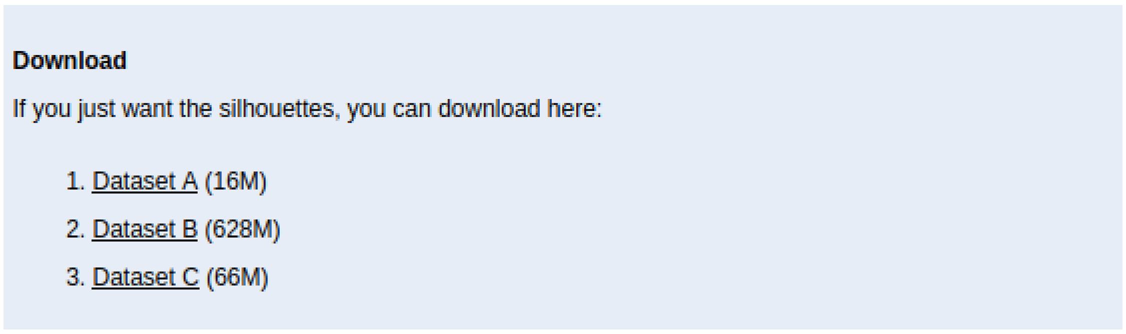
Lab 11 & 12 - Gait Recognition

Classical Gait Recognition

For gait recognition problem we will use CASIA Gait Database:
http://www.cbsr.ia.ac.cn/english/Gait%20Databases.asp

Please download “Dataset B” from the “Download” section:



Download

If you just want the silhouettes, you can download here:

1. Dataset A (16M)
2. Dataset B (628M)
3. Dataset C (66M)

You can also download dataset from here:
https://drive.google.com/file/d/1ZAmf5SI5EaHZ2Nha-Q9zTbGtfIXvQncO/view?usp=sharing

It consists of extracted human silhouettes extracted while the person was moving in different directions relative to the camera (“view angle”). After extracting the whole dataset, you will see the following folder structure:

```
person id
        |- walking status
                |- view angle
                        |- person_id-walking_status-view_angle-number.png
```

person id - number (001 - 124)
walking status - xx-nn, where “xx” determines if person walks normally (nm), with a bag (bg) or in a coat (cl) and “nn” is the ordinal number
view angle - number from 000 to 180 with a step 18

Each image is in fact binarized and has a structure of a mask: white pixels - human, black pixels - background.

Classical approach overview:

- extract GEI (Gait Energy Image) for each sequence
- train Random Forest Classifier based on flatten GEI (vector)
- validate the training results

More detailed explanation:

1. First of all we need a function to extract GEI. We will use lists to collect data from all images in one sequence (i.e. from specific `person_id/walking_status/view_angle/` path).
   For each image:
   a. load image - we can use cv2.imread method with the `cv2.IMREAD_GRAYSCALE` flag to get gray image straightforward
   b. binarize each image (cv2.threshold with some positive threshold, e.g. 127)
   c. find bounding box of the person (one and only object in the image) - cv2.boundingRect
      Note: The main idea is to extract a bounding box of person without any additional objects. If this is not possible for various reasons (no person or no suitable criterion to extract it), we can simply omit the frame and not consider it in the GEI.
   d. extract the image part inside this bounding box (i.e. extract person)
   e. resize extracted image to some predetermined height and width - cv2.resize with size equals e.g. (70, 210)
   f. change array type to float and append it to list
2. When data from all images in one sequence are gathered in the list, we can compute the GEI - mean pixels values among all images:
   $$GEI(x, y) = \frac{1}{N}\sum_{t=1}^{N} I(x, y, t)$$
   where: N - number of images in one sequence, I(x, y, t) - pixel (x, y) in the t-th image of a sequence.
   To compute GEI we can use the np.mean method along the appropriate axis.
3. It will be convenient to convert operations from point 1 and 2 into function (*path_to_sequence* and *target_size* as inputs).
4. To train our classifier we need a significant amount of data. So, we can do it this way: choose a specific angle view (e.g. 90 - it should be the easiest one) and do training and evaluation only for the chosen angle view. To split sequences into train and val sets we can use walking status - e.g. train sequences are this with walking status: ['nm-03', 'bg-01', 'nm-06', 'nm-04', 'cl-01']

and val sequences are this with walking status:
['nm-02'].

5. Extract GEI for each person and for each training walking status with chosen angle view. Flatten the output to have a 1-D vector and save it together with person id (we need reference output to train our classifier).
6. Now we can train our classifier - use RandomForestClassifier from scikit-learn with following arguments:
   ```
   RandomForestClassifier(n_estimators=500, n_jobs=-1,
   random_state=2023, verbose=1, max_depth=100, max_features=100)
   ```
   Perform fit() method on extracted data.
7. When the classifier is ready, we can perform the validation part. Like before, extract GEI for each person and for each validation walking status with chosen angle view. Flatten and save output together with person id (to check the classifier output).
8. Use predict() method and compare outputs with the reference. Count good predictions and calculate classifier score - good predictions in relation to the number of validation sequences.
9. Do some experiments (not less than 3 in total) with different angle views and walking status (also with number of walking status used in train set) and compare obtained scores (write them as a comment in your script).