Short description

Contains various small functions to extract images from the structure induced by "generate_training_images.py" to a training set for single frame and multiple frame action classifiers. Furthermore, provides the functionality to finetune this training set and to extract images from previous training sets. Finally, it can be used to merge action classes (e.g. LH-U and LH-D to lying).

Please notice that a **training set** is a folder trainset/ such that trainset/i/ consists of those images having label i. Normally, we have four different training sets: Single Frame and multiple frame as well as binary classification and total classification.

Step 1 – open spyder:

- Terminal / shell:
    - conda activate bovids
    - spyder

Step 2 – adjust parameters:

Task 1: extract images from a previous training set

- SUBSET_FOLDER = Path to an old training set, contains the subfolders 0/ and 1/ (binary classification) or, respectively, 0/, 1/, 2/ (total classification) [string]
- SUBSET_OUT = Destination (folder). Subfolders for each class will be created. [string]
- CLASS_SIZES_PER_IND = Select how many images per class and per individual will be randomly sampled. [dictionary]

Task 2: merge datasets

- INPUT_FOLDERS_JD = List of strings such that each entry is a path to a dataset (containing folders 0/, 1/, 2/). [list]
- OUTPUT_FOLDER_JD = Destination (folder) in which the subfolders 0/, 1/, 2/ will be created containing all images of the correct class from the input. [string]
- NESTED: [boolean]
    - True: All paths in INPUT_FOLDERS_JD contains subfolders that are themselves datasets rather than being a dataset themselves. This is, for instance, created by BOVIDS' OHEM.
    - False: All paths in INPUT_FOLDERS_JD are datasets, thus contain 0/, 1/, 2/.

- Keep in mind that this procedure needs normally to conducted for multiple frame and single frame independently. If additional classes are present (like 99 in OHEM), just delete them in the output folder.

Task 3: Random Upsampling

- UPSAMPLING_FOLDER = Path to a folder full of image files (.jpg). (e.g. …/MF/0/) [string]
- TARGET_SIZE = Target number of images in this folder. [integer]
- As it is suggested to train on balanced classes, it is possible to randomly sample images from an underrepresented class and duplicate them. We highlight that random upsampling is not the best nor unique possibility to deal with unbalanced data but a fairly comfortable one and we refer the reader to standard literature on deep learning.

Task 4: Produce a training set for binary classification

- INPUT_FOLDER_BD = Path (folder) to one total training set containing subfolders 0,1,2 [string]
- OUTPUT_FOLDER_BD = Destination (folder) in which subfolders 0 and 1 will be created. [string]
- BEHAVIOR_MAPPING = Dictionary describing which percentage of which class is mapped onto which target class. [dictionary]
    - E.g: {‚0‘: {‚0‘:1.0, ‚1‘: 0.0 }}: 100 % of the images of class 0 will be copied to class 0 and none of the images of class 0 will be copied to class 1.
- Keep in mind that this procedure needs normally to conducted for multiple frame and single frame independently.

Task 5: Validation Split

- VAL_SPLIT = Percentage of images that will be used as validation set. [float]
- INPUT_FOLDER_VS = Path (folder) of the training set (contains subfolders 0,1,2 or, respectively, 0,1) [string]
- OUTPUT_FOLDER_VS = Destination (folder) into which the validation images will be moved. Subfolders will be created according to the classes. [string]

- BEHAVIOR_CLASSES = List of strings indicating which classes should be used for validation. Normally, in total classification, this is [‚0', ‚1', ‚2'] and in binary [‚0', ‚1'] [list of strings]
- Keep in mind that this procedure needs normally to conducted for multiple frame and single frame independently.

Step 3 – run the script

- Extract images from previous training sets: *get_random_subset()*
- Merge datasets: *join_datasets()*
- Random Upsampling: *random_upsampling()*
- Produce a training set for binary classification: *merge_classes()*
- Validation Split: *validation_split()*