September 14, 2019

A Dependencies

The script has been written in Python 3 using the distribution Anaconda 3. The following libraries are needed to run the script: TensorFlow 1.3.0, keras 2.0.8, skimage 0.12.3, astropy 1.3 and PIL 4.0.0.

B Files

The git repository contains the following files:

- `cnn.py` the main file;
- `augmentation.py` implements the data augmentation;
- `create_training_dic.py` together with
- `create_train_ids.py` is used to create a labelled list of the file to use for the training phase;
- `load_data.py` implements the loading of the input data to the CNN;
- `resnet.py` contains the network architecture;
- the folder `HumVI_online_lensing` contains a library that is used to produce RGB images and is a slightly modified version of the HumVI library (`https://github.com/drphilmarshall/HumVI`);
- The folder data which contains the average PSFs for the g, r and i KiDS bands and a table with a list of tuples of simulated g, r and i magnitudes for producing 3-band images.

C Data preparation

Create N subfolders in *data/training/sources/* with names ranging from one to the total number of simulated files. In each subfolder put a simulated source named accordingly, e.g. in folder 1 there will be `1.fits`.
Place in data/training/lenses/ all the training lens examples. The files have to be named `xxx_g_xxx.fits, xxx_r_xxx.fits, xxx_i_xxx.fits`, one for each of

the three different KiDS bands.
Place the negative examples, in the same manner as above, in *data/training/negatives/*.
Run the script `create_training_dic.py` and, subsequently,
`create_training_ids.py`.
Put the test data in *data/test_data/* in the same manner as above.

D Parameters

At the beginning of the file cnn.py there is a list of the main parameters of the script with their default values. A description of the parameters follows:

- `nbands` either 1 or 3 to choose between the 1-band or 3-bands ConvNet;
- `input_sizes` the size of the input images;
- `batch_size` number of inputs after which there is an update of the weights of the network;
- `chunk_size` the number of images loaded in one chunk (the final number will be twice this numbers, because one chunck is loaded for each the negative and positive examples);
- `num_chunks` the total number of chunks to load;
- `normalize` if True normalize the images between 0 and 1 (for single-band only);
- `model_name` the name of the model that will be saved at training time or the name of the model to load at test time;
- `learning_rate` the learning rate, i.e., the magnitude of weight updates at training time;
- `range_min` At training time gives the minimum value of the ratio between the maximum brightness of the simulated source and the lens galaxy;
- `range_max` At training time gives the maximum value of the ratio between the maximum brightness of the simulated source and the lens galaxy.
- `augm_pred` if True, at testing time the cnn gives a prediction obtained by averaging of the p's for the original image and the images obtained operating a rotation of 90, 180 and 270 degrees;

E Running the script

From the terminal launch the command `python cnn.py resnet train` to train a CNN. When the training is complete a file with the weights of networks named is `[model_name]_weights.h5` is created. The same file is created if the training

is interrupted by the user for some reason. To run the trained CNN run the command `python cnn.py resnet predict`. The results of the test will be stored in a file named `'pred_[model_name].pkl'` as a list of the file names with the relative predictions.