Cogniac Coding Challenge Results

by Sergii Bondariev
https://github.com/sergii-bondariev/cog_chal

Idea

Extract N-dimensional vectors (embeddings) for every image from a pre-trained CNN on ImageNet, for both modified and original sets.
Use L2 norm of a difference between two vectors as a metric to judge the proximity of two images. Chose the match based on a minimum L2 norm.
Reason: During training of a CNN, there are many transformations are done to the same image, so after training a network considers transformed images to be similar to the original image.

Sequence of steps in the implementation

Please see the major steps in *run_all.sh*
Here are some explanations
1. Download modified images from Amazon AWS
2. Download images from Flickr, with the given user id and tag (people)
3. Download pretrained ResNet-101 model that was trained on ImageNet dataset
4. Extract features from modified images and from original images
In this step we extract from the penultimate fully-connected (FC) layer, which gives us a 2048 embedding.
5. Calculate similarity
As mentioned earlier, the metric here is a L2 norm of a vector which is an element-wise difference between embeddings. For fast processing, instead of calculating in a double for loop, all L2 norms are calculated at once using matrix operations.
6. Map predictions to original urls
This step is needed, because in step 5 we get a csv file that has image names are they are located on disk.

Results
Please see *results.csv* that is created after *run_all.sh* is run. I also include it into github repository.
On a validation set, *score.py* calculates 82% accuracy.

Future improvements

Had I more time (more than 2 hours), I would do the following things:
- plot modified/original pairs of images side by side and look at the images that didn't match correctly. Then having this knowledge, try to see if there is a pattern that current algorithm can't detect. For example, in the modified images I saw some images rotated about 45 degrees. If I knew that they are not matched correctly most of the time, I would then introduce several rotations to the modified images (and maybe flips) and choose the match that has the minimum L2 norm of a difference.

- try different metrics, for example, L1 norm of a difference
- try other pretrained models, for example, VGG-Net, Inception
- build a new deep learning model that takes two images and says whether they are similar, need to collect training data however.

- look online and see what methods do people use for matching images

- If this code is going to run on a much bigger set of data, think of a distributed approach (for example, using map-reduce paradigm)
- add more comments to the code
- add more checks for corner cases (for example, always check existence of files before reading from them, that all links generated for Flickr are valid)

Prerequisites to run the code (besides python)

1. Torch
http://torch.ch/

2. Install Torch modules, nn, cunn and cudnn (see https://github.com/torch)
This is needed for feature extraction. However I included into github *modified_features-101.t7* and *flickr_features-101.t7*, which are the results of script extractor.lua. So you may skip this installation step, and comment out in *run_all.sh* the lines that call *extractor.lua*.
One additional note, I run extractor.lua on GPU. If you would like a CPU version, please change 'cuda' by 'float' in this line in extractor.lua:

local output = model:forward(batch.input:**cuda**())

If you consider to do so, modules 'cunn' and 'cudnn' are not required.