Programming tasks BOX21

task 23 update bounding box action label using API

* Adapted authorizing **decorator** function to receive *‘bbox\_id’*.
* **Created endpoint** *'api/bbox/label/update'* and validated with unit test
* Connected to application programming interface (**API**) via Python **Request**
* Set remote PC (localhost) as **client** via AnyDesk
* Added *‘\_overlap’* to labels of overlapping bounding boxes
* Exported resulting dictionary to csv file.

label = add\_label\_if\_needed(label\_name=label\_name, project\_id=annotation.project\_id, category\_type=annotation.type)

Afbeelding met boom, gras, buiten, veld

Automatisch gegenereerde beschrijving 

task 22 DeepLabCut inference via API for bounding boxes without key-points

* Downloaded animal bounding boxes locally
* Downloaded **DeepLabCut model** locally
* **Looped** over each box. If not contained key-points yet, **inference** applied for key-point predictions.
* Import key-point predictions into BOX21 via **API**

*a = numpy.asarray(Image.open(directory + "/" + file\_name))  
dlc\_live.init\_inference(a)  
key\_points = dlc\_live.get\_pose(a)*

task 21 Train iteratively with different train and validation set combinations

* Created function transferring both object properties *‘AssetSet’* and *‘AssetSetAsset’* into arrays.
* **Registered** and added **blueprint** *‘asset\_set’*
* Created endpoints and validate by unit tests:
* *api/asset-sets/*
* *api/asset-sets/assets*
* *api/asset-sets/assets/add*
* *api/asset-sets/assets/delete*
* Applied **list comprehension** to get asset\_ids
* Use query.filter for objects in the **SQL** DB like .first(), .all() or .delete()

@asset\_sets.route("/api/asset-sets/", methods=["POST"])  
@allowed\_to\_view\_project  
def route\_asset\_sets():  
 project\_id = int(request.form.get('project\_id', 1))  
 asset\_sets = AssetSet.query.filter(project\_id == project\_id).all()  
 return jsonify(asset\_sets\_as\_array(asset\_sets))

task 20 Identify overlapping bounding boxes

* **Build** hierarchical **dictionary** with images, bounding boxes and coordinates.
* Created function ‘*box\_overlaps\_other\_boxes’* checking if bounding box overlaps with other bounding boxes.
* Applied function to detect all overlapping boxes in dataset. Followed up by task 23.

def box\_overlaps\_other\_boxes(box, other\_boxes, counter):  
 check\_condition\_box\_with\_other\_boxes = []  
 for i in range(len(other\_boxes)):  
 if i is not counter:  
 check\_condition = box\_overlaps\_box(box, other\_boxes[i])  
 check\_condition\_box\_with\_other\_boxes.append(check\_condition)  
 return check\_condition\_box\_with\_other\_boxes

task 19 Created endpoints add keypoint and add bounding box

* Registered and added blueprint *‘annotation’*
* **Created endpoints** and validated by unit tests:
  + *'/api/asset/keypoints/add'*
  + *'/api/asset/boundingboxes/add'*
* Added authorizing **decorator**
* Used *‘controllers’* to clean up the blueprint endpoints
* **Refactor** duplicate code into function stored directory *‘packages’*
* Use terminal to check unit tests and check logs for error handling endpoint
* **Error handling**: specifying parameters, JSON serialisation , 500 error

task 18 Create function get predecessors

* Created function *‘get\_ predecessors’* that returns a list of all parents and parents of the parents to show its hierarchy. For example ‘Browsing’ is a type of ‘Foraging’.
* Create a unit test and executed by **CMD command prompt**.

task 17 Write unit tests for asset endpoints

Improved code by writing **unit tests** for 10 endpoints

'/api/asset/favourite', '/api/asset/annotations', '/api/asset/meta/update-value', '/api/asset/meta/delete-key', '/api/mark-bboxes-validated', '/api/assets/validate', '/api/assets/invalidate', '/api/assets/delete', '/api/add-to-validation-set', '/api/remove-from-validation-set'

* Created new project and added asset belonging to project.
* **Commit** to SQL DB.
* Specify parameters. **Serialized** using dumps() and loads()
* Start session and inspect **response**.

response = session.post('http://localhost:8008/api/asset/favourite', data=params)  
assert(len(response.json()) == 11)  
assert(response.json()['liked'] == True)

task 16 Allowed for train and validation split

* **Created** **lists** train\_asset\_ids = [] and val\_asset\_ids = [].
* **Parsed** object ‘annotation’ and obtained [‘in\_validation\_set’]. Assigned that image to either of the lists
* **Specified** **arguments** *trainIndices* and *testIndices* in deeplabcut.create\_training\_dataset()

deeplabcut.create\_training\_dataset(config\_path, net\_type=self.model\_type, trainIndices = ids, testIndices = val\_asset\_ids, augmenter\_type='imgaug')

task 15 Added string with grouping values to meta-string

* Used **client** [*https://box21.ai*](https://box21.ai) to access BOX21 via API.
* **Build dictionary** with images, filename, label occurrences and export to csvAfbeelding met tafel

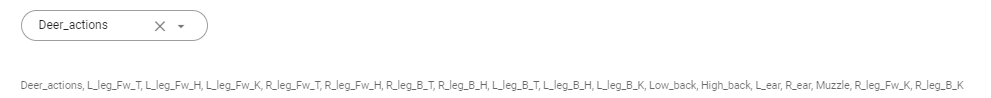
  Automatisch gegenereerde beschrijving
* **Added attribute** *‘grouping\_test’* with grouping values and imported back into BOX 21.

box21\_api.update\_asset\_meta(asset.id, 'grouping\_test', asset\_annotation['grouping\_test'])

{"filename": "8a269d42-0a40-4633-bf07-eb8a7e5f0f22.JPG"}

{"filename": "8a269d42-0a40-4633-bf07-eb8a7e5f0f22.JPG", "grouping\_test": "1\_a\_a"}

task 14 Added crop\_label label\_id to label\_ids



**2x**

* As in *key-points training configuration* bounding bboxes (*Deer\_actions*) occurred twice, the first *deer\_actions* was added to the list of keypoints (*label\_ids*) without showing to the user.
* Endpoints were created connecting different repositories: *The training repository* and *box21\_api*.

label\_ids.append(crop\_label\_id)

task 13 Build key-point inference functionality for webpage BOX21

* Created function *handle\_deeplabcut()*
* Created endpoint [*http://aiworker\_deeplabcut\_0.30/run*](http://aiworker_deeplabcut_0.30/run)
* Used existing code for YOLOv5 to adapt to DLC

task 12 Tested DeepLabCut inference locally

* Used **FTP** to download most recent model DeepLabCut from **GPU PC** to local PC
* Exported model
* Apply DeepLabCut-live (inference)

from deeplabcut.pose\_estimation\_tensorflow.export import export\_model

config\_path = self.run\_path / 'deeplabcut\_model' / 'config.yaml'

deeplabcut.evaluate\_network(config\_path, plotting=True)

export\_model(config\_path, TFGPUinference=False)

from dlclive import DLCLive, Processor  
import numpy  
from PIL import Image

dlc\_live = DLCLive('C:/Users/DLC\_inference/deeplabcut\_model/exported-models/Deer\_iteration-0\_shuffle-1', processor=Processor())  
a = numpy.asarray(Image.open('C:/Users/DLC\_inference/00add8fd-bd07-4d43-a11d-ae8c30808c7e.jpg'))  
dlc\_live.init\_inference(a)  
dlc\_live.get\_pose(a)

task 11 Added filter option ‘Last edited by’

* Added a filter option to BOX21 named ‘Last edited by’:
* Created a unit test
* Implemented functionality to webpage

{'type': 1, 'label': 'Last edited by'}

Afbeelding met tekst

Automatisch gegenereerde beschrijving

task 10 Get size directory

* **Searched the internet** for functions that get the size of a directory
* **Convert bytes** (B) to KB, MB and to GB (/1024)

def get\_size(start\_path = '.'):  
 total\_size = 0  
 for dirpath, dirnames, filenames in os.walk(start\_path):  
 for f in filenames:  
 fp = os.path.join(dirpath, f)  
 if not os.path.islink(fp):  
 total\_size += os.path.getsize(fp)  
 return total\_size

task 9 Added type annotations to functions

* Added **type annotations** to functions

def example() -> Object:

-> {}:, -> []:, -> str:, -> None:, -> bool:, -> int:

def route\_add\_annotation\_group(current\_user: User) -> str:

task 8 Added unit test for unclear API endpoint

* **Created unit test** checking if *‘unclear’* endpoint changed asset propertyfrom *False* to *True*

response = s.post('http://localhost:8008/api/assets/mark-unclear' , data=params)

asset = response.json()

assert (asset['unclear'] == True)

task 7 Added unit tests for setting endpoints

* **Added unit tests** for setting endpoint.
* **Defined** ‘Setting’ **object** **property** ‘points enabled’ and value ‘false’ to the DB and tested if the endpoint returned both parameters.
* Ran unit test in CMD command prompt.
* Worked with **objects** and **dictionaries**
* Ensured connection *http://localhost:8008* using Anydesk

assert (setting['name'] == 'points\_enabled')  
assert (setting['value'] == 'true')

**Other tasks:**

task 6 Delete job endpoint

task 5 Test and refactoring

task 4 Test two additional functions in labels.py

task 3 Endpoint child of

task 2 Test child of

task 1 Add label if needed