CS230

LeafNet: A Deep Learning Solution to Tree Species Identification

Elena Galbally
Department of Mechanical Engineering
elenagal@stanford.edu

Krishna Rao
Department of Earth System Science
kkrao@stanford.edu

Zoe Pacalin
Department of Computer Science
zpacalin@stanford.edu

Abstract

Plant identification is a key step in plant biodiversity research and conservation biology. Speeding up this process can boost our ability to protect the environment by simplifying species conservation efforts and helping educate the public. In this study we used a Residual Network (ResNet) to automatically classify 185 tree species from North America based on leaf images. Our results show reduced computation and higher recognition precision than any existing system.

1 Introduction

Understanding and preserving worldwide biodiversity is central to addressing challenges associated with resilience to climate change and reducing the impact of greenhouse gases. Amidst growing threats to biodiversity - such as deforestation, overexploitation, or pollution - species conservation becomes increasingly important. Plant species identification - a fundamental first step to quantifying biodiversity - can be challenging for both researchers and the general public. Therefore, the ability to reliably and easily identify plant species holds great potential to increase knowledge accessibility as well as facilitate greater collective ability to protect the environment. With this goal in mind we designed an algorithm that takes as an input an RGB leaf image. We then used a ResNet18 to classify the image into 1 of 185 classes corresponding to its species 1. The model was tuned for various hyperparameters to achieve an overall top-1 precision of 93.8% on the dev set. The final model is hosted on a remote server and allows anyone to use the classifier for free by logging into Google Hangouts and opening a chat with leafnetstanford@gmail.com.

2 Related work

There are four main approaches to plant classification: manual identification, image processing, machine learning, and DNA barcoding. The first is tedious and slow, since it involves searching for each species in a printed or digital key [11]. Image processing techniques allow for higher throughput, but rely on hand-crafted algorithms that extract certain features chosen a priori to identify species [8, 13]. A traditional image processing approach involves: image binarization to separate background and the leaf, detection of contours and contour corners, and geometrical derivations of leaf tooth features. Given the extreme diversity of botanical data, it is easy to see that hand-crafted methods don't provide a scalable solution. In recent years, barcoding has started to gain some traction,

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

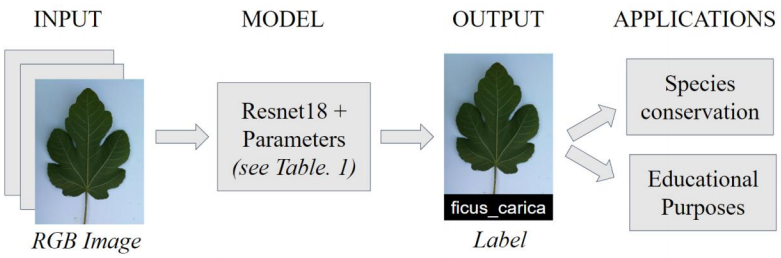

Figure 1: Visual summary of the identification system

however, it is an invasive method and it is not applicable to herbarium specimens where DNA quality has degraded [4]. Finally, to overcome these limitations, model-free approaches and machine learning methods have been introduced [1, 12].

Some state-of-the-art deep learning examples are: [9] where they report 99.7% precision by using a ResNet to identify 44 different species; [6] reached a classification mean average precision of 74.2% and won a classification competition based on the PlantCLEF dataset which consists out of 113,205 pictures of 1000 woody and herbaceous species from France and neighbouring countries. Finally, [3] used a custom network architecture on the *LeafSnap* dataset described in the next section and obtained 86.3 % top-1 precision after 200,000 iterations. We obtained higher performance in less iterations and with a simpler network.

3 Dataset and Features

For training and testing our model, we used the *LeafSnap* dataset [8] which contains 224x224 pixel images and covers 185 tree species from North America (40 to 200 examples of each species). The dataset is divided into two subsets:

- 23,147 *lab images* (Fig. 2, left) which are high-quality images of pressed leaves from the Smithsonian collection with controlled illumination.

- 7719 *field images* (Fig. 2, right) taken by different users in outdoor environments with their phones. Field images contain varying amounts of blur, noise, illumination patterns, shadows, etc. - but the image taking instructions of the *LeafSnap* project assure that each field image shows a uniform background (provided by holding a sheet of paper behind each leaf while taking the picture).

In order to reduce overfitting we enlarged the dataset by using data augmentation. In particular, we applied random rotations for a final training size of 120,000 images. As seen in 2, this led to significant improvements in precision. Additionally, the RGB values were normalized using mean=[0.485, 0.456, 0.406] and std=[0.229, 0.224, 0.225] to standardize the histograms of all three colors before passing the images into the model. This technique is useful when images are taken using different cameras or in different lighting conditions. Note that, for the dev set we only used field images, since we wanted to study the performance of our model under conditions close to those of a typical user.

Finally, we worked on incorporating geolocation as an additional input feature to help train the network and make it more robust to visual similarities. In order to do so we divided North America into 50,176 equal regions using a 224x224 grid. Using a plant database [2], we turned the grid into a binary matrix containing $1s$ in the regions where each species is most commonly found. The input to our system then became 224x224x4. The results corresponding to the four channel input remain inconclusive and will be part of our future work. Another line of work for the future will be augmenting the dataset by superimposing the current leafs on non-white backgrounds.

Figure 2: Example of a lab image (left) and field image (right) for the *Aesculus Pavi*

4 Methods

We tested several models and discovered that according to our evaluation metrics (see next section) the highest performing network architecture was ResNet18 with stochastic gradient descent (SGD) and cross-entropy loss.

$$J(\theta) = -\sum_{i} y_i log(\hat{y}_i) \tag{1}$$

In particular, we used an implementation of SGD that uses momentum (aka Nesterov SGD) In SGD with momentum, the update can be written as:

$$v = \rho * v + gp = p - lr * v \tag{2}$$

where p, g, v and ρ denote the parameters, gradient, velocity, and momentum respectively. Theoretically, it can be shown that Nesterov gradient descent is faster than stochastic gradient descent because it anticipates it clips the descent in anticipation of the next descending direction.

We chose a residual network because we wanted to be able to have a deep NN while minimizing potential issues with vanishing gradients. The main benefit of a deep network is that it can represent very complex functions and can learn features at many different levels of abstraction. This allows the network to distinguish between extremely similar species. The main drawback is that during backprop, you multiply by the weight matrix on each step, and thus the gradient can decrease exponentially quickly to zero for very large networks.

Having ResNet blocks with "skip connections" makes it very easy for one of the blocks to learn an identity function. This means that you can stack on additional ResNet blocks with little risk of harming training set performance. There is also some evidence that the ease of learning an identity function–even more than skip connections helping with vanishing gradients–accounts for ResNets' remarkable performance. ResNet18 combines identity and convolutional blocks.

All models were ran on an a 4-core GPU and coded using the following libraries: PyTorch, pandas, Pillow, pytest, h5py, sklearn, scipy, scikit-image, scikit-learn, keras [7, 10, 5]

5 Results, Metrics, and Discussions

We experimented with different architectures and tuned our hyperparameters to optimize our performance metric. Our goal was to use a sufficiently deep network to be able to learn complex features, but no deeper than necessary to minimize computation time. In other words, our optimizing metric was top-1 precision and our satisfycing metric having a model smaller than 100 Mb.

$$top - n \quad precision = \frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{n} I(\hat{y}_i = y_i) \tag{3}$$

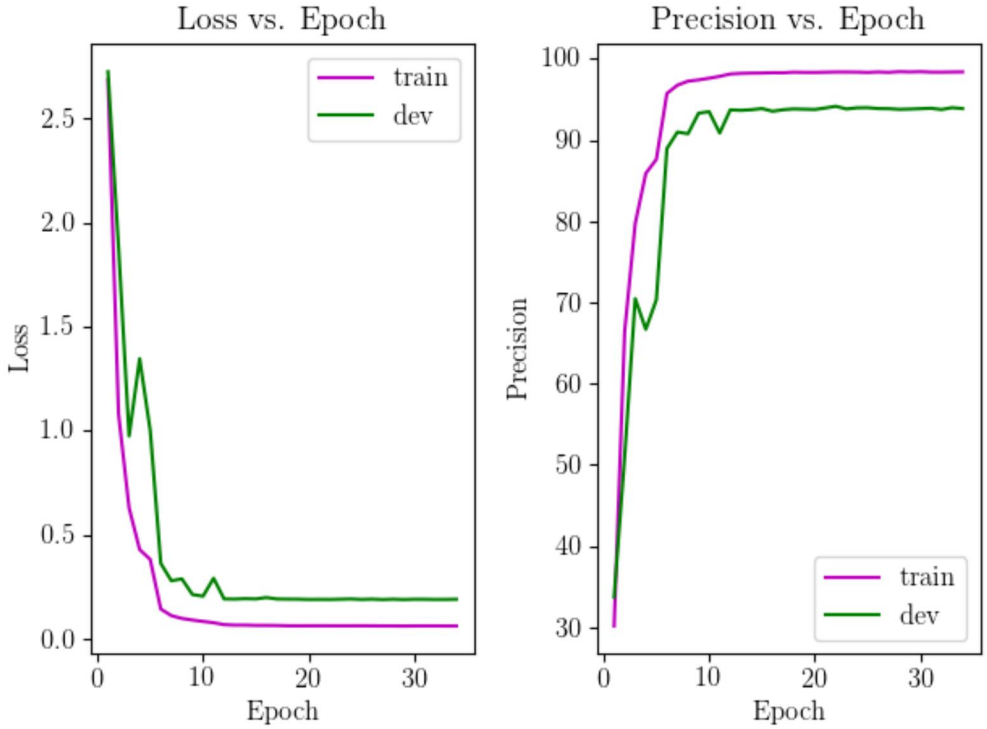

Figure 3: Precision and loss function curves for the highest performing model (dev prec-1 93.8%)

Hyperparameter	Value
Input	RGB image (224x224x3)
# of classes	185
# of epochs	stuffed
# hidden layers	stuffed
Mini batch size	128
Optimizer	Stochastic (lr = α, β_1=0.9, weight_decay=1e-4, Nesterov momentum)
Learning rate, α_0	0.1
Learning rate decay rule	$\alpha = \alpha_0 \times 0.1^{\frac{epoch}{k}}$ where k = 6
FC layer input size	512

Table 1: Hyperparameter details for our highest performing model

where (1) m is the number of example-label pairs (x_i, y_i), (2) $j = 1, ..., n$ the highest ranked species predictions according to their probability value, and (3) I is an indicator function that returns 1 when there is a match and 0 otherwise.

In summary our metric was: maximize precision while (if possible) reducing computation

Overall, we found that learning rate decay had the biggest effect on speeding up the convergence of the algorithm. Fast convergence was the key that enabled us to iterate quickly through different models and hyperparameter values. Precision benefited the most from data augmentation (rotation by 90 degrees) since it expanded the training size and also allowed the model to learn on rotation invariant properties of the leaf such as color, shape, and texture rather the orientation.

For our highest performing model (see table.1), we achieved an average top-1 precision of 93.8% and an average top-5 precision of 99.5%. We studied outliers to look for trends and improve precision. Fig. 4 shows the ten species that are responsible for the largest percentage of mispredictions.

Table 2: Parameters and performance of models used during the architecture search

Model	Batch Size	Optimizer	Input Size	l.r. decay?	epochs	top-1 prec(%)
Logistic Reg.	256	Vanilla GD	224	Yes	35	10.4
ResNet18	128	SGD	16	Yes	35	60.5
ResNet18	128	SGD	224	No	35	11.5
ResNet18	128	SGD	224	Yes	35	85.7
ResNet18	128	Adam	224	Yes	35	41.1
ResNet18	**128**	**SGD**	**224**	**Yes**	**35**	**93.8**
ResNet50	64	SGD	224	Yes	78	85.4

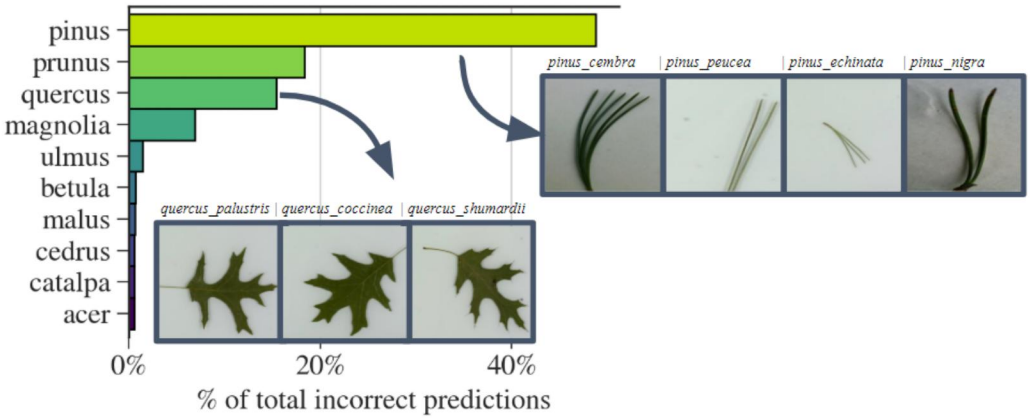

Figure 4: Family of species responsible for the errors

By looking at their corresponding images we were able to see that outliers can mostly be explained by: (1) species that show strong visual similarities or (2) species that are represented in the *LeafSnap* dataset by only a few images (making training of such species difficult). Using data augmentation mitigated the latter problem and we hope that improving geolocation will give the algorithm a new metric to distinguish leaves which despite looking almost identical may be more prevalent in entirely different locations.

The final model after all hyperparameter training was hosted on an AWS server and incorporated into a bot on Google Hangouts to allow any curious user to benefit from the work of Leafnet. At the moment, the leaf image must be taken on a white background because our model was only trained on such images. The product can be used by opening a chat window with leafnetstanford@gmail.com and saying "Hi bot" and sending a picture of any tree leaf on a white background.

6 Conclusion/Future Work

In the future it would be very interesting to use a deconvolution network to visualize the learned features. Botanists could then inspect the results and verify that the network is indeed making predictions using sensible information. Additionally, as mentioned in section 3, we would like to train our model using leaf images with non-white backgrounds and improve the geolocation feature in order to find out what its real impact on precision is.

Overall, the results of our ResNet model show that deep learning approaches offer a high-precision, high-throughput, non-invasive, solution for specimen classification. More specifically, ResNet18 with SGD and learning rate decay outperforms all other methods in terms of top-1 precision while using a relatively small number of layers and requiring less epochs to converge than other models.

7 Contributions

E.G. worked on code implementation, poster, and write-up; K.R. worked on code, error analysis, and google bot for the poster and write-up; Z.P. worked on the code implementation of the geolocation feature and helped with the poster.

References

[1] Flavia, A Leaf Recognition Algorithm for Plant Classification using PNN.

[2] PLANTS Database | USDA PLANTS.

[3] Pierre Barré, Ben C. Stöver, Kai F. Müller, and Volker Steinhage. LeafNet: A computer vision system for automatic plant species identification. *Ecological Informatics*, 40:50–56, 7 2017.

[4] CBOL Plant Working CBOL Plant Working Group, Peter M. Hollingsworth, Laura L. Forrest, John L. Spouge, Mehrdad Hajibabaei, Sujeevan Ratnasingham, Michelle van der Bank, Mark W. Chase, Robyn S. Cowan, David L. Erickson, Aron J. Fazekas, Sean W. Graham, Karen E. James, Ki-Joong Kim, W. John Kress, Harald Schneider, Jonathan van AlphenStahl, Spencer C.H. Barrett, Cassio van den Berg, Diego Bogarin, Kevin S. Burgess, Kenneth M. Cameron, Mark Carine, Juliana Chacón, Alexandra Clark, James J. Clarkson, Ferozah Conrad, Dion S. Devey, Caroline S. Ford, Terry A.J. Hedderson, Michelle L. Hollingsworth, Brian C. Husband, Laura J. Kelly, Prasad R. Kesanakurti, Jung Sung Kim, Young-Dong Kim, Renaud Lahaye, Hae-Lim Lee, David G. Long, Santiago Madriñán, Olivier Maurin, Isabelle Meusnier, Steven G. Newmaster, Chong-Wook Park, Diana M. Percy, Gitte Petersen, James E. Richardson, Gerardo A. Salazar, Vincent Savolainen, Ole Seberg, Michael J. Wilkinson, Dong-Keun Yi, and Damon P. Little. A DNA barcode for land plants. *Proceedings of the National Academy of Sciences of the United States of America*, 106(31):12794–7, 8 2009.

[5] François Chollet. Keras, 2015.

[6] Siang Thye Hang, Atsushi Tatsuma, and Masaki Aono. Bluefield (KDE TUT) at LifeCLEF 2016 Plant Identification Task.

[7] Peterson P et al Jones E, Oliphant E. SciPy: Open Source Scientific Tools for Python, 2001.

[8] Neeraj Kumar, Peter N. Belhumeur, Arijit Biswas, David W. Jacobs, W. John Kress, Ida C. Lopez, and João V. B. Soares. Leafsnap: A Computer Vision System for Automatic Plant Species Identification. pages 502–516. Springer, Berlin, Heidelberg, 2012.

[9] Sue Han Lee, Chee Seng Chan, Paul Wilkin, and Paolo Remagnino. Deep-plant: Plant identification with convolutional neural networks. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 452–456. IEEE, 9 2015.

[10] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary Devito, Zeming Lin, Luca Antiga, Orobix Srl, and Adam Lerer. Automatic differentiation in PyTorch.

[11] R D Stevenson, William A Haber, and Robert A Morris. Electronic Field Guides and User Communities in the Eco- informatics Revolution.

[12] Jana Wäldchen, Michael Rzanny, Marco Seeland, and Patrick Mäder. Automated plant species identification—Trends and future directions. *PLOS Computational Biology*, 14(4):e1005993, 4 2018.

[13] Cong Zhao, Sharon S.F. Chan, Wai-Kuen Cham, and L.M. Chu. Plant identification using leaf shapes—A pattern counting approach. *Pattern Recognition*, 48(10):3203–3215, 10 2015.