem Windowsrechner im folgenden Ordner ablegen:

...\Mask\_R\_CNN\datasets\input\<EvaluationFolderName>

### Bilder Auswerten

1. In den Ordner Mask\_R\_CNN navigieren und die ***Mask\_RCNN.code-workspace*** Datei mit einem Quellcode Editor (z.B. Visual Studio Code) öffnen.
2. Alle Einträge unter „Necessary Parameters and Data Names“ ausfüllen (Parameter ohne Deteitypendungen, Beschreibung / Standardeinstellungen der Parameter s. Tabelle 3):

* Auswertung findet nicht auf dem Cluster statt:  
  cluster = 0
* Angabe welches jedes Bild gespeichert werden soll, hier jedes 2te:

save\_xth\_image = 2

* Grenzwert für Kugelform, Verhältnis zwischen Höhe und Breite des detektierten Tropfens. Hier 0,8:

min\_aspect\_ratio = 0.8

* Pixelgröße (s. Kameraspezifikation). Für den Sonderfall, dass die Bilder mit einer Sopat-Sonde aufgenommen wurden und eine entsprechende json-Datei vorliegt „pixelsize = 0“ eingeben. Hier 1:

pixelsize = 1.00

* max. Bildgröße, z.B. 64, 128, 256, 512, 1024, 2048, ...  
  Den nächstgelegenen Wert wählen, der der größten Seite des Bildes entspricht.

image\_max = <MaxImageSize>

* Sollen zu den Bounding Boxen auch die Masken detektiert werden (funktioniert nur, wenn ein Netz genutzt wird, das die Erkennung der Masken auch gelernt hat)?   
  1 = ja, 0 = nein

masks = 0

* Wird die Auswertung auf CPU oder GPU durchgeführt? 1=GPU, 0=CPU

device = 1

* Pfad in welchem die Bilder für die Auswertung hinterlegt sind:  
  dataset\_path = r“<InputFolderName>“
* Pfad in welchem die ausgewerteten Bilder und die Excel-Datei gespeichert werden sollen:

save\_path = r“<OutputFolderName>“

* Name der Excel-Datei:  
  name\_result\_file = "<ExcelFileName>"
* Pfad in welchem die **h5-Dateien** (Gewichte) hinterlegt sind:

weights\_path = r“<FolderName>“

* h5-Datei mit den Gewichten, die für die Auswertung genutzt werden soll:  
  weights\_name = r“<FileName>“

1. Das Skript starten und abwarten bis es durchgelaufen ist.
2. Die ausgewerteten Bilder und die Excel-Datei mit der Liste der detektierten Tropfen sind im Mask R-CNN Ordner unter folgendem Pfad zu finden: Mask\_R\_CNN\datasets\output\<OutputFolderName>

# Anhang

## Liste der Terminal Befehle

Tabelle 1: Liste der Terminal Befehle

|  |  |
| --- | --- |
| $ ***r\_memusage*** <PythonScript>.py  Zum Beenden: Umschalt +Strg + C | Startet das Python Skript und gibt nach dem Beenden zurück, wie viel Arbeitsspeicher verbraucht wurde |
| $ ***sbatch*** <JobScript>.job | Job einreichen |
| $ ***squeue*** -u <UserID> | Zeigt laufende Jobs und Jobs in der Warteschlange des Benutzers an |
| $ ***scancel*** <JobID> | Job abbrechen |
| $ ***sinfo*** | Zeigt Cluster-Status an mit der Liste laufender Jobs und Jobs in der Warteschlange |
| $ ***r\_wlm\_usage -q*** | Corestunden Verbrauch anzeigen |

## Liste der SLURM Job Parameter

Tabelle 2: Liste der SLURM Job Parameter (vor jedem Parameter #SBATCH schreiben)

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Beschreibung** | **Standard-**  **einstellung** |
| --account=<ProjectName> | Den Job über ein Projekt einreichen | - |
| --job-name=<JobName> | Job Name | test |
| --output=/home/<UserID>/…/%x\_%J\_output.txt | Speicherort der Outputdatei der Job Durchführung.  „%x“ übernimmt den Namen der Datei.  „%J“ setzt die Job ID dran. | - |
| --time=d-hh:mm:ss | Laufdauer des Jobs in  Tage-Stunden:Minuten:Sekunden | 0-03:00:00 |
| --mail-user=<EmailAdress> | Email Adresse, um Informationen über den Job Fortschritt zu bekommen | - |
| --mail-type=BEGIN  --mail-type=END  --mail-type=ALL | Informationen über Job Fortschritt, die man über Email bekommen möchte | ALL |
| --ntasks=<NumberTasks> | Für Processes/MPI. Anzahl an tasks | 1 |
| --mem-per-cpu=<AmountMemory> | Arbeitsspeicherbedarf pro CPU (nicht pro task!), z. B. 10 G.  K(ilobyte)|G(igabyte)|T(erabyte) | 5 |
| --mem-per-gpu=<AmountMemory> | Arbeitsspeicherbedarf pro GPU (nicht pro task!), z. B. 10 G.  K(ilobyte)|G(igabyte)|T(erabyte) | 5 |
| --gres=gpu:1  --gres=gpu:2 | Anzahl der benötigten GPUs pro node. Die geegnete Partition wird automatisch von dem SLURM System gewählt. | 1 |

## Liste der MASK R-CNN Job Parameter

Tabelle 3: Liste der MASK R-CNN Job Parameter

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Beschreibung** | **Standard-**  **einstellung** |
| cluster=1  *For training & evaluation* | Wird das Programm in einem Cluster ausgeführt? 1 = ja, 0 = nein | 1 |
| dataset\_path=<InputFolderName>  *For training & evaluation* | Input dataset path | test |
| save\_path=<OutputFolderName>  *For evaluation* | Output path | test |
| name\_result\_file=<ExcelFileName>  *For training & evaluation* | Name of the excel result file | test |
| weights\_path=<WeightsFolderName>  *For evaluation* | Input weights path | test |
| new\_weights\_path=<WeightsFolderName>  *For training* | Output weights path | test |
| weights\_name=<WeightsFileName>  *For evaluation* | Weights file name | test |
| base\_weights=coco  *For training* | base weights, here „coco“ | coco |
| image\_max=<MaxImageSize>  *For training & evaluation* | Max. image size for scaling. The multiple of 64 is needed to ensure smooth scaling of feature maps up and down the 6 levels of the FPN pyramid (2\*\*6=64). e.g. 64, 128, 256, 512, 1024, 2048. Select the closest value corresponding to the largest side of the image. | 1024 |
| device=1  *For training & evaluation* | Is the program execution done on CPU or GPU? 1 = GPU, 0 = CPU | 1 |
| masks=0  *For training & evaluation* | Generate detection masks as in Mask-RCNN? If not, output will generate only bounding boxes like in Faster-RCNN. 1 = yes, 0 = no | 0 |
| early\_stopping=0  *For training* | Should early stopping be used?  1 = yes, 0 = no | 0 |
| images\_gpu=<NumberImages>  *For training* | Number of images to train with on each GPU.  A 12GB GPU can typically handle 2 images of 1024x1024px.  Adjust based on your GPU memory and image sizes. Use the highest number that your GPU can handle for best performance. | 1 |
| epochs=<NumberEpochs>  *For training* | Epochs to train | 5 |
| dataset\_quantity=<QuantityDataset>  *For training* | Quantity of train/validation dataset in [%].e.g. for sensitivity analysis: number of images required | 100 |
| k\_fold=<NumberFolds>  *For training* | Number of folds for k-fold cross validation. | 5 |
| k\_fold\_val=<FoldNumber>  *For training* | Fold number to use for validation. Starting with 0 | 0 |

## Liste der GPU nodes

Tabelle 4: Liste der GPU nodes

|  |  |  |
| --- | --- | --- |
| **Node name** | **GPU-cards** | **GPU-nodes** |
| login-g | 2 | 1 |
| login18-g-1 | 2 | 1 |
| login18-g-2 | 2 | 1 |
| lng01..lng09 | 2 | 9 |
| lns07..lns08 | 1 | 2 |
| ncg01..ncg48 | 2 | 48 |
| nrg01..nrg06 | 2 | 6 |

## Weitere Infos

1. Versionsauswahl vorhandener Software (erstmal nur zur Info):

<https://doc.itc.rwth-aachen.de/display/CC/modules+system>

1. Aktivierung von Paketen für Python über das Terminal:

<https://doc.itc.rwth-aachen.de/display/CC/python>

1. Nützliches zu Anaconda in Linux:  
   <https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html>
2. Alternativen um Umgebung zu erstellen:  
   https://docs.conda.io/projects/conda/en/4.6.1/user-guide/tasks/manage-environments.html#create-env-file-manually