

---

# 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

Submission Date: August 13

## 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  $R^2$  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

## 22 2 Related Works

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

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

33 The key limitation of the past approach is that it is comparatively less robust and heavily dependent  
 34 on data preprocessing to achieve positive test results. With one network responsible for predicting  
 35 6 differing traits, along with features with greatly varied scales, a boosting approach was deemed to  
 36 be more appropriate as it would iterate through each target trait and use a decision-tree approach to  
 37 decide which features are more relevant to a particular trait.

38 See below for the pseudocode outlining the XGBoost algorithm methodology in determining optimal  
 39 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
40 4  $G_L \leftarrow 0, \quad H_L \leftarrow 0$ ;
5   for  $j$  in  $sorted(I, \text{by } x_{jk})$  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

---

### 41 3 Main Results

42 See below for pseudocode detailing the entire learning pipeline, as implented fully at this GitHub  
43 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 DefineTransformationsTrainTest():
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():
4411 6. Set Up XGBoost Model ;
12 Function InitializeXGBoost():
13 7. Train and Tune XGBoost Model for Each Target Trait ;
14 Function TrainAndTuneModel():
15 if tuning_method = 'G' then
16 | Function GridSearchTuning():
17 else if tuning_method = 'B' then
18 | Function BayesianSearchTuning():
19 else
20 | Function NoTuning():
21 8. Generate Predictions ;
22 Function Predict(test_data):
23 9. Save Results ;
24 Function SaveToFile(submission_file):
25 10. Notify Results ;
26 Function SendEmailNotification(submission_file):
```

---

#### 45 3.1 Problem Formulation

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

## 52 3.2 Preliminary Models and Preprocessing

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

## 63 3.3 Boosting and Hyperparameter Tuning

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

## 72 3.4 Ablation Studies

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

## 80 4 Conclusion

81 Experiments indicated that a hybrid approach combining CNN and ancillary feature extraction with  
82 tuned XGBoost for regression can significantly enhance prediction accuracy for plant traits. The  
83 key takeaway is the importance of choosing the right architecture and preprocessing strategy suited  
84 to the data. Observations revealed that preprocessing choices, such as log transformation and nor-  
85 malization, significantly influenced performance of the preliminary, fully connected model. Many  
86 of these choices had a negligible impact for boosting. In retrospect, starting with a robust boosting  
87 approach from the beginning would have allowed for deeper exploration of model variations, while  
88 reducing the need for excess testing of preprocessing techniques.

## 89 4.1 Future Directions

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

## 96 Acknowledgement

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

## 100 References

- 101 Chen, T. and C. Guestrin (2016). “XGBoost: A Scalable Tree Boosting System”. In: *Proceedings*  
102 *of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.  
103 KDD ’16. ACM.
- 104 Contributors, X. (2018). “XGBoost Parameters”. Accessed: 2024-08-05.
- 105 PyTorch (2024). “resnet50: Models and pre-trained weights”. Accessed: 2024-08-01.
- 106 Schiller, C., S. Schmidlein, C. Boonman, A. Moreno-Martínez, and T. Kattenborn (2021). “Inte-  
107 grating deep learning with plant trait prediction models”. *Scientific Reports*, vol. 11.
- 108 Yu, Y. (2024). “CS480/680: Introduction to Machine Learning”. Accessed: 2024-07-26.