Curious Containers: Framework zur Reproduzierbarkeit von digitalen Experimenten

Christoph Jansen (Vortragender), Bruno Schilling, Klaus Strohmenger, Michael Witt, Jonas Annuscheit, Dagmar Krefting

cbmi Center for biomedical image and information processing
Research. Innovation. Incubation.

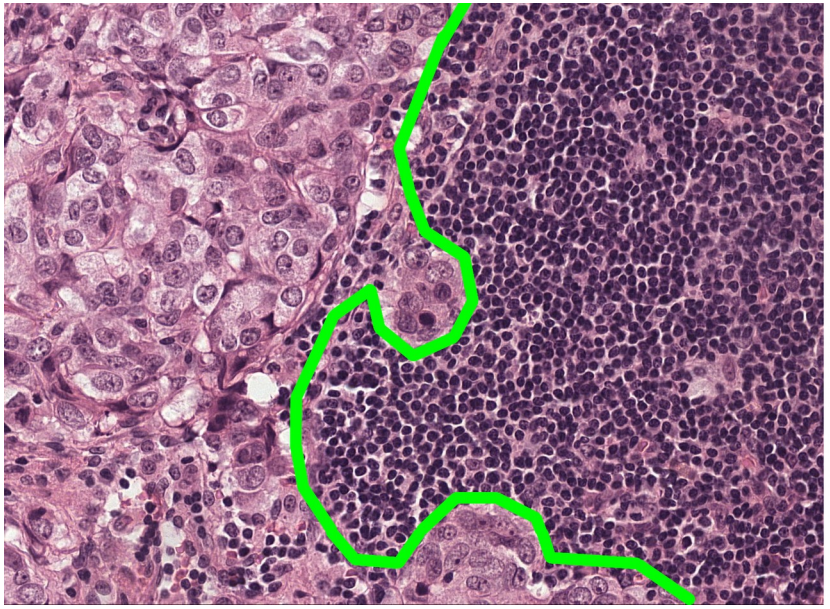
htw Hochschule für Technik und Wirtschaft Berlin
University of Applied Sciences

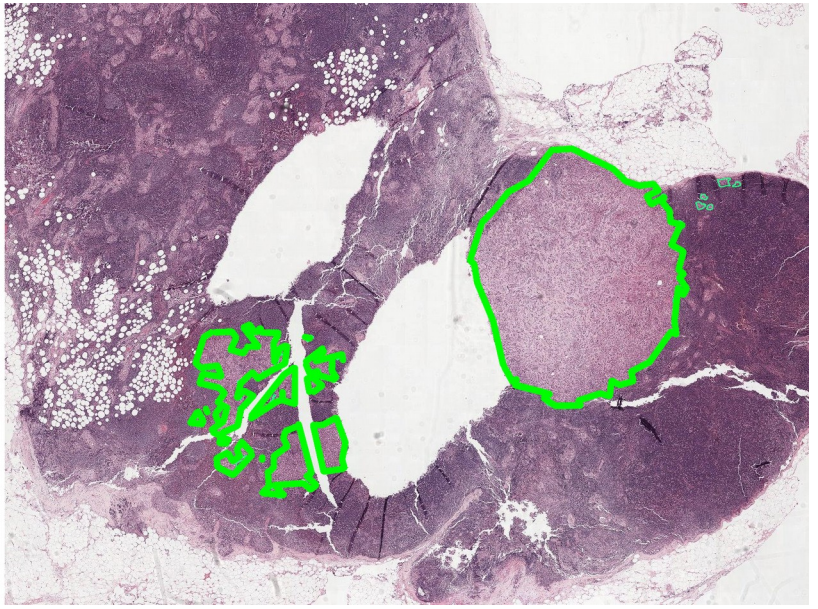
Motivation

- Daten-basierte Experimente
  - Use-Case: Convolutional Neural Networks (CNN) Training
- Ziel 1: Format für publizierbare Experimente
  - Beschreiben, Ausführen, Teilen, Archivieren, **Reproduzieren**
- Ziel 2: Automation
  - Exepriment ist unabhängig von einem bestimmten Computer
  - Cluster-Computing
- Ziel 3: Publikationsprozess

Digitale Pathologie

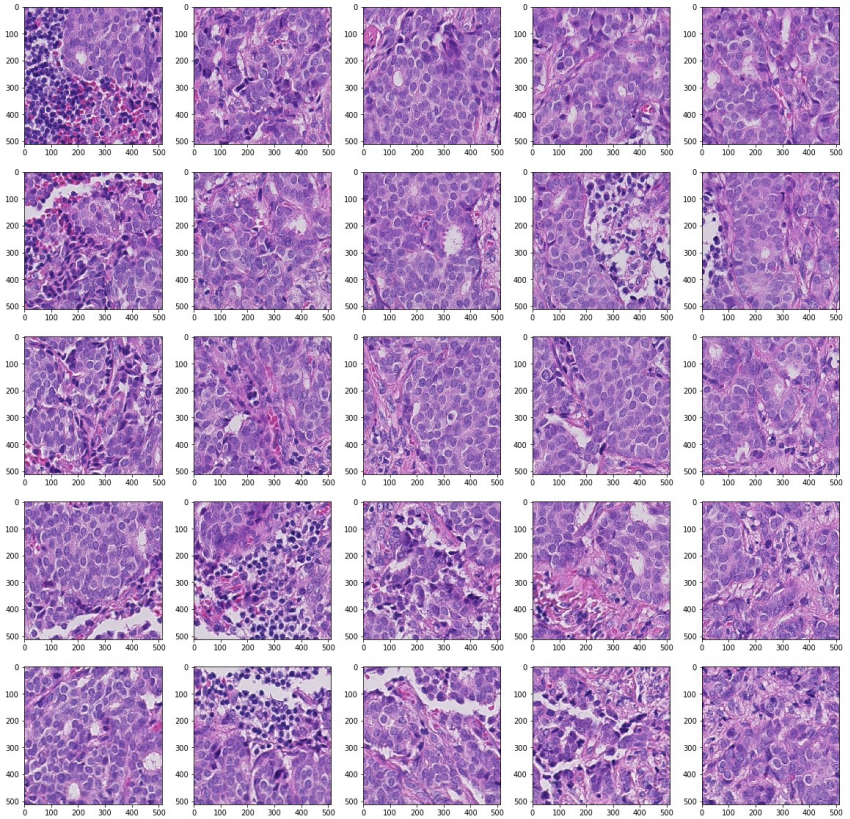
- WSI – Whole Slide Images (~4 GB per File)
  - Krebstumor in Lymphknoten



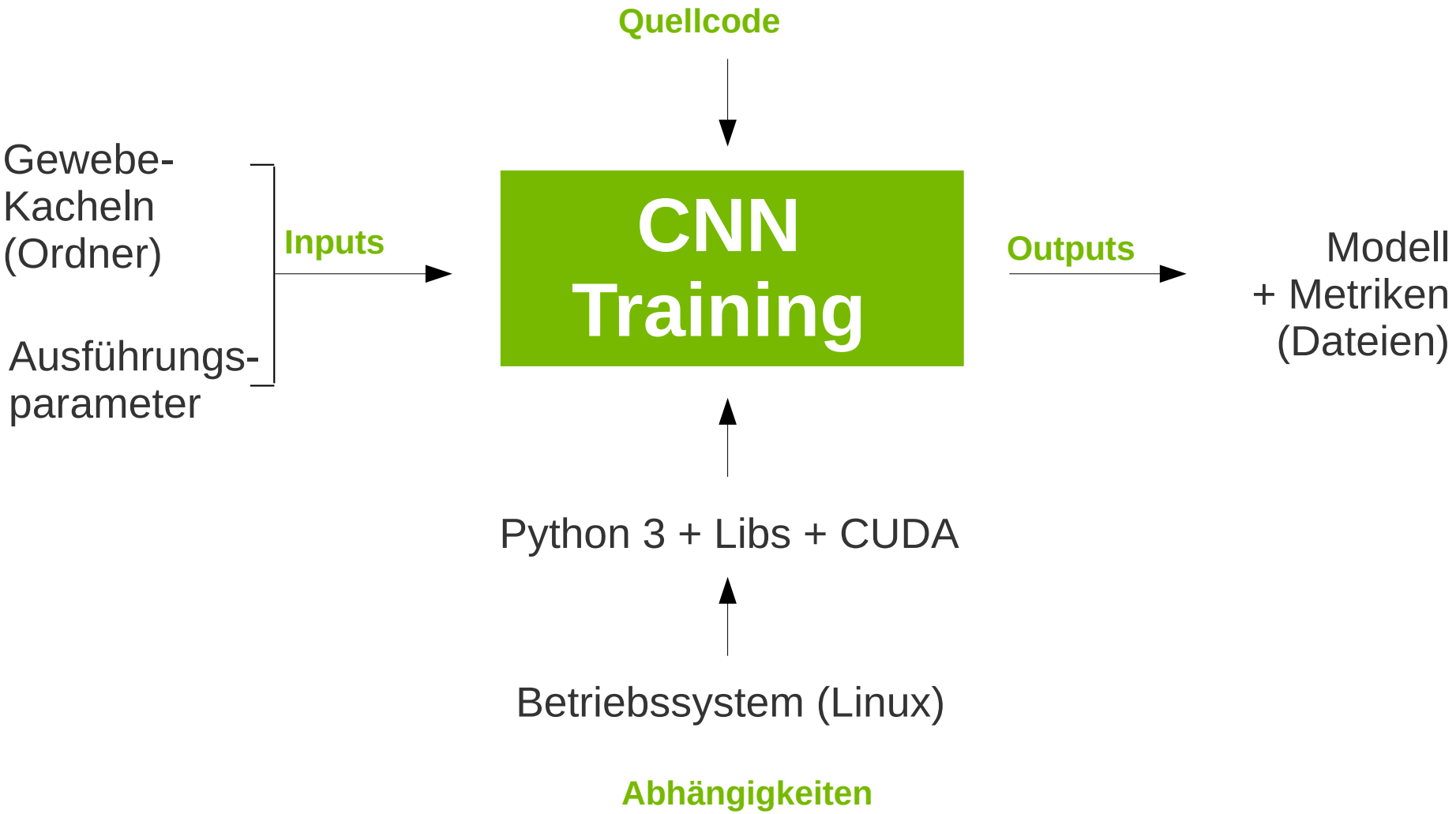


CNN Training

- Gewebekacheln aus WSI extrahieren (>1.5 TB)
- Benötigt GPUs zur Beschleunigung



Experiment Übersicht



FAIR Guiding Principles

Findable

Accessible

Interoperable

Reusable

Wilkinson et al. 2016. The FAIR Guiding Principles for scientific data management and stewardship. Scientific data 3, 160018

FAIR Guiding Principles

Findable → Globale, eindeutige IDs

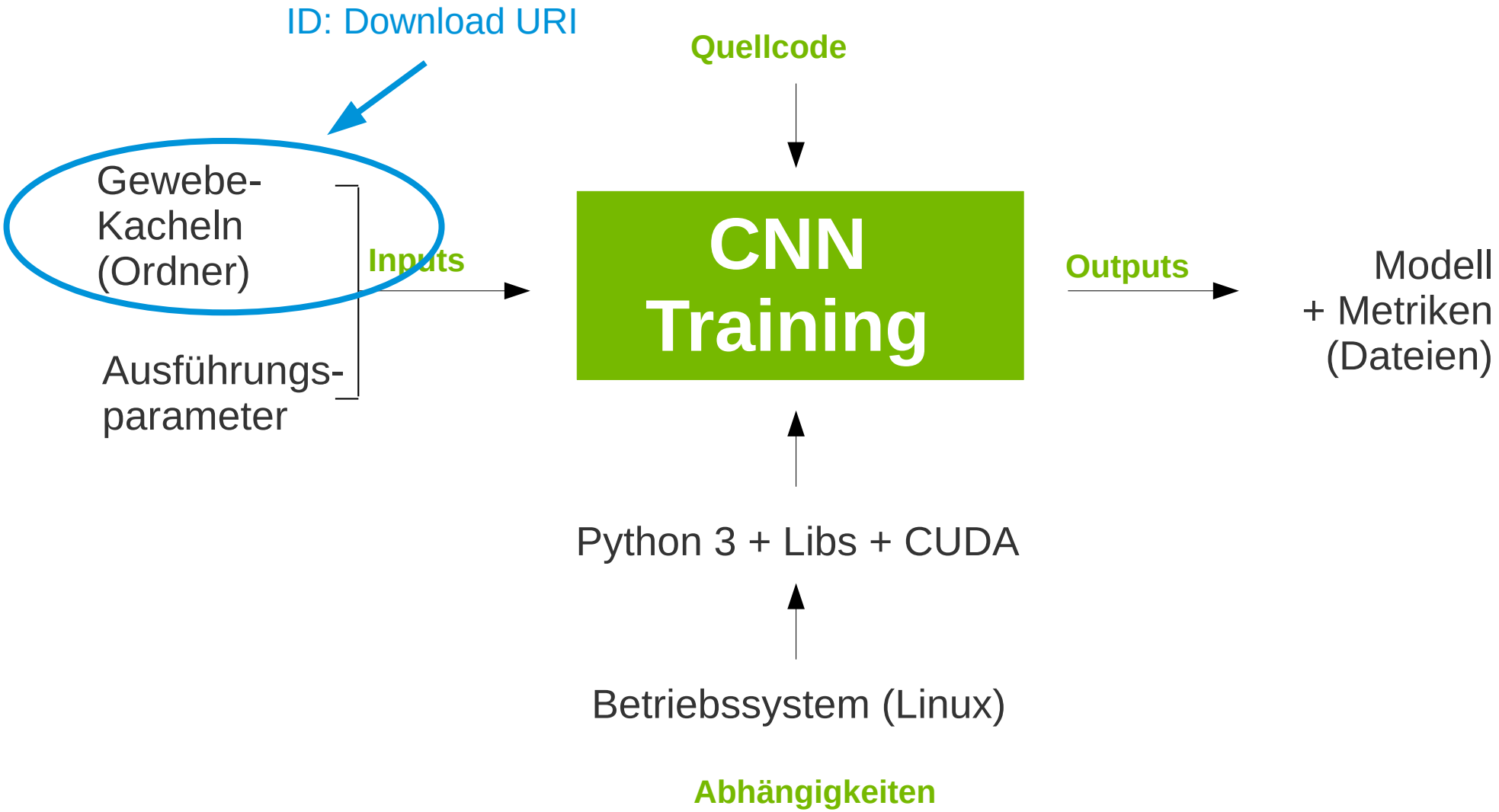
Accessible → Standards zur Übertragung / Auth.

Interoperable

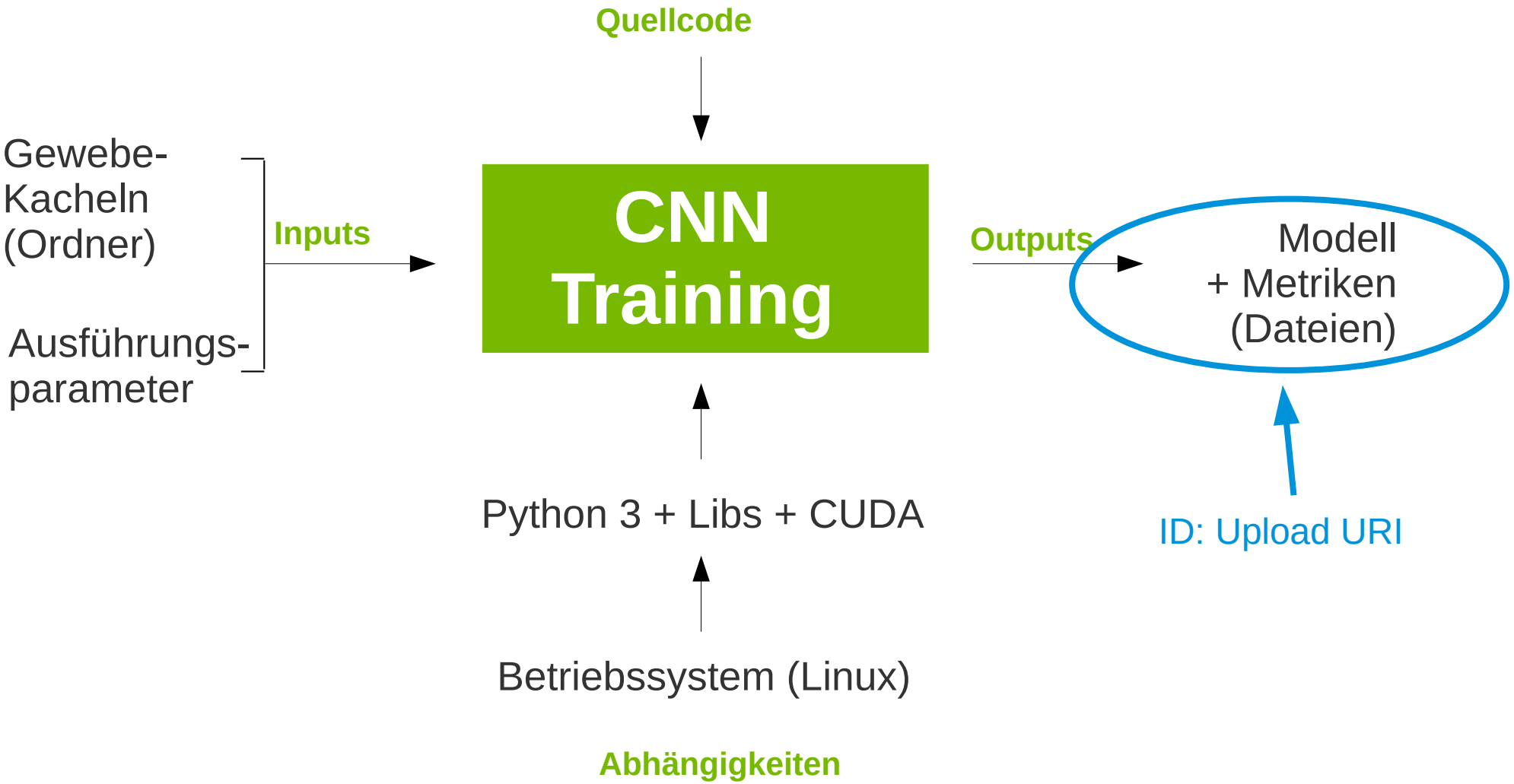
Reusable

Wilkinson et al. 2016. The FAIR Guiding Principles for scientific data management and stewardship. Scientific data 3, 160018

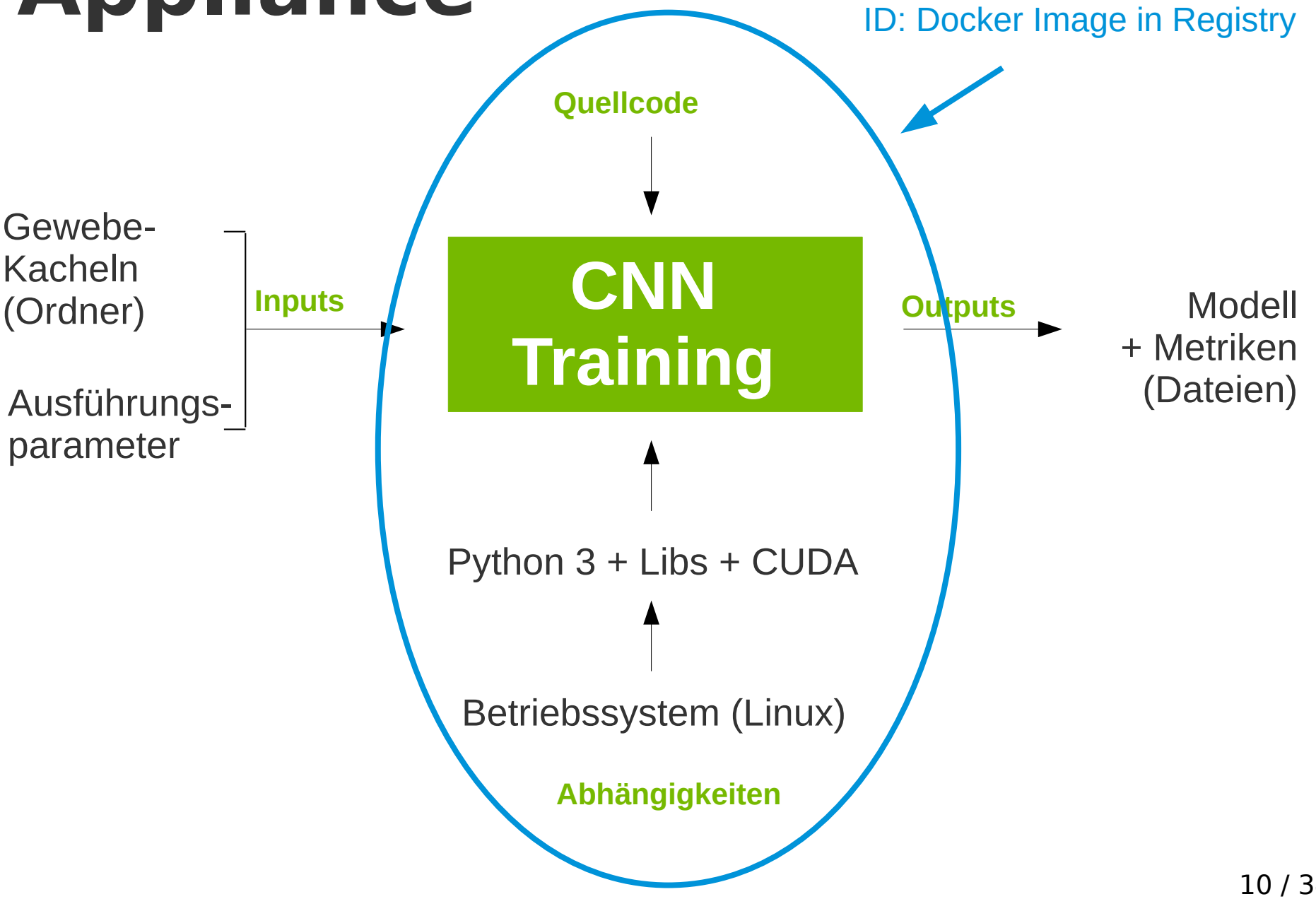
Inputs


ID: Download URI
Quellcode
Gewebe-
Kacheln
(Ordner)
Inputs
Ausführungs-
parameter
CNN
Training
Outputs
Modell
+ Metriken
(Dateien)
Python 3 + Libs + CUDA
Betriebssystem (Linux)
Abhängigkeiten

Outputs


Quellcode
Gewebe-
Kacheln
(Ordner)
Ausführungs-
parameter
Inputs
CNN
Training
Outputs
Modell
+ Metriken
(Dateien)
ID: Upload URI
Python 3 + Libs + CUDA
Betriebssystem (Linux)
Abhängigkeiten

Appliance


ID: Docker Image in Registry
Quellcode
Gewebe-
Kacheln
(Ordner)
Ausführungs-
parameter
Inputs
CNN
Training
Outputs
Modell
+ Metriken
(Dateien)
Python 3 + Libs + CUDA
Betriebssystem (Linux)
Abhängigkeiten

FAIR Guiding Principles

Findable

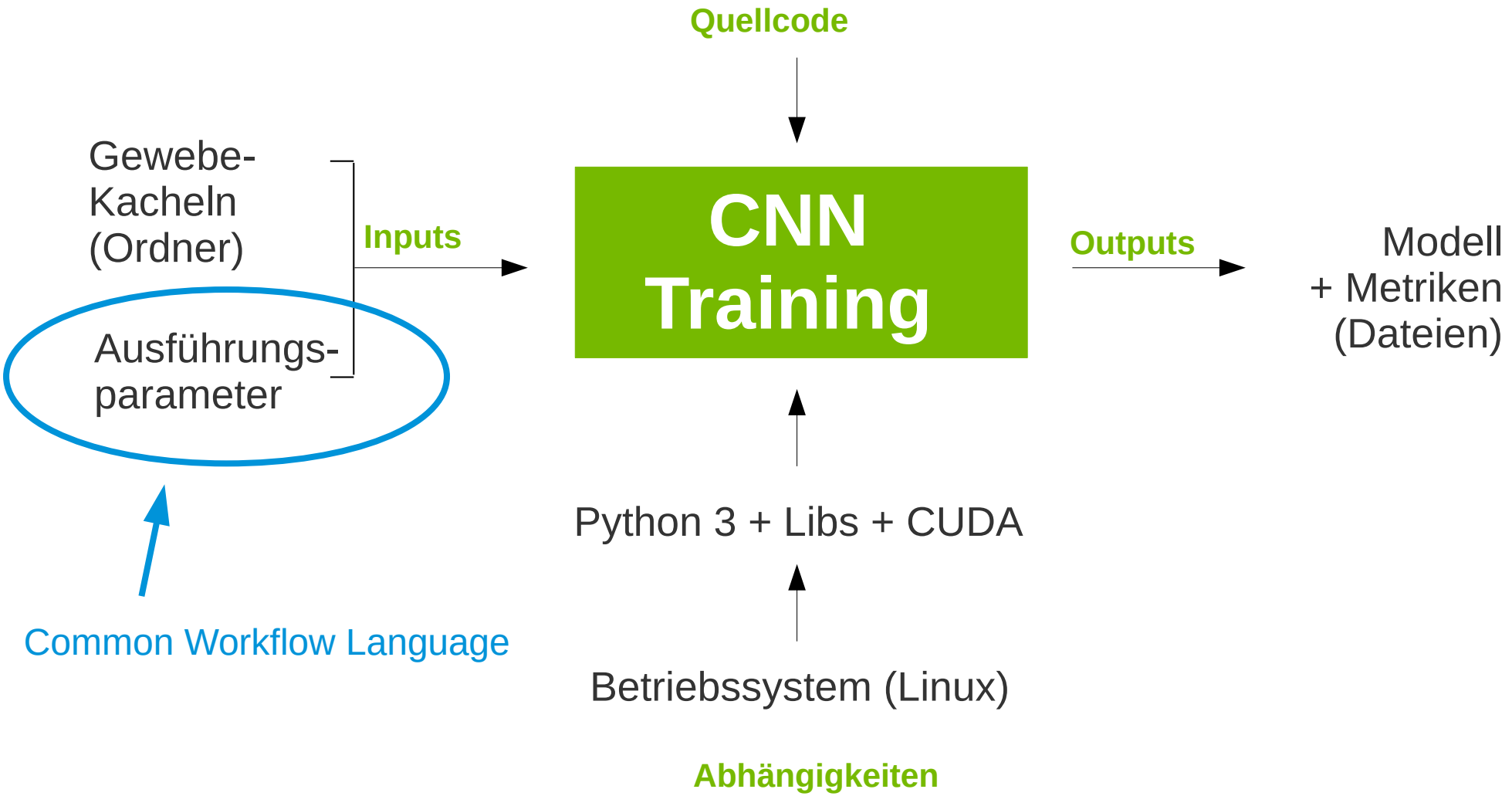
Accessible

Interoperable → Offene Dateiformate

Reusable → Community Standards folgen

Wilkinson et al. 2016. The FAIR Guiding Principles for scientific data management and stewardship. Scientific data 3, 160018

Experiment Übersicht



CWL Example

cnn-training.cwl

```
baseCommand: training.py
inputs:
  tissueTiles:
    type: Directory
    inputBinding:
      position: 0
outputs:
  model:
    type: File
    outputBinding:
      glob: model.hdf5
```

CWL Example

cnn-training.cwl

```
baseCommand: training.py
inputs:
  tissueTiles:
    type: Directory
    inputBinding:
      position: 0
outputs:
  model:
    type: File
    outputBinding:
      glob: model.hdf5
```

job.yml

```
tissueTiles:
  class: Directory
  location: /tiles.hdf5
```

CWL Example

cnn-training.cwl

```
baseCommand: training.py
inputs:
  tissueTiles:
    type: Directory
    inputBinding:
      position: 0
outputs:
  model:
    type: File
    outputBinding:
      glob: model.hdf5
```

job.yml

```
tissueTiles:
  class: Directory
  location: http://www...
```

CWL Example

cnn-training.cwl

```
baseCommand: training.py
inputs:
  tissueTiles:
    type: Directory
    inputBinding:
      position: 0
outputs:
  model:
    type: File
    outputBinding:
      glob: model.hdf5
```

job.yml

```
tissueTiles:
  class: Directory
  location: http://www...
```

Ziel 1: Format

RED Struktur (YAML)

```
redVersion: “7”
cli: ...                # CWL
inputs: ...             # Connectors
outputs: ...            # Connectors
container: ...          # Container Engine (Docker)
execution: ...          # RED Execution Engine
```

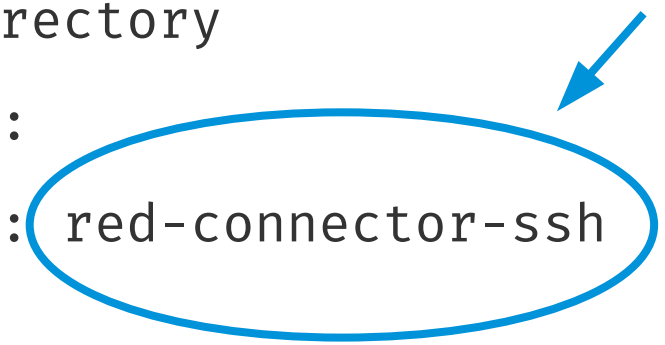
RED Connectors

```
inputs:
  tissueTiles:
    class: Directory
    connector:
      command: red-connector-ssh
      access:
        host: cbmi.htw-berlin.de
        auth:
          username: de-rse
          password: conf2019
        dirPath: /data/tiles
```

RED Connectors

```
inputs:
  tissueTiles:
    class: Directory
    connector:
      command: red-connector-ssh
      access:
        host: cbmi.htw-berlin.de
        auth:
          username: de-rse
          password: conf2019
        dirPath: /data/tiles
```

CLI Programm in Container Image



Teilen und Archivieren

```
inputs:
  tissueTiles:
    class: Directory
    connector:
      command: red-connector-ssh
      access:
        host: cbmi.htw-berlin.de
        auth:
          username: de-rse
          password: conf2019
        dirPath: /data/tiles
```

Teilen und Archivieren

```
inputs:
  tissueTiles:
    class: Directory
    connector:
      command: red-connector-ssh
      access:
        host: cbmi.htw-berlin.de
        auth:
          username: {{cbmi_username}}
          password: {{cbmi_password}}
        dirPath: /data/tiles
```

Default: Download

```
inputs:
  tissueTiles:
    class: Directory
    connector:
      command: red-connector-ssh
      access:
        host: cbmi.htw-berlin.de
        auth:
          username: {{cbmi_username}}
          password: {{cbmi_password}}
        dirPath: /data/tiles
```

Download 1.5 TB
into Container?

Mount / Stream via FUSE

SSHFS or HTTPDirFS

```
inputs:
  tissueTiles:
    class: Directory
    connector:
      command: red-connector-ssh
      mount: true
      access:
        host: cbmi.htw-berlin.de
        auth:
          username: {{cbmi_username}}
          password: {{cbmi_password}}
        dirPath: /data/tiles
```

Mount / Stream via FUSE

```
inputs:
  tissueTiles:
    class: Directory
    connector:
      command: red-connector-ssh
      mount: true
      access:
        host: cbmi.htw-berlin.de
        auth:
          username: {{cbmi_username}}
          password: {{cbmi_password}}
        dirPath: /data/tiles
```

Training via SSHFS vs. SSD:

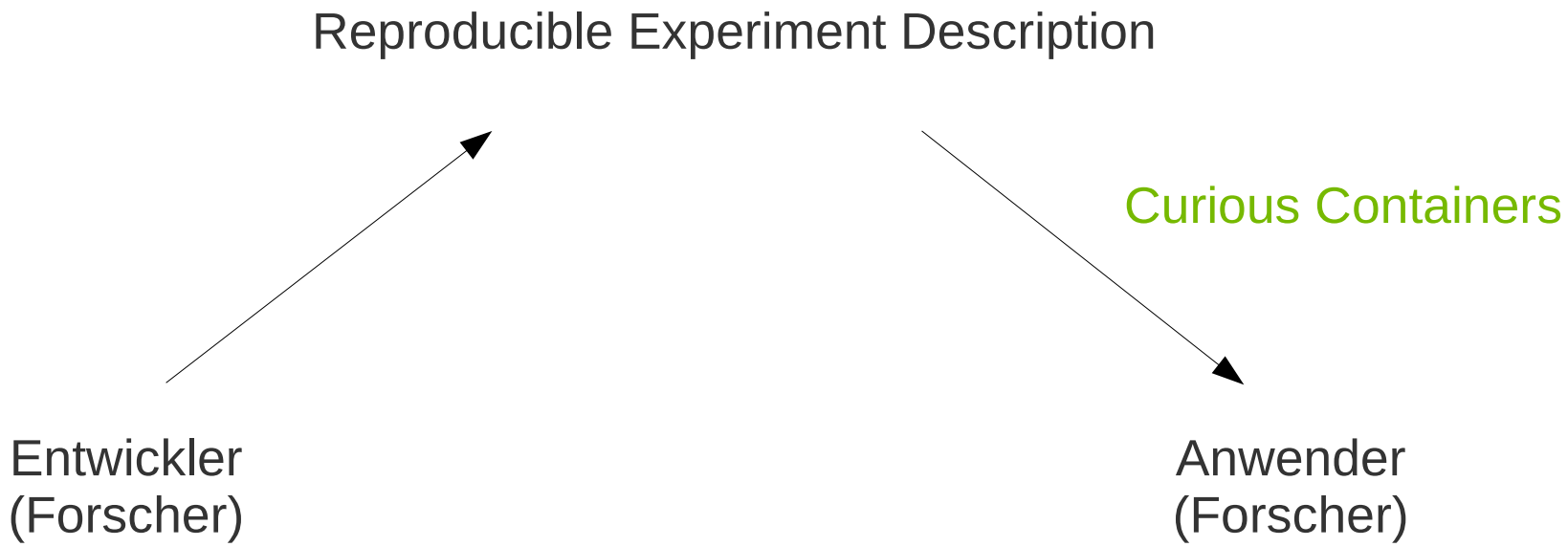
1,8 mal langsamer
über 2 x 10 Gbit Netzwerk

Nvidia-Docker Engine

Nvidia-Docker Engine

```
container:
  engine: nvidia-docker
  settings:
    image:
      url: docker.io/life/cnn-training
      auth:
        username: {{registry_username}}
        password: {{registry_password}}
    ram: 32768
    gpus:
      - minVram: 8192
      - minVram: 8192
```

Ziel 2: Automation

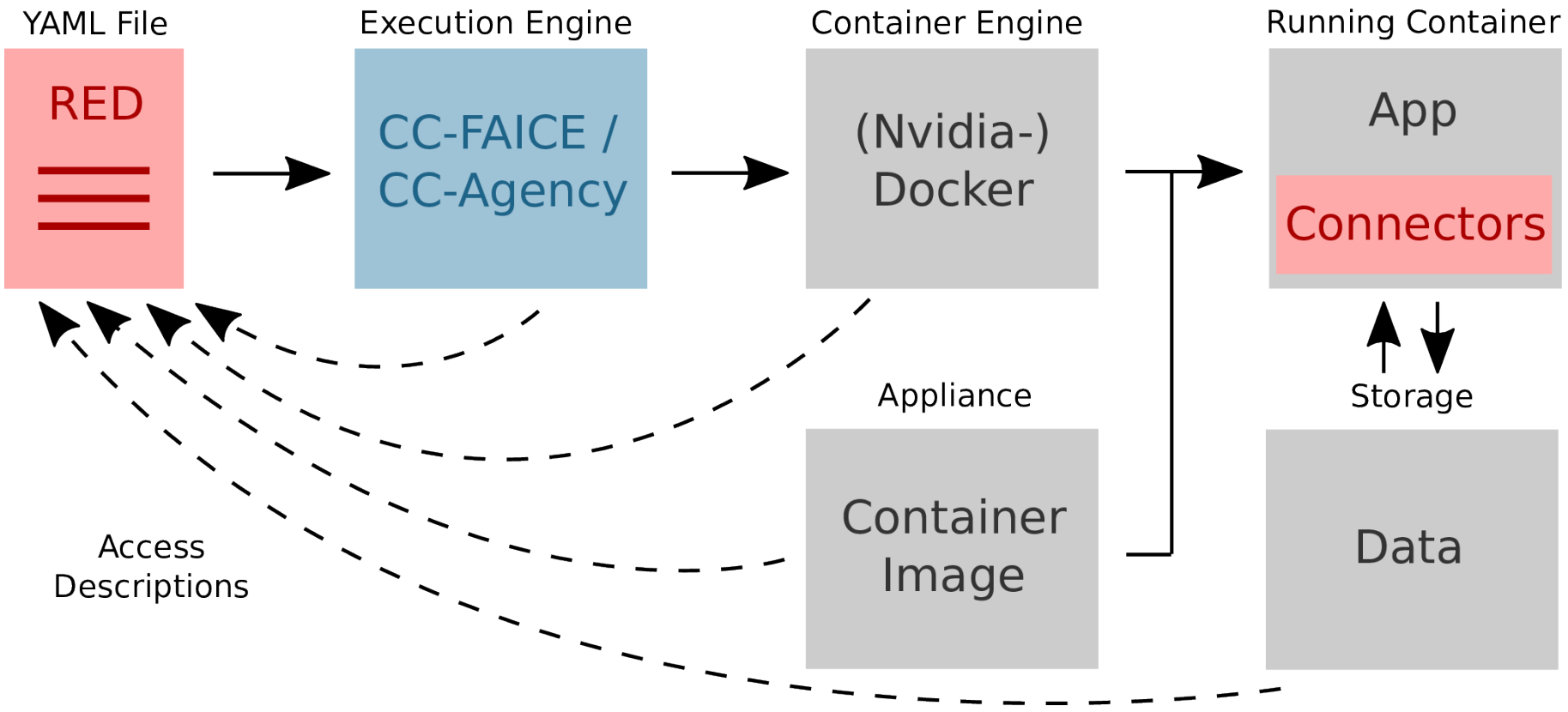


Curious Containers

RED Execution Engines

- CC-FAICE (FAIR Collaboration and Experiments)
  - Lokale Ausführung
  - Einfach zu installieren
- CC-Agency
  - Serverseitige Ausführung
  - Verbindet sich mit Docker-Cluster
  - Geplante (parallele) Ausführung
  - Unterstützt CPU und GPU Server innerhalb eines Clusters

Komponenten



Ziel 3: Publikationsprozess

Vorschlag: Öffentliche Git-Repositories (z.B. Github)

- Repo 1: Anwendung → Release
  - Lizenz nicht vergessen
- Repo 2: Dockerfile zum Bau der Appliance
- Docker-Registry: Appliance (Container-Image)
- Repo 3: RED Datei
- Zenodo: DOI für Repos (Optional)

Aufwand?

- (Noch) kein Tooling zum Generieren einer RED-Datei
- Vorbereitung der Komponenten im Nachgang mit relativ hohem Aufwand verbunden

- ABER: Curious Containers im Entwicklungsprozess bietet Vorteile
  - Testen verschiedener Konfiguration in Containern
  - Speichern von RED-Dateien für interne Dokumentation
  - Cluster-Computing → Parallele Experimente
  - Wenig Aufwand zur Veröffentlichung im Nachgang

Lizenz: AGPL-3.0

Dokumentation: https://www.curious-containers.cc

Code: https://github.com/curious-containers

Christoph.Jansen@htw-berlin.de

Lizenz dieser Präsentation

CC-BY-SA 4.0