Transfer learning models for tree species detection in agroforestry

Erich Trieschman
Stanford University, Department of Statistics
etriesch@stanford.edu

Abstract

This research develops a novel dataset of tree photographs and trains a deep learning model to identify tree species from images in the silvopasture setting. This deep learning model is part of ongoing work, which develops a mobile-based application for tree carbon estimation. In this paper, I compile a novel dataset of over twenty-four thousand labeled tree images and evaluate six transfer learning models.

My dataset is constructed predominantly from image searches for each tree species through the Microsoft Bing search engine. I pair this dataset with images of several other tree databases. I augment my dataset with vertical reflections and random croppings of each image and then filter it with a binary classification model trained to detect trees.

With my selected model, I achieve a Top-1 accuracy of 80.0% on the held-out data. The highest performing model is ResNet-50, retrained on the last convolutional block of the model. I find that this model is better optimized for mobile-based tree species identification than other models such as vision transformers because of its lower model parameter count. The selected model performs best at detecting Loblolly Pine trees, with a precision and recall of 0.911 and 0.941 respectively. The model performs least well in identifying Black Locust, Black Walnut, and Pecan trees, with precision and recall ranges of 0.748-0.768 and 0.718-0.746 respectively.

1. Introduction

Trees are a large carbon pool for our planet. They play a vital role in offseting anthropogenic carbon emissions and attenuating the effects of climate change. We can grow this pool through afforestation and deferred deforestation, increasingly supported by carbon offset payments to forest landowners.

One high-potential opportunity for afforestation is through the transition of pastureland operations to silvopasture operations. Silvopasture is simply the agricultural practice of combining tree cropping with livestock management.

In order for a payment system like this to work, we need an accurate estimate of the amount of carbon stored in trees. Fortunately, decades of researchers have invested into tree growth projections and, today, allometric equations are available for many tree species, which can accurately estimate above and below-ground tree biomass as a function of tree height, tree diameter, species, and local climate.

This research develops a novel dataset of tree photographs and trains a deep learning model to identify tree species from images in the silvopasture setting. This deep learning model is part of ongoing work, which develops a mobile-based application for tree carbon estimation. In the future, photographs from this mobile-based application will be used to augment and improve the models developed in this paper.

This paper compiles a novel dataset of high-fidelity tree photographs, taken in profile perspective. The dataset contains classified images of the 7 most common tree species used in silvopasture in the southeastern US, where pilot projects for this silvopasture transition are underway. I then train a deep learning model to predict tree species from these profile-vew tree photographs.

2. Related work

Image classification is a common problem solved using convolutional neural networks. Beginning in 2012, we have experienced step changes in the performance of neural network models for image classification, first with the introduction of AlexNet and later with further improvements through neural net models like VGGNet, ResNet, and GoogLeNet [8, 12, 13]. These latter models each incremented the field with refinements to parallelization and the ability to exploit backpropagation into deeper and deeper networks.

Within this time period, there has been a plethora of work focused on image classification, from general problems to more specific, with only limited research focused on profile-perspective tree species identification. Below I summarize some of the research focused on this problem and the methods that support it.

In Feng et al.'s work on long-tailed object detection, the team explores training classification models on highly-similar objects, a similar challenge to that conducted in this paper [6]. Feng et al. achieve increased performance via an Equilibrium Loss (EBL) and a Memory-augmented Feature Sampling (MFS) method, improving detection performance.

Carpentier et al. use a self-collected dataset to solve a very similar classification problem to my own: tree detection through bark images [4]. The team trains deep neural network models for this classification task and achieves an accuracy of over 90% in predicting tree species for over 23 different species of trees and tree diameters. Interestingly, the team finds that increasing the number of individual trees in the training database is one of the most important factors to achieving a high accuracy score.

Fricker et al. also attempts to classify trees, however from an aerial perspective instead of the profile perspective taken in this paper [7]. The team uses field-based training data, high spatial resolution airborne hyperspectral imagery, and a convolutional neural network classifier to train a model for future detection using lower-cost input data. Fricker finds significant improvements in the accuracy of his model when using hyperspectral data over simple RGB, color layers I would be excited to incorporate in the future.

3. Methods

This paper's objective is to develop a model that most accurately identifies tree species from photographs curately (i.e., the highest Top-1 accuracy and lowest Top-1 misclassification error). The model balances accuracy against the size of the model. These metrics best align with the downstream use case of the model: tree species identification using a mobile device.

I construct several shallow baseline models and several deep transfer-learning models (see Subsection 3.1). I train all models on a training subset of my novel tree photograph dataset and use a validation subset of this dataset to evaluate the performance of my trained models (see Subsection 3.2). Finally, I use a test subset of this dataset to determine the final performance metrics of my selected model (see Results section below).

3.1. Model construction

I evaluate four baseline classificaiton models. These models provide a standard against which I compare higher-performing convolutional neural networks and transformers. I implement a transfer learning approach for these higher-performing models where I leverage pretrained model weights and focus on retraining only the outermost layers of the model, specific to my novel dataset.

I provide a summary of all models I consider in Table 1.

Model	Loss	Layers/Vers.	Transf. learn.
Softmax	CrossEnt	1	False
SVM	Hinge	1	False
FCN	CrossEnt	2	False
CNN	CrossEnt	3	False
ResNet	CrossEnt	50	True
ConvNext	CrossEnt	"Tiny"	True
Transformer	CrossEnt	12: 16x16pix	True

Table 1. Summary of evaluated models

3.1.1 Baseline models

The baseline models I evaluate are multiclass softmax model, multiclass support vector machine (SVM), two-layer fully connected network (FCN), and 3-layer convolutional network (CNN). The softmax and SVM models provide the highest-level performance metrics for my tree classifier. The the FCN helps me to glean potential gains from including non-linear activation functions in a species detection model, and the CNN helps me to identify potential gains from capturing image structure. I construct all models with Pytorch's "nn.Sequential" module [10].

To construct the softmax model, I first flatten my images, removing all spatial structure, then transform the output to a 7-class dimension with learnable weights. I use a softmax activation function paired with the negative log likelihood loss function to train the model. Together, these are equivalent to the cross entropy loss function, which I use in all other models except SVM. The cross entropy loss function is given by

$$\mathcal{L} = \{l_1, \ldots, l_N\}^T, l_i = \log \frac{\exp x_{i,y_i}}{\sum_{c=1}^{C} \exp x_{i,c}} \tag{1}$$

Where s_{y_i} is the score for the correct class.

To construct the SVM model, I flatten my images, transform the output to a 7-class dimension with learnable weights, and train my model using the multi-margin loss function given by

$$\mathcal{L} = \{l_1, \ldots, l_N\}^T, l_i = \sum_{i=1, i \neq y}^{N} \max(0, x_i - x_{y_i} + 1) \tag{2}$$

To construct the FCN model, I flatten my images and define a single hidden layer of dimension 500 with a ReLU activation function. I transform the output to a 7-class dimension with learnable weights. To construct the CNN model, I preserve the structure of my images and pass them through two convolutions, the first with kernel size of 5 and 32 channels, and next with kernel size of 3 and 16 channels. Both layers use a ReLU activation function, and I pad images so that

the output image shape is preserved throughout the model. Finally I flatten the structured data and transform the output to a 7-class dimension with learnable weights.

3.1.2 Transfer learning models

The first transfer learning model I consider is a deep convolutional neural network with residual connections (ResNet) introduced by He at al. [8]. The original model has 152 layers and relies on a residual learning layers which are easier to train and optimize than conventional feed-forward layers used in my baseline modeling section. This residual learning framework addresses degrading training accuracy in deeper networks by incorporating residual mappings between sets of convolutional layers. Although the original architecture consists of 152 layers, smaller versions of the same network are also available; I choose ResNet-50 because it significantly reduces the number of parameters and may offer faster operation performance on mobile devices.

The next transfer learning model I consider is a vision transformer (ViT) introduced by Dosovitskiy et al. [5]. The original model has 32 layers and 16x16 pixel input patches and relies entirely on attention modules that process smaller, 2D patches of the image, and demonstrating that models without convolutional layers can perform as well or better than models with them. This framework also requires substantially fewer resources to train than a convolutional neural network and can accelerate training times and reduce training computation and economic costs. Smaller versions of the same network are also available. I choose the 12-layer 16x16 pixel input patch model because it significantly reduces the number of parameters and may offer a faster operation performance on mobileprediction performance on mobile devices.

The final transfer learning model I consider is a modernized convolutional network with residual features (ConvNeXt) introduced by Liu et al. [9]. The original model is based on the original ResNet-152 model, but reevaluated at macro and micro levels for similarities and differences with modern-day vision transformers. This framework is meant to provide a high-performing convolutional framework with equal performance to vision transformers. Smaller versions of the same network are also available. I choose the "tiny" version because it significantly reduces the number of parameters and may offer a faster operation performance on mod prediction performance on mobile devices.

I use the PyTorch pretrained model implementations of ResNet, ConvNeXt, and ViT, available through its "models" module [10]. In each transfer learning model, I substitute the last fully connected layer with a two-layer fully connected network. I use a hidden layer dimension of 128 and a ReLU activation function in all transfer learning models. The final layer output is of size 7, the number of species in my target tree-classification model.

3.2. Model training

I begin my model training procedure by randomly splitting my dataset (described in more detail in Section 4) into train, validation, and test datasets. I withold 10% of my data (2,789 images) for final evaluation of my selected model (see Section 5). Out of the remaining dataset, I use 25% (6,274 images) for validation and 75% (18,819 images) for training.

I train all models described above in minibatches of 32 images. I use stochastic gradient descent with Nesterov momentum of 0.9 as my optimization algorithm. Stochastic gradient descent computes gradients on batches of training data and performs gradient updates on all weights. Including Nesterov momentum provides a technique for computing this gradient descent at a point "ahead" of the current location of the weights, which can greatly accelerate convergence [3]. The equations for SGD with a Nesterov accelerated gradient follow

$$v_t = \mu_{t-1} v_{t-1} - \epsilon_{t-1} \nabla f(\theta_{t-1} + \mu_{t-1} v_{t-1})$$
$$\theta_t = \theta_{t-1} + v_t$$

Where θ_t are the model parameters, v_t the velocity, $\mu_t \in [0, 1]$ momentum (decay) coefficient (0.9 in my case) and $\epsilon_t > 0$ the learning rate at iteration t, $f(\theta)$ is the objective function and $\nabla f(\theta)$ the gradient.

I train all baseline models with a learning rate that starts at 0.0001 and decays every two epochs by a factor of 10. I train all baseline models for four epochs and all transfer learning models for 6 epochs.

Last to mention, I train all three transfer learning models under two regimes. First, I freeze the weights of all but the final two-layer fully connected hidden network I add to the training models. In this regime, only this final two-layer network is trained on all images and I leverage the existing weights of the pretrained models for all other layers.

Second, I freeze the weights of all but the additional two-layer fully connected network *and* the final convolutional or decoder block in each transfer learning model. For ResNet I train the fourth and final convolutional block, for ConvNeXt I retrain the 25th and final convolutional network block, and for ViT I retrain the 11th and final encoder module.

4. Dataset and features

This paper develops a novel dataset of high-fidelity tree photographs, taken in profile perspective, labeled with the tree species. The species included are the seven most common species of tree used in silvopasture in the southeastern US: Black Locust, Black Walnut, Honey Locust, Loblolly Pine, Northern Red Oak, Pecan, Chinese Chestnut.

I compile these photographs and labels from images scraped from the internet. I augment my dataset with reflections and random scaled croppings of each image. All original and tranformed images are center-cropped and scaled for use in a deep learning model. Lastly, I filter my dataset using a binary classifier trained to identify profile images of trees. I do so to ensure that the image dataset scraped from the internet has a high likelihood of only containing trees. I provide sample images from this final dataset in Figure 1 and I provide full detail of how this novel dataset is constructed in the subsections below.

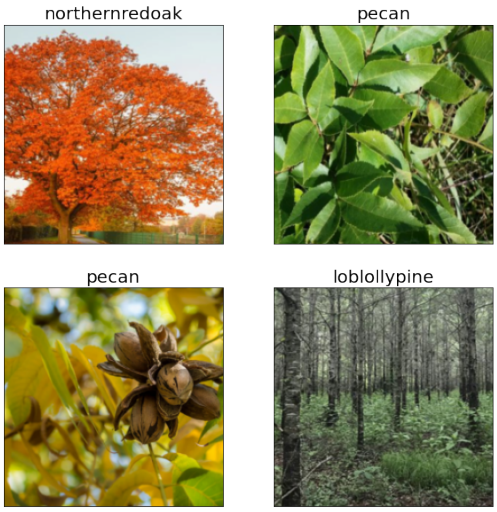

Figure 1. Sample images from final dataset

4.1. Image scraping

I use three data sources to generate a dataset of tree images: the Harvard Arboretum Plant Image database [1], the Arbor Day Foundation website [2], and the results of Microsoft Bing image searches for each tree species of interest. I also evaluated BarkNet [4], a dataset of close-up photographs of tree bark supporting a tree-bark identification model, however, this dataset did not contain any of the species of interest in this project.

I develop website-specific python codes to scrape and download images of each tree species from each source. I download all available images from the Harvard Arboretum and Arbor Day sources for each of the species of interest. And for the results of my Bing Image searches, I download all of the first 750 images that were not protected from download access. I use Bing image search instead of Google image search because of recent changes in Google's website layout that make web scraping more difficult.

This results in a preliminary dataset of 7,707 images, detailed in Table 2.

Species (N)	Total	Bing	Arb. Day	Harvard
Black Locust	605	562	0	43
Black Walnut	651	563	4	84
Honey Locust	501	501	0	0
Loblolly Pine	495	488	7	0
North. Red Oak	579	531	7	41
Pecan	611	591	6	14
Chinese Chestnut	662	467	3	192
Total	7707	6905	54	748

Table 2. Number of images, by data source and species

4.2. Dataset augmentation

I augment the dataset with a reflection of each image about its vertical axis. I do not include horizontal and diagonal reflections because all tree images classified with this model are expected to be up-right. I also augment the dataset with the original and mirror image of a random cropping of each image at 3 different relative sizes: 10%, 25%, and 50%.

In total, this augmentation increases my dataset by a factor of eight and results in a dataset of 32,832 images.

4.3. Image cropping, scaling, and transforming

I center and scale all images to the same shape and size for use in a deep learning model. I first upscale all images so that the minimum dimension of each image (height or width) is 1024 pixels. I next center-crop each image to a 1024x1024 pixel square. And finally I downscale all images to 224x224 pixels and normalize them based on the mean and standard deviation of the datset used by Pytorch developers to pretrain their available models [1]. These transformations are implemented so that this image dataset follows the common dimensions and normalizations used by the pretrained models available for use through Pytorch [10].

4.4. Dataset filtering

Lastly, I filter my dataset to only include, with high likelihood, profile-perspective photographs of trees. I choose to filter my dataset because the method I use to construct my dataset introduces erroneous images with somewhat high likelihood. I confirmed this with manual inspection, observing errors like drawings of trees or images loosely related to trees (e.g., a table made from the wood of a black walnut tree). The former is more of a concern in the Harvard Arboretum and Arbor Day Foundation datasets, and both the former and latter are concerns in the results of Bing image searches. My dataset also includes random croppings of all images, which may not include any component of a tree and

[1] RGB means = [0.485, 0.456, 0.406]; RGB standard deviations = [0.229, 0.224, 0.225]

I choose to filter these croppings too.

There are no publicly available binary classification models for tree images, so I use transfer learning to leverage the pretrained weights of ResNet50 and construct a binary classifier [8] specific to this research. I do so by replacing the final fully connected layer of ResNet50 with the same fully connected layers and pooling described in the methods section above.

To train these final layers of the pretrained ResNET model, I create a separate dataset of tree and not-tree images, labeled as such, by scraping Bing for select search terms. I include images of "tree photograph", "tree bark photograph", and "tree leaf photograph" in the tree class, and I include images of "tree drawing", "table", "hand", "map", and "diagram" in the not-tree class; I determine these not-tree search terms with a manual review of a sample of observations of my original tree species dataset. I transform these images with the same scaling and normalizations as described above. With ten epochs of training, this binary classifier achieves 99.06% accuracy on the validation dataset.

I filter my tree species dataset by running this classifier on my images and only keeping those images whose prediction probability for the tree class exceeded a threshold of 85%. I include a histogram of tree class predictions for all images in my database in Figure 2. This filtering results in a final dataset of 24,944 images, 75.97% of my original augmented dataset.

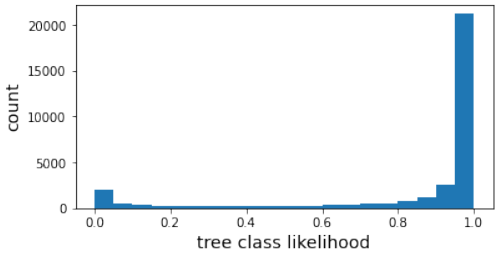

Figure 2. Histogram of tree class likelihoods

Details of my final tree species dataset are presented in Table 3 by tree species and image source.

5. Results and discussion

In this section, I evaluate the models described above and my final model selection. I then evaluate my selected model on my withheld test dataset with both qualitative and quantitative metrics. I also include a brief description of the evaluation metrics I use for my selected model.

5.1. Training results and model selection

All models I trained achieved maximum training and validation accuracy before their final training epoch. This gives me confidence that they have achieved at least local maxima, and also that I am not overfitting these models, as I find that the training accuracy does not significantly exceed validation accuracy, especially for the transfer learning models.

I find that the ResNet model, trained on the final CNN block achieves highest validation accuracy accross my training regime; I select this model for future production within tree carbon quantification apps. While ViT also performs well, its storage costs are almost four times as high as the other comparably-performing models, which may pose a limitation in the mobile app setting.

My training results are presented in Table 4.

Species (N)	Total	Bing	Arb. Day	Harvard
Black Locust	3587	3325	0	262
Black Walnut	3845	3327	23	495
Honey Locust	2925	2925	0	0
Loblolly Pine	3106	3059	47	0
North. Red Oak	3535	3236	42	257
Pecan	3875	3754	42	79
Chinese Chestnut	4071	2905	17	149
Total	24944	22531	171	2242

Table 3. Final tree species image dataset

Model	Acc. train	Acc. val	N params
Softmax	0.233	0.220	1,053,696
SVM	0.278	0.196	1,053,696
FCN	0.284	0.264	19,268,480
CNN	0.406	0.327	5,620,656
ResNetFC	0.571	0.561	25,557,032
ResNet	**0.841**	**0.825**	**25,557,032**
ConvNextFC	0.612	0.608	28,589,128
ConvNext	0.657	0.649	28,589,128
ViTFC	0.634	0.615	86,567,656

Table 4. Model accuracy and storage size. The suffix "FC" denotes transfer learning models where only the final fully connected layer was learned

5.2. Final model evaluation

5.2.1 Evaluation metrics

Top-K accuracy scores refer to the percent of images for which the correct class was within the top K predicted scores. As my classifier only considers the seven common species of trees in agroforestry, I only consider small Ks, specifically Top-1 and Top-2 accuracy.

Precision, recall, and the F1 score provide alternative methods of evaluating the predictive power of a model, capturing the relationship between the true positives, false positives, and false negatives produced by a model. They are

defined as follows

$$\text{Precision} = \frac{TP}{TP + FP}$$
$$\text{Recall} = \frac{TP}{TP + FN}$$
$$\text{F1} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

I report these scores for each class, and also consider the weighted average scores across all classes. A helpful way to visualize the components of these scores is with a confusion matrix. In a confusion matrix, each row of the matrix represents the instances in an actual class while each column represents the instances in a predicted class. I present a confusion matrix in my results as well.

Lastly, I implement methods developed by Simonyan et al [11] to qualitatively visualize each of my classes as represented by the model. I do so for each class through gradient *ascent*, generating an image that the network will recognize as the target class. This process follows the following optimization

$$I^* = \arg\max_I (s_y(I) - R(I))$$

where $s_y(I)$ is the score from a deep learning model for image I and $R(I)$ is a regularizer (e.g., $\lambda \mid\mid I \mid\mid_2^2$).

5.2.2 Quantitative evaluation results

I now present the evaluation metrics for my final selected model: ResNet-50 with the final CNN block retrained. This model achieves an 80.0% Top-1 accuracy and a 92.7% Top-2 accuracy. In a field setting, this implies that farmers submitting photographs for species identification could classify their images correctly 80 out of 100 times, and could narrow their choice to two tree species 92 out of 100 times.

Next, I consider the results of class-wise precision and recall scores, presented in Table 5. Here I observe that Loblolly Pine trees have highest class precision and recall (0.911 and 0.941 respectively), which might be explained by the distinctive pine needles of these trees, quite different than the leaf-bearing trees in my dataset. Black Locust, Black Walnut, and Pecan trees all share similarly low precision and recall (0.748-0.768 and 0.718-0.746 respectively). In weighted average, this model achieves precision and accuracy scores of 0.800 and 0.800 respectively.

I also evaluate the results of a confusion matrix for these predictions in Table 6. Here I observe the same trends described above, but can identify which trees are being misclassified by which other trees. The only strong misclassification trend that I observe is between Black Locust trees and Honey Locust trees: almost 15% of Honey Locust trees are mislabeled as Black Locust trees. As the names suggest, this missclassification is likely justified by shared attributes between Honey and Black Locust trees; alternatively, this misclassification may simply suggest issues with my underlying search-image-based dataset.

	precision	recall	f1-score	support
BL	0.748	0.764	0.756	377.0
BW	0.768	0.751	0.759	401.0
C	0.854	0.867	0.860	398.0
HL	0.782	0.714	0.747	301.0
LP	0.911	0.941	0.926	357.0
O	0.770	0.840	0.804	307.0
P	0.758	0.718	0.737	354.0
weighted avg	0.800	0.800	0.799	2495.0

Table 5. Precision and recall for ResNet-50-predicted images from test dataset **Abbreviations:** BL=Black Locust, BW=Black Walnut, HL=Honey Locust, LP=Loblolly Pine, O=Northern Red Oak, P=Pecan, C=Chinese Chestnut

	BL	BW	C	HL	LP	O	P
BL	0.76	0.04	0.02	0.09	0.02	0.04	0.02
BW	0.04	0.75	0.05	0.03	0.01	0.04	0.07
C	0.03	0.03	0.87	0.01	0.01	0.02	0.04
HL	0.14	0.03	0.01	0.71	0.01	0.06	0.04
LP	0.01	0.01	0.01	0.00	0.94	0.01	0.01
O	0.03	0.05	0.01	0.01	0.02	0.84	0.04
P	0.03	0.10	0.06	0.02	0.02	0.04	0.72

Table 6. Confusion matrix for ResNet-50-predicted images from test dataset. **Abbreviations:** BL=Black Locust, BW=Black Walnut, HL=Honey Locust, LP=Loblolly Pine, O=Northern Red Oak, P=Pecan, C=Chinese Chestnut

5.2.3 Qualitative evaluation metrics

To assist with visual interpretability and to qualitatively understand the performance of my final model, below I include images with the highest score for the correct class in Figure 3, and images with the lowest score for the correct class in Figure 4. From the highest-scoring images, it is clear how my model can identify some of the key characteristics of specific tree species, like acorns, which appear in the Northern Red Oak tree image, and chestnuts, which appear in the Chinese Chestnut tree photograph. From the lowest-scoring images, I observe non-tree features such as a mushroom and a highly blurred image of a treetop.

I also examine synthetic visualizations for each class in Figure 5, as described above. Here I can gleen defining features for each tree class in my model. For example, I can observe oak leaves and acorns in the Northern Red Oak class

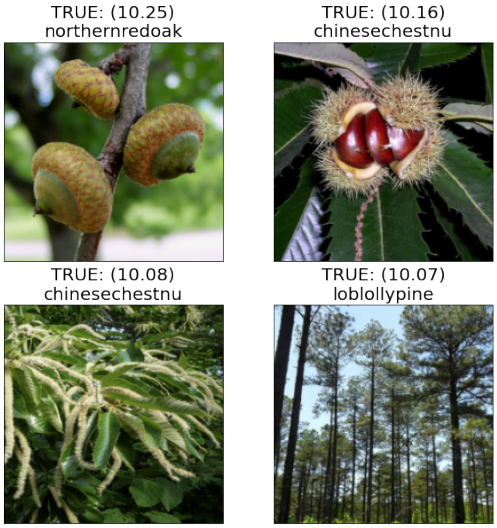

Figure 3. Images with highest scores for the correct class

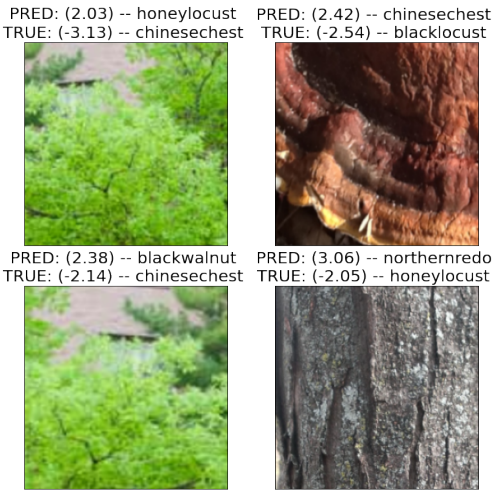

Figure 4. Images with lowest scores for the correct class

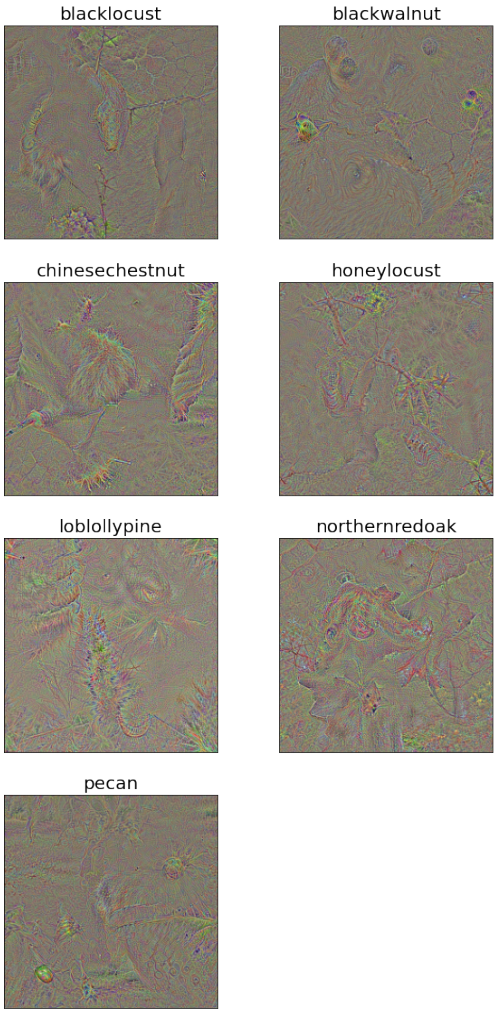

Figure 5. Synthetic visualizations for each class

synthetic image, pine needles in the Loblolly Pine class synthetic image, locust pods in the Honey Locust class synthetic image, chestnuts in the Chinese Chestnut class sythetic image, and pecans in the Pecan class synthetic images.

6. Conclusion and future work

In this paper I train a tree species classifier to support carbon measurements in the agroforestry setting. I compile a novel dataset of over 24k labeled tree images and evaluate six transfer learning models for highest performance. My dataset is constructed from scraped web images, which I augment with vertical reflections and random croppings of each image and finally filtered with a binary classification model trained to detect trees.

I achieve a Top-1 accuracy of 80.0% on held-out data by applying transfer learning to the ResNet-50 model, training the entire last convolutional block of the model. I find this model to be better optimized for mobile-based tree species identification than other models because of its (relatively) smaller model parameter count. I find this final model performs best at detecting Loblolly Pine trees with a precision and recall of 0.911 and 0.941 respectively. The model performs least well in identifying Black Locust, Black Walnut, and Pecan trees with precision and recal ranges of 0.748-0.768 and 0.718-0.746 respectively.

As next steps in this research, I intend to supplement my dataset with higher fidelity images, which will come from the mobile application leveraging this model. The majority of my training data comes from image searches on the internet so I suspect that increasing the share of higher quality images will likely improve Top-1 accuracy of my models.

I also plan to apply the transfer learning techniques developed above on larger models like ResNet-152 and to retrain more of the convolutional or encoder layers of the model. I was able to achieve nearly a 20 percentage point increase in Top-1 accuracy for ResNet by retraining one layer deeper into the model and would like to explore how much more performance I could achieve by going deeper still.

7. Contributions and acknowledgements

This research is solo authored by Erich Trieschman and conducted entirely within the context of the Stanford course CS231N taken in the Spring Quarter of 2022. I leverage a few utility functions from homework assignments from this class as well as from tutorials provided by Pytorch [10].

All of my code, results, and the pretrained models are available in a project repo on my GitHub, available at https://github.com/etrieschman/tree-finder

References

[1] *Harvard Arboretum, ArbPIX: Plant Image Search*, Accessed 2022 at http://arboretum.harvard.edu/plants/image-search/. 4

[2] *Arbor Day Foundation, Shop For Trees*, Accessed 2022 at https://shop.arborday.org/nursery. 4

[3] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu. Advances in optimizing recurrent networks. *CoRR*, abs/1212.0901, 2012. 3

[4] M. Carpentier, P. Giguère, and J. Gaudreault. Tree species identification from bark images using convolutional neural networks, 2018. 2, 4

[5] A. Dosovitskiy, L. Beyer, and A. K. et al. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020. 3

[6] C. Feng, Y. Zhong, and W. Huang. Exploring classification equilibrium in long-tailed object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3417–3426, October 2021. 2

[7] G. A. Fricker, J. D. Ventura, J. A. Wolf, M. P. North, F. W. Davis, and J. Franklin. A convolutional neural network classifier identifies tree species in mixed-conifer forest from hyperspectral imagery. *Remote Sensing*, 11(19), 2019. 2

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015. 1, 3, 5

[9] Z. Liu, H. Mao, C. Wu, C. Feichtenhofer, T. Darrell, and S. Xie. A convnet for the 2020s. *CoRR*, abs/2201.03545, 2022. 3

[10] A. Paszke and e. a. Gross. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 2, 3, 4, 8

[11] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Workshop at International Conference on Learning Representations*, 2014. 6

[12] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition, 2014. 1

[13] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions, 2014. 1