Using CNNs and Hyperparameter-Optimized Boosting Techniques to Predict Vital Plant Traits

Prem Patel
School of Computer Science
University of Waterloo
Waterloo, ON, N2L 3G1
p352pate@uwaterloo.ca
August 9, 2024

Abstract

A pretrained Resnet50 convolutional model was used to extract 2048 key photographic features of plant photographs, captured through citizen science. Pairing this with 163 ancillary features allowed for a rich feature set, which was then pipelined into a tuned boosting algorithm (XGBoost) to accurately predict target features of unseen images. Ultimately, this resulted in an average R^2 test score of approximately 0.3 across the six target features, showcasing successful model performance. See GitHub for the final source code used to achieve this task, using libraries such as PyTorch for ResNet feature extraction, XGBoost for boosting, and scikit-optimize for Bayesian hyperparameter tuning.

1 Introduction

The primary task was to predict plant properties (traits) using plant photographs along with supporting numerical data. A large training database of 43,363 images was made available, along with 6,391 test images to evaluate model performance. The accuracy of these model predictions were evaluated using an R2 score, with 6 target variables to be predicted. With access to this large dataset, it was necessary to employ a rich feature extraction method, i.e. ResNet50. To produce as robust a model as possible, it was also necessary to utilize the supporting numerical data. This started off as a combined neural network using both the ResNet model (with the final connected layer removed) and ancillary data as inputs. After thorough exploration of various architectures and a re-examination of the problem, it was clear that a boosting strategy was more desirable. This resulted in a final model that utilized XGBoost, an effective boosting algorithm to combine many weak learners to create a more robust model for trait prediction.

Figure 1: Example of plant photograph used for model training

2 Related Works

Given the photographic nature of the training data, it was natural to consider using a convolutional network in some capacity. This approach has been used in the past, where a fully connected ResNet CNN was used to predict plant traits from similar photographic data, with bioclimatic data being fed into the model as well (Schiller et al., 2021). This work was done by Christopher Schiller, Sebastian Schmidtlein, Coline Boonman, Alvaro Moreno-Martínez & Teja Kattenborn.

A key difference from this model is the removal of the final layer, as the ResNet model was simply used as a means of extracting features in this case, rather than generating final predictions. As such, ancillary data was concatenated and then fed to the XGBoost algorithm, generating predictions on a trait-by-trait basis. Another key difference is that hyperparameter tuning was utilized to optimize the performance of this boosting strategy.

The key limitation of the past approach is that it is comparatively less robust and heavily dependent on data preprocessing to achieve positive test results. With one network responsible for predicting 6 differing traits, along with features with greatly varied scales, a boosting approach was deemed to be more appropriate as it would iterate through each target trait and use a decision-tree approach to decide which features are more relevant to a particular trait.

See below for the pseudocode outlining the XGBoost algorithm methodology in determining optimal decision tree splits (Chen & Guestrin, 2016).

```
Algorithm 1: Exact Greedy Algorithm for Split Finding
Input: $I$, instance set of current node
Input: $d$, feature dimension
1 $gain \leftarrow 0$;
2 $G \leftarrow \sum_{i \in I} g_i, \quad H \leftarrow \sum_{i \in I} h_i$;
3 for $k = 1$ to $m$ do
4     $G_L \leftarrow 0, \quad H_L \leftarrow 0$;
5     for $j$ in sorted($I$, by $x_{jk}$) do
6         $G_L \leftarrow G_L + g_j, \quad H_L \leftarrow H_L + h_j$;
7         $G_R \leftarrow G - G_L, \quad H_R \leftarrow H - H_L$;
8         $score \leftarrow \max\left(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda}\right)$;
Output: Split with max score
```

3 Main Results

See below for pseudocode detailing the entire learning pipeline, as implemented fully at this GitHub link. Design decisions and relevant insights into model adaptation will then be explained.

Algorithm 2: Plant Trait Prediction Pipeline (Simplified)

```
Input: $train\_data$, Numerical training data
Input: $test\_data$, Numerical test data
Input: $images\_train$, Training images
Input: $images\_test$, Test images
Input: $tuning\_flag$, Hyperparameter tuning method ('G', 'B', or 'N')
Output: $submission\_file$, Predictions file
1  1. Define Data Transformations ;
2  Function GetTransforms():
3  2. Preprocess Data ;
4  Function Preprocess($train\_data, test\_data$):
5  3. Prepare DataLoaders ;
6  Function CreateDataLoaders($images\_train, images\_test$):
7  4. Extract Features ;
8  Function ExtractFeatures($train\_loader, test\_loader$):
9  5. Combine Features ;
10 Function CombineFeatures():
11 6. Set Up XGBoost and Tuning Models ;
12 Function InitializeXGBoost():
13 7. Train and Tune XGBoost Model for Each Target Trait ;
14 Function TrainAndTuneModels():
15 if $tuning\_method$ = 'G' then
16     Function GridSearch():
17 else if $tuning\_method$ = 'B' then
18     Function BayesianSearch():
19 else
20     Function XGBoostManual():
21 8. Generate Predictions ;
22 Function Predict($test\_data$):
23 9. Save Results ;
24 Function SavePredictions($submission\_file$):
25 10. Notify Results: ;
26 Function SendEmail($submission\_file$):
```

3.1 Problem Formulation

The problem of predicting plant traits from photographs was framed as a supervised regression problem. The challenge was to map complex visual features from plant images to numerical target traits, leveraging both image-based features and ancillary data. Mathematically, this can be represented as learning a function $f : X \to Y$, where X is the combined feature space of image features and ancillary data, and Y is the set of target traits. The goal was to find a function that minimizes the mean squared error between predicted and actual trait values across six target traits.

3.2 Preliminary Models and Preprocessing

Initial experiments involved designing a fully connected neural network that integrated features from a convolutional neural network (CNN) and ancillary data. Ancillary preprocessing steps included log transformation of target values to stabilize variance, followed by min-max normalization of both training and test datasets, to ensure uniform scaling. Outlier removal consisted of eliminating data points exceeding three standard deviations from the mean, to mitigate their impact on model performance. Images were also preprocessed through 224x224 pixel resizing, normalization and tensor conversion. Training images were also augmented at random, using flipping and colour jitter. Despite these efforts, preliminary models were prone to overfitting and achieved an average R^2 score below 0.2 on unseen test images. Regularization techniques including early stopping, batch normalization and dropout were used but did not significantly improve model generalization.

3.3 Boosting and Hyperparameter Tuning

A major improvement came with the incorporation of XGBoost, a boosting algorithm known for its robustness and performance. Preprocessing and augmentation remained the same as above, now with the CNN and ancillary data being used strictly as feature inputs to the boosting process. Bayesian hyperparameter tuning was used to optimize the model, exploring a broad search space that balanced exploration with practical computational constraints. This approach led to a significant improvement, with the final model achieving an average R^2 score of 0.3. Hyperparameter tuning involved fine-tuning parameters like learning rate, number of estimators, and tree depth, which were crucial for improving model accuracy.

3.4 Ablation Studies

Various configurations were tested to assess their impact on model performance. Different preprocessing methods (e.g., log-transforming targets vs. all training data), normalization techniques (z-score vs. min-max), and architecture choices (e.g., ResNet18 vs. ResNet50) were compared. The effectiveness of combining CNN features with ancillary data versus using them separately was also evaluated. The switch to a CNN-XGBoost pipeline was motivated by the need for a more robust solution, where tuning could be used to optimize model performance. Various tuning methods were implemented (Bayesian search, Grid search, no tuning) for further ablation.

4 Conclusion

Experiments indicated that a hybrid approach combining CNN and ancillary feature extraction with tuned XGBoost for regression can significantly enhance prediction accuracy for plant traits. The key takeaway is the importance of choosing the right architecture and preprocessing strategy suited to the data. Observations revealed that preprocessing choices, such as log transformation and normalization, significantly influenced performance of the preliminary, fully connected model. Many of these choices had a negligible impact for boosting. In retrospect, starting with a robust boosting approach from the beginning would have allowed for deeper exploration of model variations, while reducing the need for excess testing of preprocessing techniques.

4.1 Future Directions

Constraints of time and computational resources were notable, impacting the breadth of exploration. With additional resources, expanding the search space for hyperparameter tuning could yield even better results. Exploring more sophisticated feature extraction techniques might further enhance model performance, such as applying principal component analysis to reduce dimensionality. Balancing computational feasibility with model complexity remains a crucial consideration in machine learning projects.

Acknowledgement

Thank you to Christopher Schiller, Sebastian Schmidtlein, Coline Boonman, Alvaro Moreno-Martínez & Teja Kattenborn for providing initial inspiration, as well as Yao-Liang Yu for providing the academic inspiration necessary in producing this work.

References

Chen, T. and C. Guestrin (2016). "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. ACM.

PyTorch (2024). "resnet50: Models and pre-trained weights". Accessed: 2024-08-01.

Schiller, C., S. Schmidtlein, C. Boonman, A. Moreno-Martínez, and T. Kattenborn (2021). "Deep learning and citizen science enable automated plant trait predictions from photographs". *Scientific Reports*, vol. 11.

XGBoost Contributors (2018). "XGBoost Parameters". Accessed: 2024-08-05.

Yu, Y. (2024). "CS480/680: Introduction to Machine Learning". Accessed: 2024-07-26.