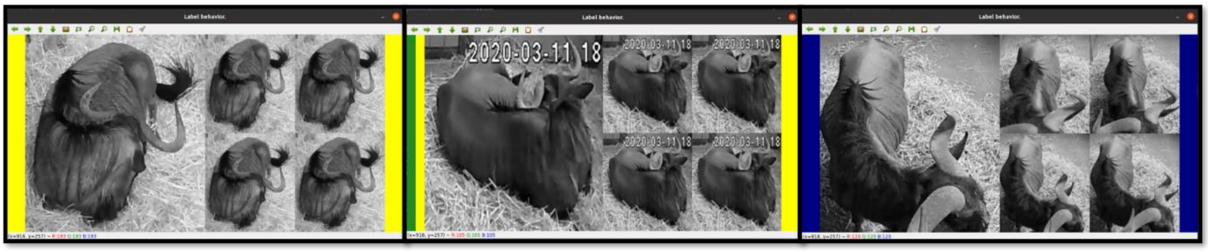
Short description

Script containing a GUI used for evaluating how accurate the action classifiers of BOVIDS work. Is used to mine hard examples and to re-train the action classifiers based on manual labels. The user evaluation is saved in the same .csv file as created by "hard_example_mining_efficientnet.py" and can be re-read by the script permanently.

Furthermore, the script can be used to copy the manually annotated images from FINAL_PREDICTION_CUTOUT into a new folder based on the manually given label to finetune a network.

User can give four labels (standing, lying LHU, sleeping LHD and 'error'). Error can be used if either the object detector drew a bounding box around a false object or if the user cannot reliably infer the label from the given image material.



Color codes exist for each activity class, on the l.h.s. the label by the single frame classifier can be found and on the r.h.s. the one by the multiple frame classifier. The inner color bars are the label given by the user.

Requirements

- packages: numpy, pandas, scikit-learn, opencv
- files generated by "hard_example_mining_efficientnet.py"
- FINAL_PREDICTION_CUTOUT (see readme.md)

Step 1 – open spyder:

- Terminal / Shell:
  - conda activate bovids
  - spyder

Step 2: adjust parameters:

- FOLDERPATH_IMAGES = Path (folder) to the output of „hard_example_mining_efficientnet.py", contains a subfolder for each individual. [string]
    - E.g.: '…/ExampleMining/'
- FOLDERPATH_CSV = Path (folder) to the output of „hard_example_mining_efficientnet.py", contains a csv-file for each individual. Is, normally, the same as FOLDERPATH_IMAGES. [string]
- INDIVIDUAL_CODE_LIST = List of individual code of the individual that will be evaluated. [string]
    - E.g.: ['Eland_FancyZoo_2', 'Eland_FancyZoo_4']
- DEST_IMAGES = Destination (folder) in which the manually annotated images will be copied to. [string]
- KI_CUTOUT = Path to the final storage of the cutout images, FINAL_STORAGE_CUTOUT. [string]
- START_INDEX = If script shall not start displaying the first image, set it to a different number as 0. [integer]

Control elements:

- MOVE_LEFT = '4'
- MOVE_RIGHT = '5'
- END_EVALUATION = 'p'

- SET_STANDING = 'w'
- SET_LYING = 'a'
- SET_SLEEPING = 's'
- SET_ERROR = 'f'

Further configuration:

- NEXT_IMAGE_IF_KEY: If True, the subsequent image will be shown if a label is given. Otherwise, the user needs to change the image manually (MOVE_RIGHT).
- DISPLAY_WIDTH = Size of the image displayed (px) [integer]

- COLORS = dictionary such that keys are the action classes and the corresponding values indicate which color should be used on GUI, BGR code. [dictionary]

Step 3 – run the script

- Start labelling: ***evaluate()***
- Show important key statistics: ***stats()***
- Copy images from KI_CUTOUT to DEST_IMAGES by manual label: ***move()***