Transfer learning models for tree species detection in agroforestry

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

Abstract

Section 0: Abstract (1-2 paragraphs) The abstract is required. It should contain a 1-2 sentence summary of each major section of the paper.

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, oftentimes supported through carbon offset payments by sectors emitting carbon to the landowners growing these forests.

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 a highly accurate estimate of the amount of carbon stored in trees. Fortunately, decades of research has been invested into tree growth projections and today, allometric equations are readily 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. These tools will be combined with a broader suite of tools being developed to accurately estimate tree carbon from phone-based tree height and diameter measurements to enable carbon payments on agricultural operations transitioning to silvopasture. In the future, these phone-based measurements will be used to augment and improve the models developed in this paper.

Specifically, in this paper I compile 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

(0.5-1 page): You should find existing papers, group them into categories based on their approaches, and talk about exemplary ones in each category: Discuss strengths and weaknesses. In your opinion, which approaches were clever/good? What is the state-of-the-art? Do most people perform the task by hand? You should aim to have at least 10 references in the related work. Include previous attempts by others at your problem, previous technical methods, or previous learning algorithms.

Image classification is a common problem solved using convolutional neural networks. [[DESCRIPTIONS OF IMAGENET, ALEXNET, VGGNET, GOOGLENET, AND RESNET]] [8, 11, 12].

[[PAPER]] Expanded the possibilities of using pretrained models in the described frameworks for use in more targeted settings. [[SUMMARY OF APPROACH]]. [[SUMMARY OF FINDINGS]].

In Feng et al.'s work on long-tailed object detection, they explore training classification models on highly-similar objects, a similar challenge to that conducted in this paper [6]. [[SUMMARY OF APPROACH]]. [[SUMMARY OF FINDINGS]]

Carpentier et al. use a self-collected dataset to a very similar problem: tree detection through bark [4]. [[SUMMARY OF APPROACH]]. [[SUMMARY OF FINDINGS]].

Fricker et al. also attempts to classify trees, however from an aerial perspective instead of the profile perspective taken in this paper [7]. [[SUMMARY OF APPROACH]]. [[SUMMARY OF FINDINGS]]

3. Methods

The objective of this paper is to develop a model with highest Top-1 accuracy (lowest Top-1 misclassification error) for the identification of tree species from tree photographs, while also balancing the size of the model. These

metrics best align with the downstream use case of the model, which is to assist in the proper identification of agroforestry trees through a mobile-based app.

To achieve a model with highest Top-1 accuracy, 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 in this report. These models serve as a standard against which I can aim to improve Top-1 accuracy through the use of deep convolutional neural networks. I implement the remaining models in this research through a transfer learning approach, where I leverage pretrained model weights of existing, high-performing image classification models, and focus only on relearning the outermost layers of the model, trained specifically to my novel dataset.

I provide a summary of these models in Table 1 and describe them in more detail in the sections below.

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	XXX	True

Table 1. Summary of evaluated models

3.1.1 Baseline models

The models I consider are a multiclass softmax model, a multiclass support vector machine model (SVM), a two-layer fully connected network (FCN), and a 3-layer convolutional network (CNN). The softmax and SVM models serve as the highest-level attempts at class identification, while the FCN was meant to reveal potential gains from including non-linear activation functions, and the CNN to reveal 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 then use a softmax activation function paired with the negative log likelihood loss function. Together, these are equivalent to the cross entropy loss function, which I use in all but the SVM model. 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 then 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 output layer output is of size 7, the number of species in my target tree-classification model.

3.2. Model training

In this section I describe my model training process and the set-up for my model selection. I begin 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 with those batch-computed gradients. 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, additional 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 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 convolutional network block, and for ViT I retrain the 11th 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 augment my dataset with reflections and random scaled croppings of each image. All original and tranformed images are then center-cropped and scaled for use in a deep learning model. Lastly, the dataset is filtered using a binary classifier trained to identify profile images of trees to ensure that the image dataset scraped from the internet has a high likelihood of only containing trees. I provide sample images from this dataset with their tree likelihood in Figure 1 and I provide full detail of how this novel dataset is constructed in the subsections below.

Figure 1. Four sample images with species labels 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, 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 developed website-specific python codes to scrape and download images of each tree species from each source. I downloaded 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 downloaded 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 scraped from each data source, by 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

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].

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

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 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 determined this 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.

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 discuss my results. I start with an evaluation of the models described above and my final model selection. I then proceed with an evaluation of 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.

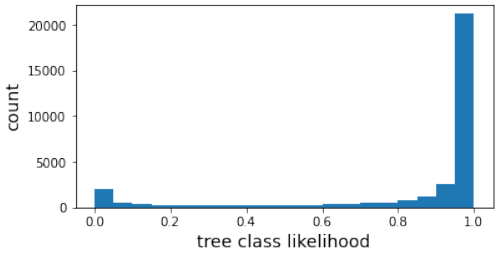

Figure 2. Histogram of tree class likelihood in tree species augmented dataset

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, after augmentation and filtering, by species and source

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 [[MODEL]] highest validation accuracy accross my trianing regime; I select this model for my final evaluations and for future production within tree carbon quantification apps. While ViT also performs well, its storage costs is almost four times as high as the other comparably-performing models and this may pose a limitation in the mobile app setting. My results are presented in Table 4

Model	Acc. train	Acc. val	N params
Softmax			1,053,696
SVM			1,053,696
FCN			19,268,480
CNN			5,620,656
ResNetFC			25,557,032
ResNet			**25,557,032**
ConvNextFC			28,589,128
ConvNext			28,589,128
ViT			86,567,656

Table 4. Model accuracy and storage size

5.2. Final model evaluation

5.2.1 Evaluation metrics

Top1 and Top2 accuracy, Confusion matrix, Precision/recall, class visualization

5.2.2 Quantitative evaluation results

I now present the evaluation metrics for [[MODEL]], the final model I select through this research. [[mODEL]] achieves a XXXX Top-1 accuracy and a XXXX Top-2 accuracy. In a field setting, this implies that farmers using this tooling could identify their photographed trees XX out of 100 times, and could narrow their choice to two tree species XX out of 100 times.

Next I consider the results of a confusion matrix for these predictions in Table 5. Here I observe that [[MODEL]] is most effective at identifying [[species]] and least at identifying [[species]]. I also consider the results of class-wise precision and recall scores, presented in Table 6. Here I observe FINDINGS

	BL	BW	HL	LP	O	P	C
BL	3587	3325	0	262			
BW	3845	3327	23	495			
HW	2925	2925	0	0			
LP	3106	3059	47	0			
O	3535	3236	42	257			
P	3875	3754	42	79			
C	4071	2905	17	149			

Table 5. Confusion matrix for [[MODEL]]-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

	Precision	Recall	F2
BL	3587	3325	0
BW	3845	3327	
HW	2925	2925	
LP	3106	3059	
O	3535	3236	42
P	3875	3754	42
C	4071	2905	17
Overall	24944	22531	171

Table 6. Precision and recall [[MODEL]]-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

Highest-scoring images and their scores Lowest scoring images and their scores Class visualizations

6. Conclusion and future work

Section 6: Conclusion/Future Work (1-3 paragraphs)

Summarize your report and reiterate key points. Which algorithms were the highest-performing? Why do you think that some algorithms worked better than others? For future work, if you had more time, more team members, or more compute, what would you explore?

7. Appendices

Section 7: Appendices

All the text in sections before this point must fit on eight (8) CVPR-style pages or less. Extra figures can go beyond the limit, but please do not put all your figures at the end just to fit the page limit. TAs will also be focusing their attention largely on the previous sections, so anything critical to your project should go in those sections, if possible. (If you are unsure of something, please feel free to make a private Ed post).

This section is optional. Include additional derivations of proofs which weren't core to the understanding of your proposed algorithm. Usually, you put equations or other details here when you don't want to disrupt the flow of the main paper.

8. Contributions and acknowledgements

Section 8: Contributions Acknowledgements (not part of page limit)

In this section, you must explicitly state what each person on your team did for the project. If you made use of public code (e.g. from GitHub), please provide a link to the original repo. Additionally, you must mention any non-CS231N collaborators and include a brief sentence on what they did for your project. See the AlphaGo paper's contributions statement for an example. If you're part of a research lab and made use of their job scheduling, containerization, or GPUs, briefly include a sentence description on this as well.

9. References

References/Bibliography (No page limit): This section should include citations for: (1) Any papers mentioned in the related work section. (2) Papers describing algorithms that you used which were not covered in class. (3) Code or libraries you downloaded and used. This includes libraries such as scikit-learn, TensorFlow, PyTorch, etc.

References

[1] *Harvard Arboretum, ArbPIX: Plant Image Search*, Accessed 2022 at http://arboretum.harvard.edu/plants/image-search/. 3

[2] *Arbor Day Foundation, Shop For Trees*, Accessed 2022 at https://shop.arborday.org/nursery. 3

[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. 1

[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. 2

[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. 1

[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. 1

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015. 1, 2, 4

[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 H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 2, 3, 4

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

[12] 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