Open-set Plant Identification Using an Ensemble of Deep Convolutional Neural Networks

Mostafa Mehdipour Ghazi[1], Berrin Yanikoglu[1], Erchan Aptoula[2]

Sabanci University, Istanbul, Turkey
Gebze Technical University, Kocaeli, Turkey
{mehdipour, berrin}@sabanciuniv.edu, eaptoula@gtu.edu.tr

Évora, Portugal, 6th of September, 2016

This work was supported by the Scientific and Technological
Research Council of Turkey (TÜBİTAK) under the grant number 113E499

Outline

1. Our group
2. Our strategy
3. Training & data augmentation
4. Dealing with non-plant classes
5. Results and conlusion

Our Group

- We have been participating at all the plant identification contests of *CLEF since 2011
  - Prof. Berrin Yanikoglu, Sabanci University
  - Assoc. Prof. Erchan Aptoula, Gebze Technical University

- Along with several hard-working students:
  - Mostafa Mehdipour Ghazi, 2016, 2015 (deep learning)
  - Ö. Muslu, M. C. Özdemir, 2015 (local descriptors, ...)
  - S. Tolga Yıldıran, 2014, 2013 (dense SIFT,...)
  - Çaglar Tırkaz, 2014, 2012, 2011 (SVM and classifier ensemble with complex/hand selected sets of features)

Deep Learning

- Deep learning based approaches have not only dominated LifeCLEF 2015, but also resulted in a performance leap w.r.t. previous years; thanks to:
  - Parallel architectures, accelerating training and learning.
  - Data augmentation, dealing with overfitting
  - ReLU's better/faster convergence
  - The dropout of neurons during training which brings the benefits of ensemble averaging, etc.

Our Strategy for 2016

Either train deep neural networks from scratch

-> requires A LOT of data and resources (we lack both ☹)

-> abandoned it, after a preliminary exploration

Or fine-tune, i.e. adapt a pre-trained deep network from a similar task domain (e.g. object recognition) to the plant identification problem

-> provides reasonable performance even with modest data amounts and training times ☺

Choosing a net to fine-tune..

- GoogLeNet
  - Winner of ILSVRC 2014, 6.8 million parameters

  Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: IEEE Conference on Computer Vision and Pattern Recognition. (2015)

- VGGNet
  - Runner-up in ILSVRC 2014, 144 million parameters

  Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. Computing Research Repository (CoRR) (2014) arXiv: 1409.1556.

How to fine-tune?

-> **How many iterations**: as many as you can afford.

-> **Data augmentation**: how many more samples are to be generated per input image?

-> **Batch size**: the number of input samples used during gradient calculation for weight update

+ **learning rate**, **weight decay**, **momentum** (typically initialized to default values)

The first 3 affect training time strongly!

How to fine-tune?

- Preliminary work focused on identifying the relative importance of iteration number, batch size, and data augmentation

- All tests were run on a linux system with a Tesla K40c and 12GB of video memory

- Average training time per iteration with a batch size of 20: 1.79 seconds for VGGNet and 0.45 seconds for GoogLeNet

Data Augmentation

- Very helpful in gaining robustness against overfitting

-> 5 random square patches around the center
-> 2 square patches obtained by rotation
-> All (5+2+1=8) scaled to 256x256 pixels

x8

From these 8 images:
-> 5 patches were extracted from their corners and center: 224x224 and each was horizontally reflected

x10

- Thus, an **80-fold** data augmentation has been achieved.
- Score level averaging of the 8 images has been employed.

Fine-Tuning settings

- Both networks were fine-tuned as follows:
  - Input size: 224x224
  - Output size: 1000
  - Batch size: 20
  - Weight decay factor: 0.0002
  - Learning rate: 0.001
  - Updating iteration size for the learning rate: 12,000
  - Updating gamma for the learning rate: 0.96
  - Momentum: 0.9
  - Number of iterations: 100,000
  - Data augmentation: 10-fold

All determined empirically!

Initial Results

- The 3 critical parameters were empirically optimized across many runs using the LifeCLEF 2015 dataset.

  Notation: (100K, 20, 10x): 100K as number of iterations, 20 as batch size and 10-fold data augmentation.

- LifeCLEF 2015 was used both for training and testing:
  - Training set: 91,758 labeled images of different plant organs from 1,000 species.
  - Testing set: 21,446 images.

	Branch	Entire	Flower	Fruit	Leaf	LeafScan	Stem	Overall
GoogLeNet (100K, 20, 10x)	44.09	38.36	67.93	57.65	60.57	94.16	37.01	61.06
GoogLeNet (300K, 20, 10x)	55.08	46.05	76.23	67.96	68.07	95.77	45.76	68.57
GoogLeNet (500K, 20, 10x)	55.02	47.66	76.43	69.11	69.13	95.58	45.49	69.11
GoogLeNet (100K, 40, 10x)	45.63	41.03	70.05	61.23	60.73	93.20	36.49	62.48
GoogLeNet (100K, 60, 10x)	50.98	41.09	73.07	63.59	65.50	94.19	41.52	65.18
GoogLeNet (100K, 20, 80x)	54.01	48.06	73.38	64.99	68.63	94.85	39.84	67.32
GoogLeNet (300K, 60, 80x) (∗)	67.56	60.28	83.30	76.88	76.30	96.93	54.33	76.87

- The best GoogLeNet results are obtained with (300K,60,80x) as 76.87% on the test set.
- Increasing the number of iterations is much more beneficial than increasing the batch size, for the same total running time (68.57% vs. 65.18 %).
- Data augmentation is the second most important training parameter (69.11 % vs. 67.32 %)

Exploring GoogLeNet and VGGNet

	Branch	Entire	Flower	Fruit	Leaf	LeafScan	Stem	Overall
GoogLeNet (100K, 20, 10x)	44.09	38.36	67.93	57.65	60.57	94.16	37.01	61.06
GoogLeNet (300K, 20, 10x)	55.08	46.05	76.23	67.96	68.07	95.77	45.76	68.57
GoogLeNet (500K, 20, 10x)	55.02	47.66	76.43	69.11	69.13	95.58	45.49	69.11
GoogLeNet (100K, 40, 10x)	45.63	41.03	70.05	61.23	60.73	93.20	36.49	62.48
GoogLeNet (100K, 60, 10x)	50.98	41.09	73.07	63.59	65.50	94.19	41.52	65.18
GoogLeNet (100K, 20, 80x)	54.01	48.06	73.38	64.99	68.63	94.85	39.84	67.32
GoogLeNet (300K, 60, 80x) (∗)	67.56	60.28	83.30	76.88	76.30	96.93	54.33	76.87
VGGNet (100K, 20, 10x)	44.68	42.58	66.37	59.59	61.90	95.05	38.87	61.93
VGGNet (500K, 20, 10x)	56.33	52.81	77.53	68.93	70.47	97.29	51.68	71.24
VGGNet (500K, 20, 80x) (∗∗)	68.09	64.37	84.40	77.00	76.67	97.92	58.75	78.44
GoogLeNet (∗) & VGGNet (∗∗)	**71.24**	**65.16**	**86.90**	**79.13**	**78.93**	**98.02**	**59.45**	**80.18**

The final fusion classifier achieves 0.752 average inverse rank score on the 2015 test dataset (corresponding to 80.18% classification accuracy).

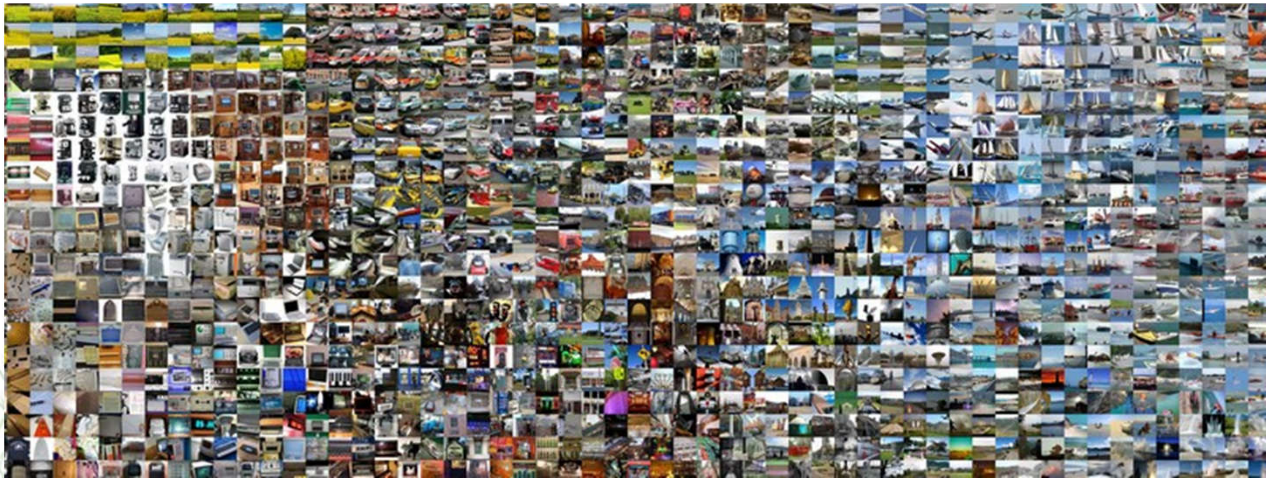
Getting rid of non-plant images

- Test images that do not belong to any known (in the training set) class should be rejected.
  - e.g. photos of people, furniture, but also tropical plants!

- Can it be accomplished through confidence scores ?
  - Initial analysis showed confidence scores to be unreliable, especially for plants of unknown classes.
    - Non-plant objects had often low scores, but there were also plant images with low scores.
    - On the other hand, some objects such as tables had higher scores due to their similarity to some *stem* images.

Plant/Non-Plant Binary Classifier

- We trained a binary classifier for identifying non-plants using the **ImageNet** ILSVRC’12 dataset as negative examples (**without** the potted plant categ.).
- Then, we reject those images that are classified as ***both*** non-plant ***and*** that have received a low confidence score in the main plant identification system.



Campaign Results

- Run 1: the fine-tuned plant identification system, thresholding low confidence scores of non-plant images, as provided by the binary classifier (mAP **0.738 first runner up**)
- Run 2: the fine-tuned plant identification system (mAP **0.736 third runner up**)
- Run 3: the fine-tuned plant identification system with manual rejection of 90 non-plant images (mAP **0.737 second runner up**)

- **However, all these results are purely image based.** Unfortunately, observation IDs have been skipped this year.

- Bluefield (Run4) is the winner with mAP **0.742**.

Thank you for your attention!

The SabanciUGebzeTU Team

Berrin Yanikoglu
berrin@sabanciuniv.edu

Mostafa Mehdipour Ghazi
mehdipour@sabanciuniv.edu

Erchan Aptoula
eaptoula@gtu.edu.tr