Ant Genera Identification Using an Ensemble of Convolutional Neural Networks

Alan Caio R. Marques,Marcos M. Raimundo,Ellen M. B. Cavalheiro,Luis F. P. Salles,Christiano Lyra,Fernando J. Von Zuben.

Abstract

Works requiring taxonomic knowledge face several challenges, such as arduous identification of many taxa and an insufficient number of taxonomists to identify a great deal of collected organisms. Machine learning tools, particularly convolutional neural networks (CNNs), are then welcome to automatically generate high-performance classifiers from available data.

We propose an ensemble of CNNs to identify ant genera directly from the head, profile and dorsal perspectives of ant images.

Transfer learning is also considered to improve the individual performance of the CNN classifiers.

Citation: Alan Caio R. Marques,Marcos M. Raimundo,Ellen M. B. Cavalheiro,Luis F. P. Salles,Christiano Lyra,Fernando J. Von Zuben. Ant Genera Identification Using an Ensemble of Convolutional Neural Networks. **protocols.io**
dx.doi.org/10.17504/protocols.io.kzpcx5n
Published: 04 Jan 2018

Protocol

Install Prerequisite Softwares and Libraries

Step 1.

This protocol requires:

- python (https://www.python.org/)
- Numpy (http://www.numpy.org/)
- Jupyter Notebook (http://jupyter.org/)
- Caffe - Deep learning framework (http://caffe.berkeleyvision.org/)

Each instruction can be verified on property websites.

The main steps are the following: dataset preparation, CNN training (using text terminal), CNN output gathering (using Jupyter notebook) and a phase to create an Ensemble (Ensemble building and output gathering).

EXPECTED RESULTS

Install prerequisites (Python, Caffe, Notebook, Numpy)

Run the script for dataset gathering

Step 2.

To download the dataset with images available in www.antweb.org, use the code in Zenodo (Can be reached by DOI: http://doi.org/10.5281/zenodo.1134690).

In Linux run the commands:

```
$ wget
https://zenodo.org/record/1134690/files/marcosmrai/antweb_crawler-v0.1.zip
$ unzip antweb_crawler-v0.1.zip
$ cd marcosmrai-antweb_crawler-526379e/
$ python crawler.py
```

Set project and image folders (you can leave it in blank to choose the default), and choose option 2. It will download all images from antweb.org.

EXPECTED RESULTS

Download all ant images from antweb.org.

Some images might be corrupted. If that happens, you might have to download another version direct from the site.

Manage data split

Step 3.

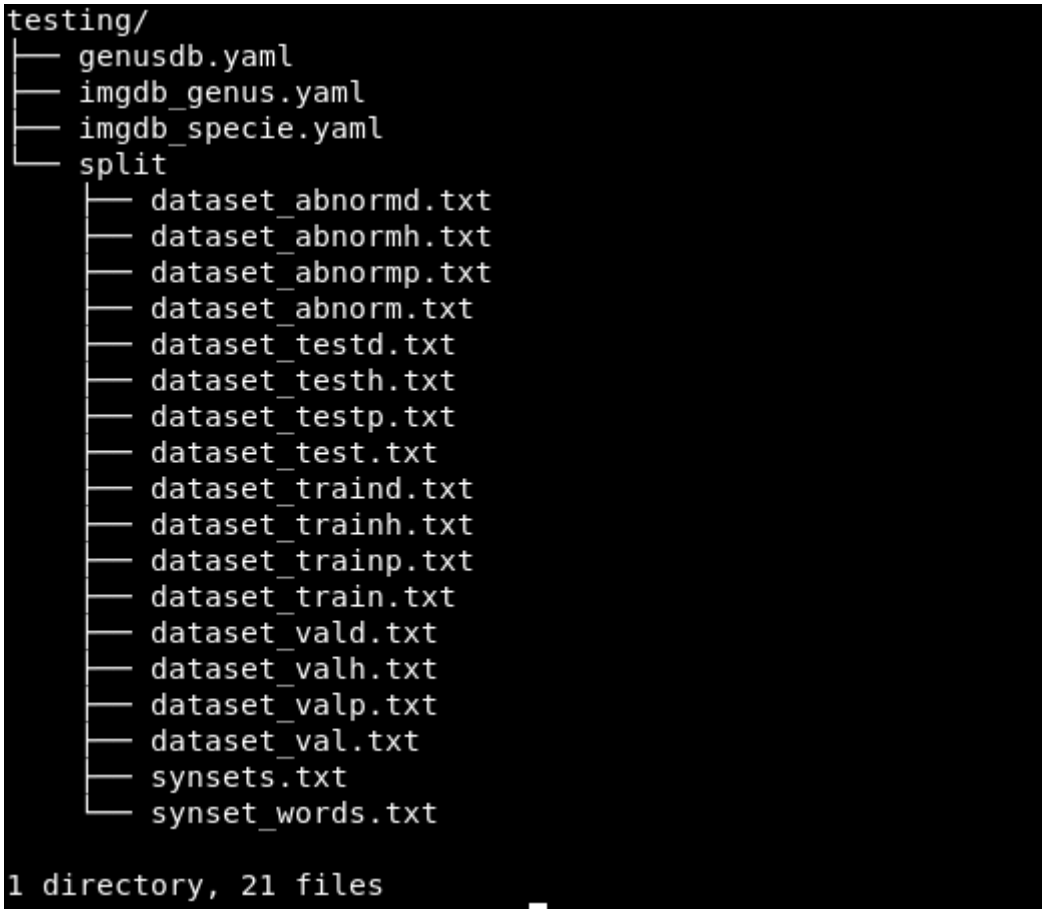
Run the command (in Linux) below (keeping the same folders selected previously) and select option 3 to split the dataset.

```
$ python crawler.py
```

Running the command below, in Linux, it's possíble see the generated folder and files.

```
$ tree testing
```

You will see the following result.



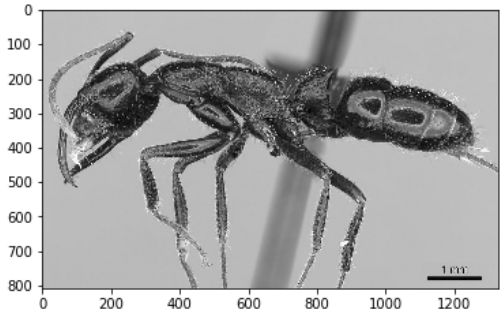
The files inside the paste 'split' are the dataset split required by Caffe.

Inside the dataset57 folder are the files used in our experiment, if you want to use the same data split and download the same images.

Image Example (Leptogenys/casent0095124/casent0095124_p_1_high.jpg)

```
plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7f7d051a8cf8>
```



EXPECTED RESULTS

Construct a dataset split needed by caffe.

Preparing Data to Caffe

Step 4.

Caffe uses database .mdb to make the training.

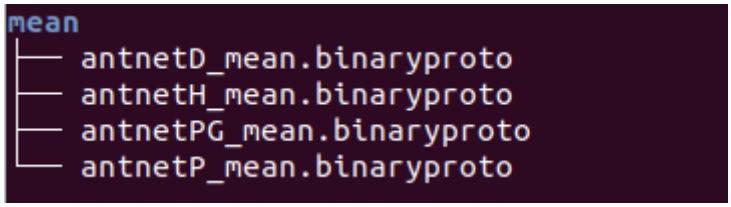
The file **examples/imagenet/create_imagenet.sh** was used to create this dataset. The file was modified to generate 2 datasets (training and validation) in order to test the different hypothesis.

The generated files (data.mdb and lock.mdb) are inside the folders ants#PB_train_lmdb and ants#PB_val_lmbd, where # represents the code for each one of perspectives (Head - H, Profile - P, Dorsal - D, all perspectives - null)

All files generated can be seen with the command 'tree testing'; with the following results:

The file with the average of the images was created to assist the training using the following command:

```
$ /caffe/install/dir/compute_image_mean ./data/antnet?_mean.binaryproto
```



Transform mean.binaryproto in _mean.npy

Step 5.

In order to use the mean.binaryproto in the testing phase it is necessary to transform .binaryproto in to .npy. Using the following code in Python.

```
import numpy
import caffe
import numpy as np
import sys
```

Prepare for changes

```
blob = caffe.proto.caffe_pb2.BlobProto()
data = open( '/directory/mean/antnet?_mean.binaryproto' , 'rb').read()
blob.ParseFromString(data)
arr = np.array( caffe.io.blobproto_to_array(blob))
out = arr[0]
```

Save the new archive

```
np.save('/image_mean/directory/antnet?_mean.npy'' , out )
```

Training models with Caffe

Step 6.

The models were trained in Linux with Caffe.

To train one model with all perspective was used:

```
$ /caffe/install/dir/caffe train --solver=models/bvlc_general/solver.prototxt
```

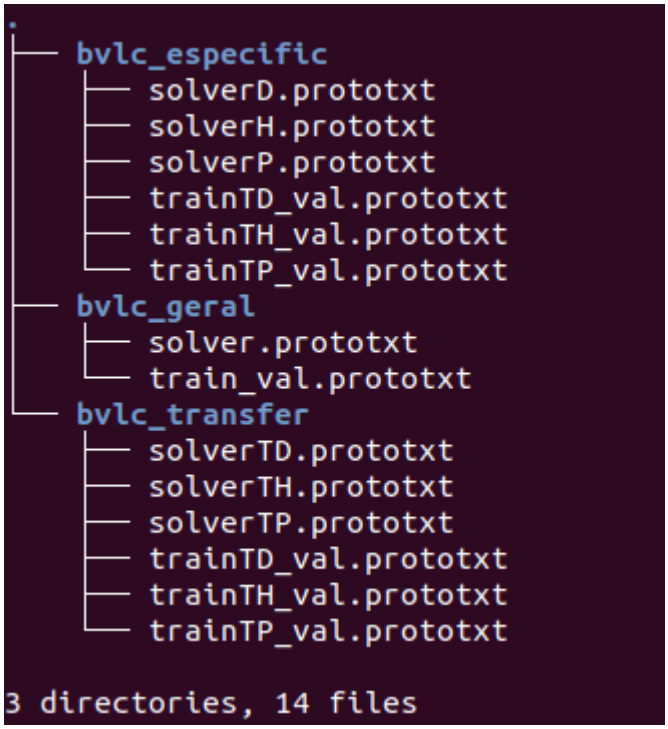
To train one model to each perspective was used:

```
$ /caffe/install/dir/caffe train --
solver=models/bvlc_especific/solverD.prototxt
$ /caffe/install/dir/caffe train --
solver=models/bvlc_especific/solverH.prototxt
$ /caffe/install/dir/caffe train --
solver=models/bvlc_especific/solverP.prototxt
```

To train one model to each perspective with a feature extracted to the machine with all perspectives was used:

```
$ /caffe/install/dir/caffe train --
solver=models/bvlc_transfer/solverTD.prototxt--
snapshot=models/bvlc_antnet/snapshot/antsPB_train_iter_50000.solverstate
$ /caffe/install/dir/caffe train --
solver=models/bvlc_transfer/solverTH.prototxt--
snapshot=models/bvlc_antnet/snapshot/antsPB_train_iter_50000.solverstate
$ /caffe/install/dir/caffe train --
solver=models/bvlc_transfer/solverTP.prototxt--
snapshot=models/bvlc_antnet/snapshot/antsPB_train_iter_50000.solverstate
```

The command 'tree testing' results in:



EXPECTED RESULTS

Train the different models using Caffe.

Gathering the outputs from Caffe CNN

Step 7.

The testing analysis was created using Jupyter Notebook.

#Necessary imports

```
import numpy as np
import caffe
```

#Prepare paths and files

```
model_file = '/module/directory/bvlc_my_model.caffemodel'
deploy_prototxt = '/deploy/directory/deploy.prototxt'
imagemean_file =
np.load('/image_mean/directory/antnet#_mean.npy').mean(1).mean(1)
```

#Use caffe.set_mode_cpu() to compute in CPU

```
caffe.set_mode_gpu()
```

#Prepare the module

```
net = caffe.Net(deploy_prototxt, model_file, caffe.TEST)

transformer = caffe.io.Transformer({'data': net.blobs['data'].data.shape})
transformer.set_mean('data', imagemean_file)
transformer.set_transpose('data', (2,0,1))
transformer.set_raw_scale('data', 255.0)

net.blobs['data'].reshape(50,3,256,256) #batch size 50, 3 chanells but all
grayscale, height 256, width 256
```

#Loading one image for test (the file test dataset_test.txt can be read in a loop)

```
img =
caffe.io.load_image('/image_example/Leptogenys/casent0095124/casent0095124_p_
1_high.jpg')
forward(img, net, transformer)
```

```
def forward(image, net, transformer):
     net.blobs['data'].data[...] = transformer.preprocess('data', img)
     output = net.forward()
     output_prob = output['prob'][0]
     return output_prob
```

(Result to Example)

#9.96067882e-01 Leptogenys

EXPECTED RESULTS

Gather the output from dataset using Jupyter Notebook. Save results for use in Ensemble.

Using Results in Ensemble

Step 8.

Executing the commands on Step 7 but changing the command below you can load a different model:

```
model_file = '/module/directory/bvlc_my_model.caffemodel'
```

As a result, we can load the models:

```
model_gen #general model
model_spe_head #head specific model
model_spe_profile #profile specific model
model_spe_dorsum #dorsum specific model
model_tra_head #head transfered model
```

```
model_tra_profile #profile transfered model
model_tra_dorsum #dorsum transfered model
```

Next, build the following ensembles - the `forward(image, net, transformer)` python method can be seen in step 7.

General Ensemble:

```
img_gen_head =
caffe.io.load_image('/directory/image_example/Leptogenys/casent0095124/casent
0095124_h_1_high.jpg')
prob_gen_head = forward(img_gen_head, model_gen, transformer)
img_gen_profile =
caffe.io.load_image('/directory/image_example/Leptogenys/casent0095124/casent
0095124_p_1_high.jpg')
prob_gen_profile = forward(img_gen_head, model_gen, transformer)
img_gen_dorsum =
caffe.io.load_image('/directory/image_example/Leptogenys/casent0095124/casent
0095124_d_1_high.jpg')
prob_gen_dorsum = forward(img_gen_dorsum, model_gen, transformer)

prob_gen_ensemble = prob_gen_head+prob_gen_profile+prob_gen_dorsum
```

Specific Ensemble:

```
img_spe_head =
caffe.io.load_image('/directory/image_example/Leptogenys/casent0095124/casent
0095124_h_1_high.jpg')
prob_spe_head = forward(img_gen_head, model_spe_head, transformer)
img_spe_profile =
caffe.io.load_image('/directory/image_example/Leptogenys/casent0095124/casent
0095124_p_1_high.jpg')
prob_spe_profile = forward(img_gen_head, model_spe_profile, transformer)
img_spe_dorsum =
caffe.io.load_image('/directory/image_example/Leptogenys/casent0095124/casent
0095124_d_1_high.jpg')
prob_spe_dorsum = forward(img_gen_dorsum, model_spe_dorsum, transformer)

prob_spe_ensemble = prob_spe_head+prob_spe_profile+prob_spe_dorsum
```

Transfer ensemble:

```
img_tra_head =
caffe.io.load_image('/directory/image_example/Leptogenys/casent0095124/casent
0095124_h_1_high.jpg')
prob_tra_head = forward(img_tra_head, model_tra_head, transformer)
img_tra_profile =
caffe.io.load_image('/directory/image_example/Leptogenys/casent0095124/casent
0095124_p_1_high.jpg')
prob_tra_profile = forward(img_tra_head, model_tra_profile, transformer)
img_tra_dorsum =
caffe.io.load_image('/directory/image_example/Leptogenys/casent0095124/casent
0095124_d_1_high.jpg')
prob_tra_dorsum = forward(img_tra_dorsum, model_tra_dorsum, transformer)

prob_tra_ensemble = prob_tra_head+prob_tra_profile+prob_tra_dorsum
```

The Ensemble of all models (considering the models already loaded):

```
prob_all_ensemble = prob_gen_head+prob_gen_profile+prob_gen_dorsum+\
prob_spe_head+prob_spe_profile+prob_spe_dorsum+\
prob_tra_head+prob_tra_profile+prob_tra_dorsum
```