# Instructions for training/testing of your designs:

## Dataset:

CUHK03 is a very popular dataset containing 13,164 images of 1,360 pedestrians. Sample images from this dataset are presented below:

For the purpose of this coursework you are provided with both images from the dataset and feature vectors extracted from the images. Images in this dataset were captured with use of two cameras, what is more, each person was photographed a number of times. Therefore, each identity (person) is represented in the dataset approximately 8-9 times. All the data you need is contained in **CW2_data.zip**. Please unpack it, and you should find the following contents:

1. Folder **images_cuhk03** contains all the images from the dataset. Their names may seem illogical at first, but they are properly annotated in the file **cuhk03_new_protocol_config_labeled.mat**.
2. Matlab file **cuhk03_new_protocol_config_labeled.mat.** Once you open this file with Matlab or Python, you will find 6 main components (in the form of arrays):

a. **camId** specifies whether image was taken from camera 1 or camera 2. While evaluating (testing) your algorithms, you should not consider images of your current query identity taken from the same camera. For example, when you create ranklist for the first query image (index 22, label 3, camera 1, name "1_003_1_02.png"), you should not include images with indexes 21, 23, 24 in this ranking list, as these are images of the same person (label 3) captured by the camera with index 1.

b. **filelist** specifies correspondences between names of files in **images_cuhk03** and their indexes.

c. **gallery_idx**, which specifies indexes of the part of the dataset from which you compose your ranklists during testing phase

d. **labels** contains ground truths for each image

e. **query_idx** contains indexes of query images

f. **train_idx** contains indexes of images that can be used for training and validation

To load **cuhk03_new_protocol_config_labeled.mat** into Python you can use the code from this example (and repeat for other arrays listed above):

```
train_idxs = loadmat('cuhk03_new_protocol_config_labeled.mat')
                ['train_idx'].flatten()
```

with Matlab you can simply use *load('cuhk03_new_protocol_config_labeled.mat')* or double click on the file in File Explorer.

Apart from two files mentioned above, file **feature_data.json** contains an array 14096x2048 with some feature vectors representing each image in the dataset. **YOU SHOULD PERFORM YOUR COMPUTATIONS WITH USE OF FEATURE VECTORS PROVIDED, IMAGES ARE THERE ONLY FOR THE REFERENCE AND VISUALISATION**. With Pyhon you can simply load it as follows:

```
import json
with open('feature_data.json', 'r') as f:
   features = json.load(f)
```

and with Matlab:

```
features = jsondecode(fileread('feature_data.json'));
```

Please note it takes a while to load a file as it contains considerably large amount of data.

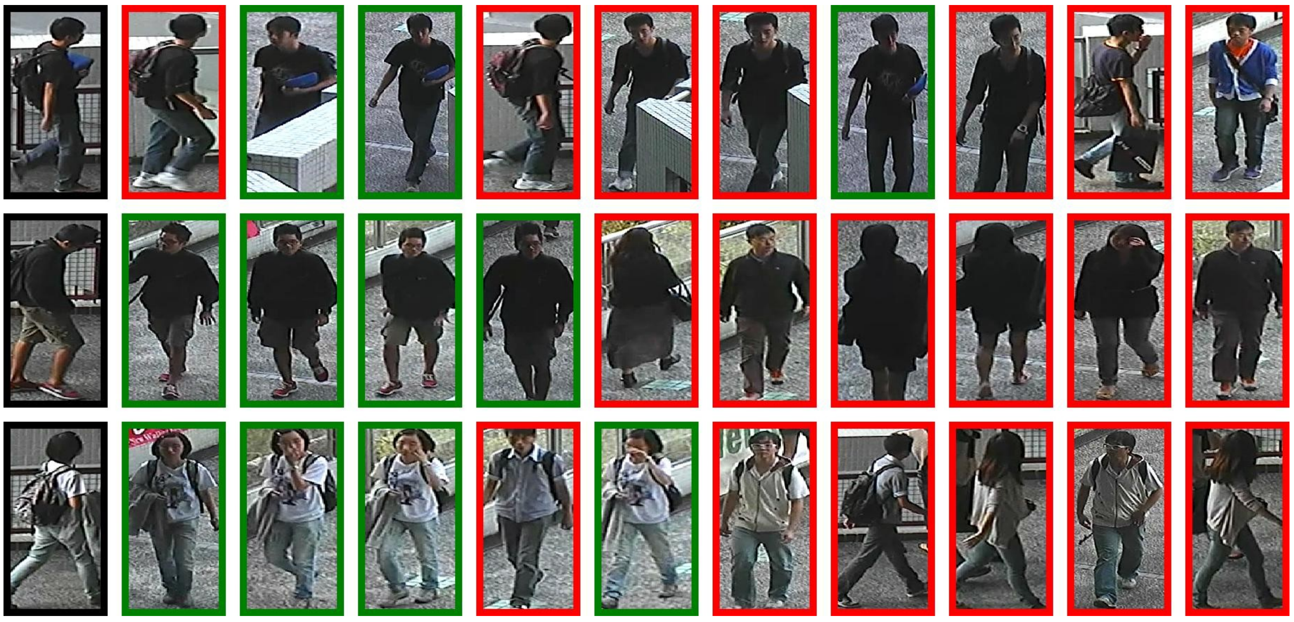
## Training, validation, testing:

**1. Training:**
For the purpose of training use feature vectors provided in the file **feature_data.json**. Indexes of vectors that can be used for training are stored in the **train_idx** vector belonging to **cuhk03_new_protocol_config_labeled.mat**.

**2. Validation:**
You can use 100 randomly selected identities from training set as validation set. Please note each identity is represented in the dataset 7-10 times (there is a number of photos/feature vectors of each person in the dataset), and you should include all of them in the validation set, therefore around 700-1000 different feature vectors will be used for validation. There exists an idea in Computer Vision that you use validation set only to specify the number of iterations that is optimal for your design and then you include your validation set into train set and perform final training (without validation set) for this fixed amount of iterations. You can try this idea as well. Once again, use feature vectors provided in the file **feature_data.json**. Indexes of vectors that can be used for training/validation are stored in the **train_idx**.

**3. Testing:**
Finally, you perform testing. Both **query_idx** and **gallery_idx** specify your test set. The task is to take all the feature vectors from query set, one by one, and create a ranklist for each of them, by finding nearest neighbours within **gallery_idx**, deleting images/feature vectors of considered query identity, captured by the same camera (same **label** and same **camId**). Sample ranklists after

deletion, with use of dataset images for visualisation (green frame indicates positive matches, red are negative ones, black are query images, you can see positive matches presenting different views of the query identity, due to deletion mentioned above):



Once having your ranklists you can calculate @rank1, etc. scores for such ranklists, achievable scores for simple nearest neighbour and Euclidean distance are:
top1:0.469286 top5:0.668571 top10:0.750000 mAP:0.432923 (I managed to do a bit better with simple NN on Matlab - @rank1 = 0.47, so don’t worry if you achieve a bit better scores).

**Please do not mix sub-datasets:**
**train_idx**: training, validation (optional)
**query_idx** + **gallery_idx**: only for testing your design, follow the procedure above

## **References**:

W. Li, R. Zhao, T. Xiao, and X. Wang, “Deepreid: Deep filter pairing neural network for person re-identification,” in 2014 IEEE Conference on Computer Vision and Pattern Recognition, June 2014, pp. 152–159.

Z. Zhong, L. Zheng, D. Cao, and S. Li, “Re-ranking person re-identification with k-reciprocal encoding,” CoRR, vol. abs/1701.08398, 2017. [Online]. Available: http://arxiv.org/abs/1701.08398

CUHK03 dataset:
http://www.ee.cuhk.edu.hk/~xgwang/CUHK_identification.html
https://github.com/zhunzhong07/person-re-ranking/tree/master/CUHK03-NP