Large Scale Hard Sample Mining with Monte Carlo Tree Search – Supplementary material –

Olivier Canévet[1,2] and François Fleuret[1]
[1]Idiap Research Institut, Switzerland
[2]École Polytechnique Fédérale de Lausanne (EPFL), Switzerland
{olivier.canevet,francois.fleuret}@idiap.ch

Abstract

We present additional information that can help understanding our MCTS Bootstrapping method.

1. Clustering datasets

When a dataset comes with no explicit structure such as Microsoft Coco [1] or Pascal, we perform a top-down clustering that produces a more consistent hierarchy of the images.

To this purpose we make a 48×64 thumbnail of each image, compute its features (either the gradient or the GIST features) and then perform the clustering: All thumbnail features are clustered into k groups of thumbnails with a k-mean algorithm, then each of them are clustered again into k clusters until reaching a cluster of size 1. This recursive procedure produces a hierarchy of the images based on the similarity between image features (gradient or GIST).

Figure 1 shows a small scale example of a top-down $k-$mean based clustering of the Coco dataset. We see that a simple clustering such as this one is able to put similar images together: planes over blue sky, pictures of courses (pizzas, plates, food), landscapes with a strong horizontal line (sky on the top of the image, earth in the bottom), etc.

When the MCTS Bootstrapping procedure recursively explores such a structure dataset, it figures out quickly that sub-branches of the tree are more informative than others to find hard samples:

- When a branch contains many images of planes/bird over a blue sky, it does not generate many false detections, nor does a landscape.
- When a branch exhibits high frequency patterns such as stairs, vertical structures (trains, vehicles), or circles (pizza, dishes) that can easily be mistaken for a face.

2. Designing exploitation scores

In the usual bandit setup, the rewards are either 0 or 1. In the UCB1 strategy, the score of a bandit k is

$$\frac{x_k}{n_k} + \sqrt{\frac{2 \ln t}{n_k}},$$

where x_k is the number of rewards (0 or 1), n_k how many times it was played and $t = \sum_i n_i$ it the number of plays done so far.

The exploration part $\sqrt{2 \ln t / n_k}$ is a decreasing function of the time. After some time, $n_k \gg \ln t$ and therefore the UCB1 policy will only select the most promising arms.

When porting the bandit-based approach to bootstrapping, the scores should be carefully designed to prevent the exploration score to dominate the exploitation score for too long, otherwise the exploration is too long and by the time the exploitation starts, enough hard samples are found and the MCTS does not bring anything compared to a uniform approach.

We encountered this problem with the vanilla density of hard samples $d = h/S$, where h is the number of hard samples found in an image and S its size. As a detector can be evaluated $\sim$100,000 times on a 640×480 image, finding 10 hard samples would lead to a reward of 0.0001. Therefore the rewards do not fully range in $[0, 1]$ and the exploration phase would be too long.

This is why we used the normalised density $\tilde{d}$. This sore is designed to scale the density in the full range $[0, 1]$:

$$\tilde{d} = \min \left\{ 1, \frac{1}{2Z} \frac{h}{S} \right\},$$

where Z is the mean of the density of hard samples that is estimated as the images are visited. Very dense images then receive a reward of 1.

3. Evolution of performance for DPM

The DPM face detector we trained has 3 components as described in [2]:

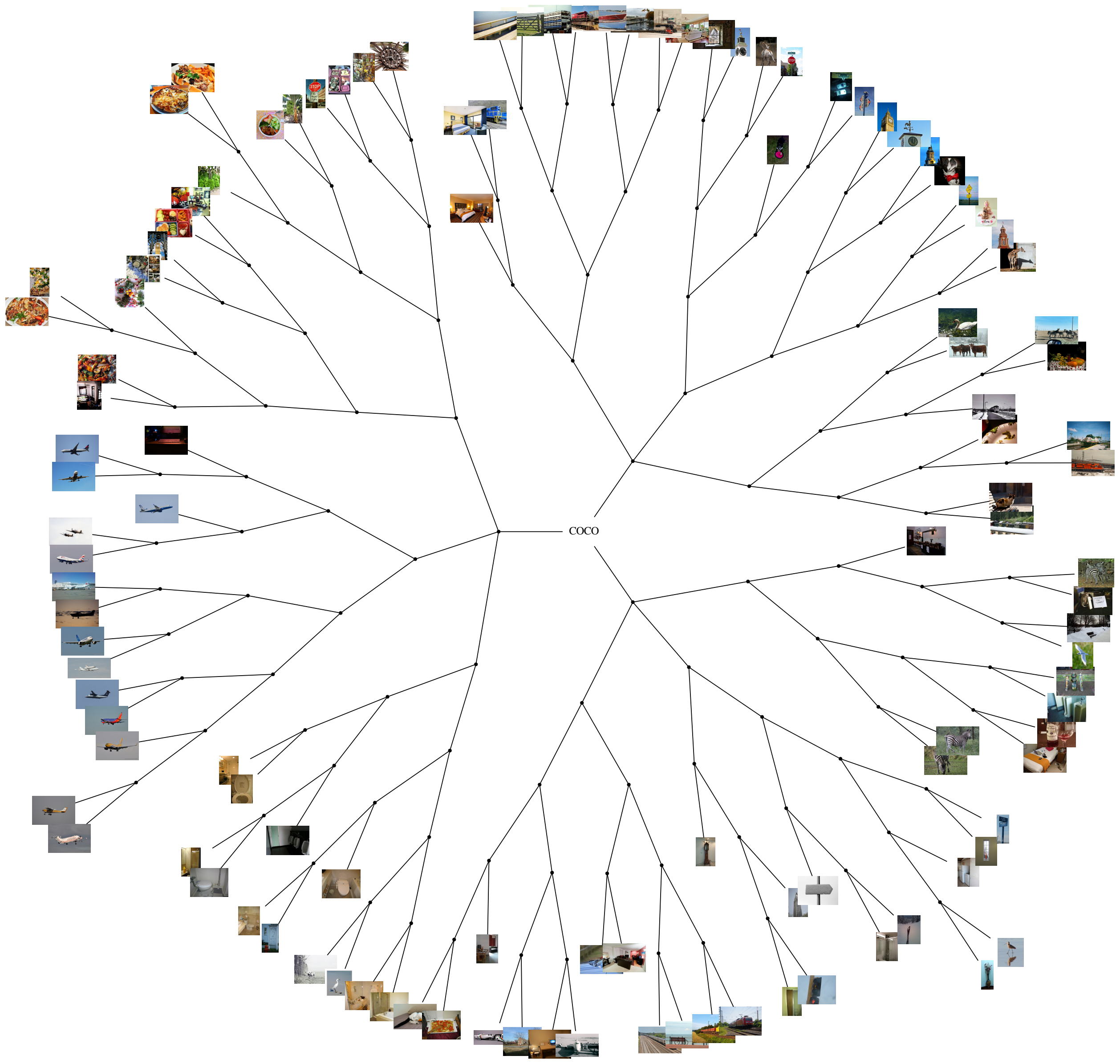


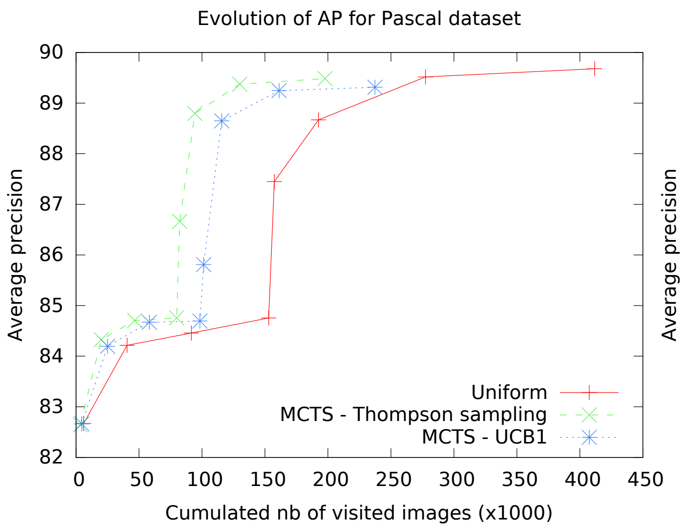
Figure 1: A subset of Microsoft Coco dataset organised as a tree with gradient based top-down clustering. Similar images are put in the same branches. A MCTS-based approach can then identify which branches are promising to find hard samples.

- One for the frontal faces
- One for the semi profile (yaw in $[20, 60]$ degrees)
- One for the profile (yaw in $[60, 100]$ degrees).

We performed 4 steps of relabelling (when the positive samples swap components based on strongest response of a given component) to train the root and the parts.

Figure 2 depicts the evolution of the performance of the detector as the relabelling steps are performed. As the MCTS based strategies visit less images, it reaches top performance faster compared to the uniform approach.

Figure 3 present the average-precision curves our DPM-face detector. The curves are averaged over 4 runs. We also put the curve of the original work by Mathias *et al.* [2] to show that the two different implementations yield comparable results.



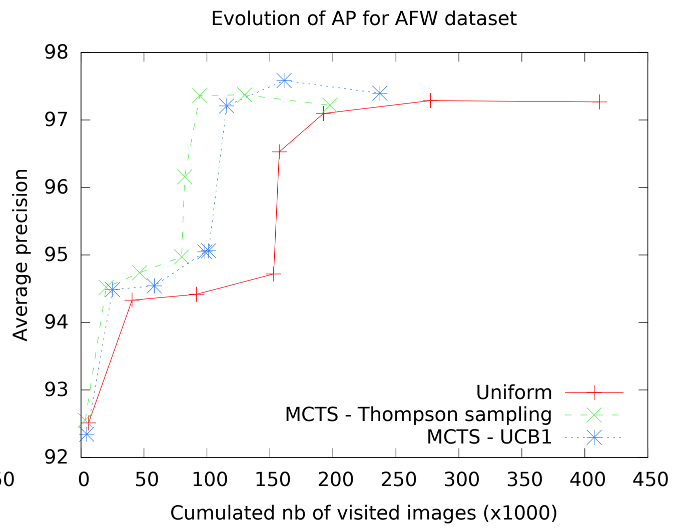
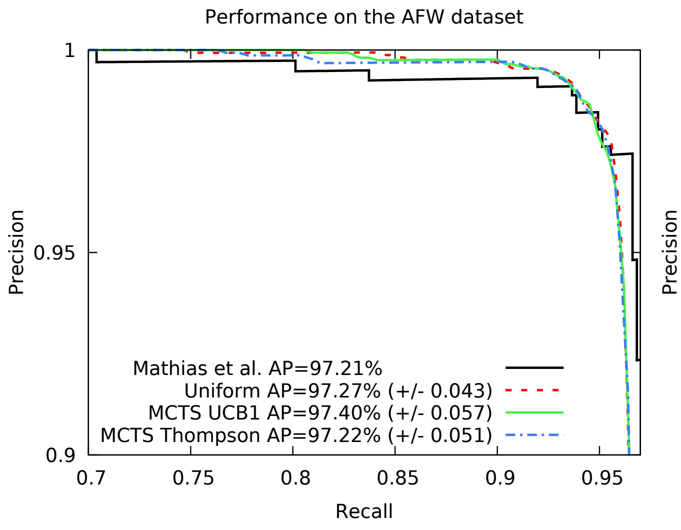


Figure 2: Evolution of the AP score as the relabelling steps are performed. The MCTS strategies reaches top performance earlier than the uniform baseline.



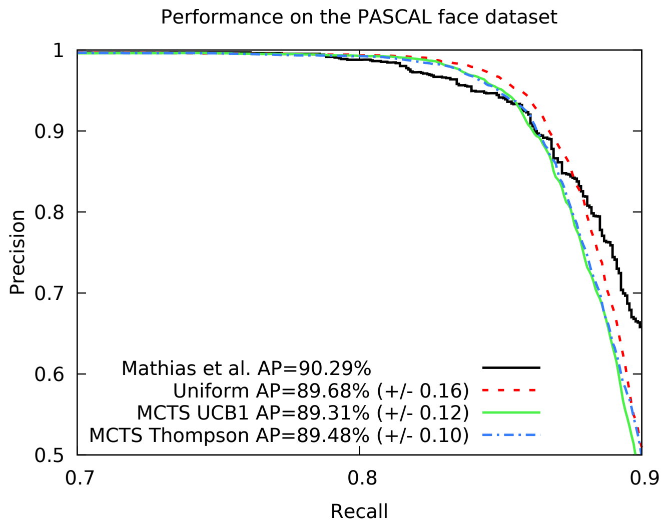


Figure 3: Our MCTS strategies (averaged over 4 runs) lead to the same accuracy as the uniform approaches (including [2]). The AP scores can be misleading since AP curves cross one another.

4. Selected images

Figure 4 shows images that were often visited when training a DPM face detector (top rows) and images that belonged to seldom visited branches (bottom rows).

Images that are frequently visited exhibit patterns that are reminiscent of a face (circles, animals, etc.) whereas images not frequently have large uniform patches (sea, sky, etc.)

References

[1] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *Computer Vision – ECCV 2014*, pages 740–755. Springer, 2014. 1

[2] M. Mathias, R. Benenson, M. Pedersoli, and L. Van Gool. Face detection without bells and whistles. In *Computer Vision–ECCV 2014*, pages 720–735. Springer, 2014. 1, 2, 3

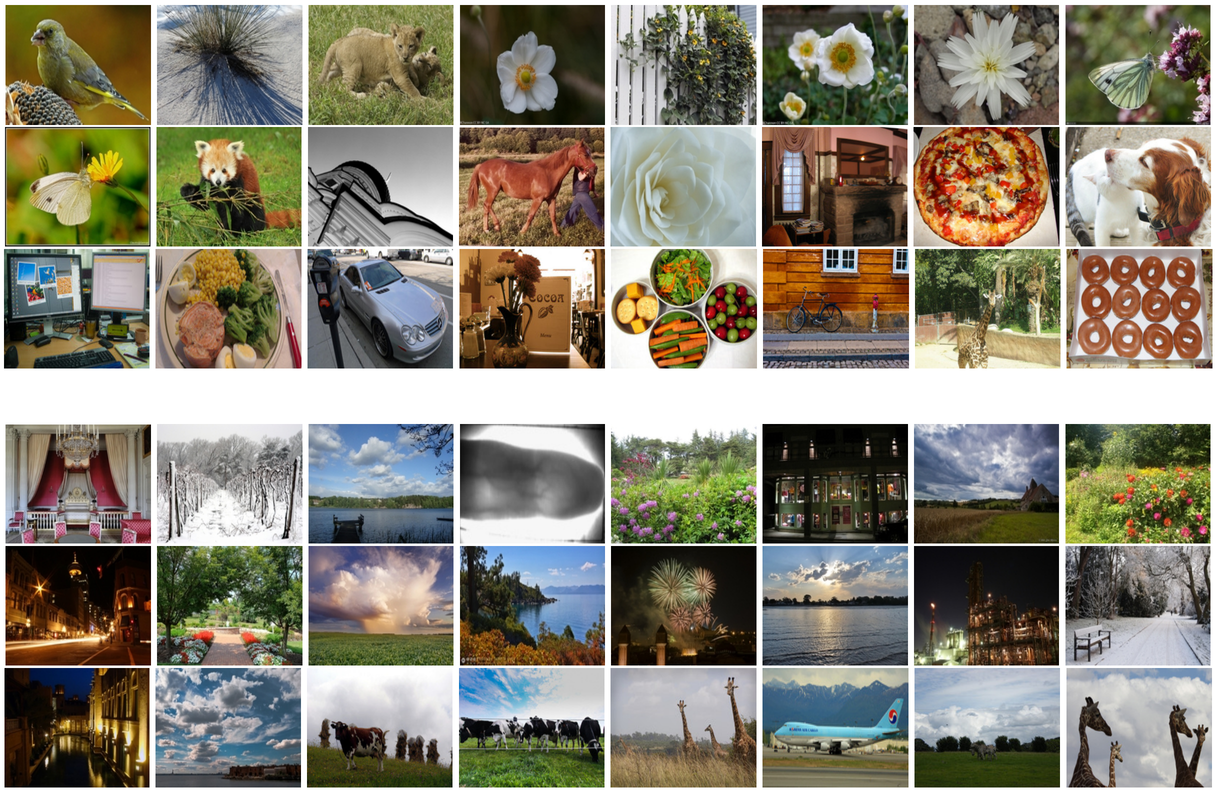


Figure 4: Mined negative images – Images visited when training a DPM face detector. Top: Images revisited 20 times. Bottom: Images revisited 1 time.