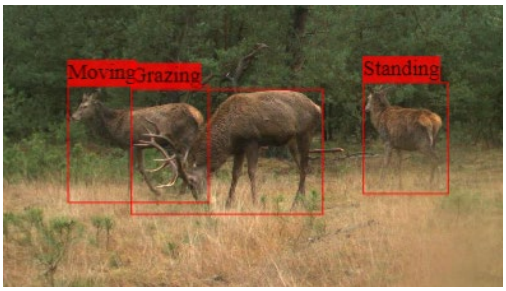
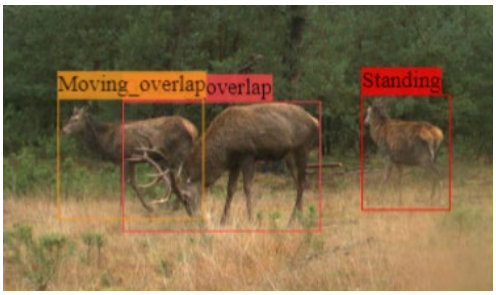
Programming tasks BOX21

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)
```





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* to access BOX21 via API.
- **Build dictionary** with images, filename, label occurrences and export to csv

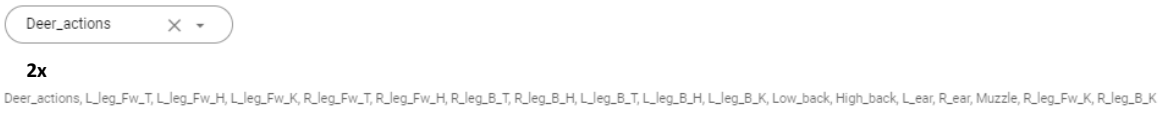
	A	B	C	D	E	F
1	asset_id	filename	Standing	Moving	Grazing	grouping_test
2						
3	160518	8a269d42-0a40-4633-bf07-eb8a7e5f0f22.JPG	1	0	2	1_a_a
4						
5	160739	3d4be7b6-1e3a-493f-8fb0-084b74251822.JPG	3	1	12	1_a_b
6						
7	160666	9104c029-a85a-44d3-9e5d-cf92e97e597e.JPG	0	1	1	2_b_b

- **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



- 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*
- 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)

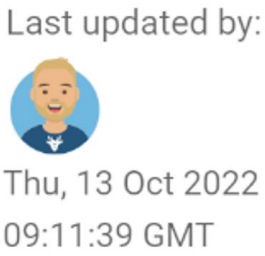
```
from deeplabcut.pose_estimation_tensorflow.export import export_model
config_path = self.run_path / 'deeplabcut_model' / 'config.yaml'
deeplabcut.evaluate_network(config_path, plotting=True)
export_model(config_path, TFGPUinference=False)
```

```
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'}
```



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 property from *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