Multioutput Regression for Plant Traits

Barry Zhang
School of Computer Science
University of Waterloo
Waterloo, ON, N2L 3G1
b442zhan@uwaterloo.ca
report due: August 12

Abstract

The goal of this project is to is to predict 6 plant traits given an image of a plant and values for 163 ancillary traits. The 6 output traits to be predicted are "Stem specific density", "Leaf area per leaf dry mass", "Plant height", "Seed dry mass", "Leaf nitrogen (N) content per leaf area", and "Leaf area". These traits are labelled as “X4”, “X11”, “X18”, “X26”, “X50”, and “X3112”, respectively. To make predictions, I created an ensemble model that takes the weighted average of the predictions from 4 models. Performance was measured based on the R^2 value, and it was found that the weighted average ensemble produced a better performance on the validation set than each model individually. The ensemble produced an R^2 value of approximately 0.38 on the validation set and an R^2 value of approximately 0.25 on the public test set.

Github link: https://github.com/bluebarryz/CS480-Project

1 Introduction

The 4 models used in the final ensemble model are as follows: (1) a “stacked” neural network that combines the output from an “image branch” with the output from an “ancillary trait branch” to produce a final prediction, (2) a multi-output linear regression model, (3) a multi-output k-nearest neighbours model, and (4) a multi-output random forest model. See Figure 1 for the full architecture.

2 Related Works

A similar approach to my "StackedNN" model, which has two branches (one for images, one for ancillary traits) was used in the paper "Deep learning and citizen science enable automated plant trait predictions from photographs" by Schiller et al. The difference with my approach is that I incorporate different measures to prevent overfitting, such as training additional models and combining them in a weighted average ensemble, as well as incorporating dropout layers. Furthermore, my model training incorporated more ancillary traits about the plants (163 traits in total), including traits about local climate, soil, and satellite data.

2.1 Preprocessing Method

As seen in Figure 2, the scale of the output traits differs greatly. Thus, I normalized each trait by subtracting the mean value for the trait and dividing by the standard deviation for the trait. This normalization was also done for the 163 ancillary traits. This normalization is similar to the one

Submitted to CS480/680, 2024 Spring. Do not distribute.

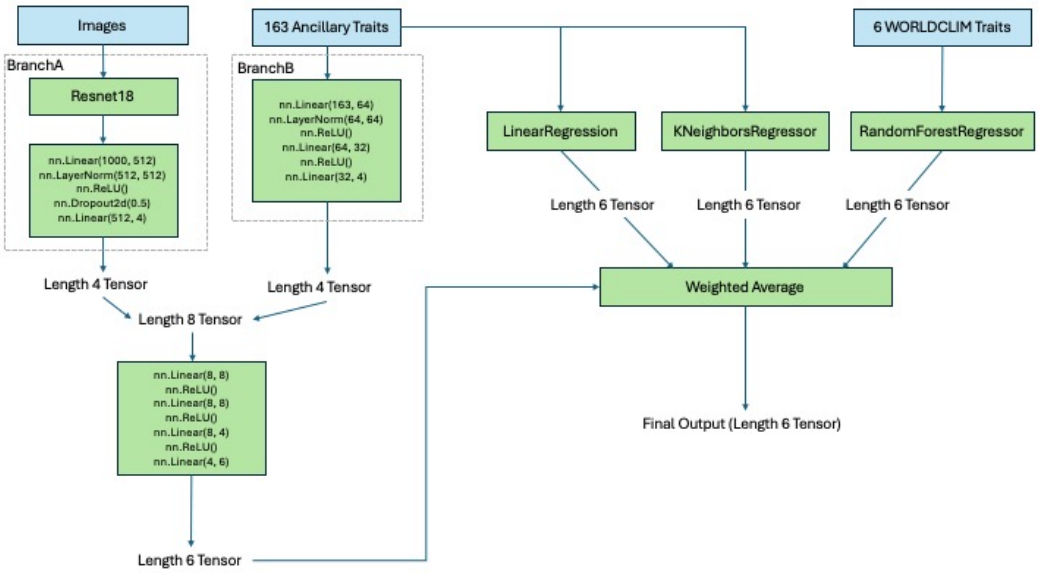

Figure 1: The architecture of the ensemble model. We see that the weighted average is computed using the 4 tensors of length 6 produced by the 4 individual models.

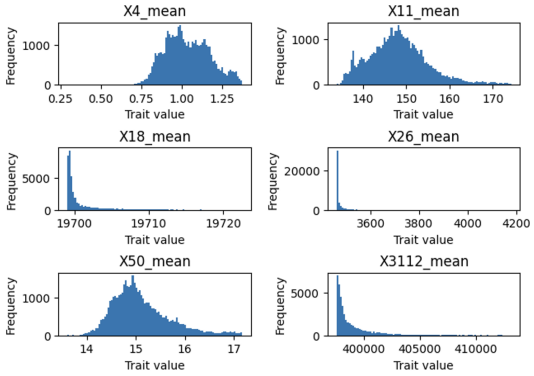

Figure 2: Histograms for the output trait values for all plants in the training dataset.

done by Schiller et al, but they subtracted the minimum trait value from the targets and divided by the range.

Some preprocessing was done on the images too. The images were upsampled from 128x128 to 256x256, then centrally cropped to be 224x224. Then they were normalized with mean = [0.485, 0.456, 0.406] and std = [0.229, 0.224, 0.225]. These steps were necessary in order to feed the images into the Resnet18 layer (which will be explained later), as this is the expected input for the pre-trained Resnet18 model I utilized.

3 Models and Main Results

Four models were trained separately and eventually combined using weighted averaging to form the ensemble model. We will explain each of these four individual models and their results below.

3.1 StackedNN

The StackedNN model uses an idea similar to stacking to produce an ensemble model (which in turn is used in the final ensemble model along with the 3 other models). The StackedNN model contains 2 separate models within it, which we will call "BranchA" and "BranchB", as shown in Figure 1. The outputs of these two models were then concatenated and fed into another layer to produce the prediction for the 6 traits.

"BranchA" takes a preprocessed image as input and feeds it into a pre-trained Resnet18 model. The motivation behind using Resnet was to leverage its capabilities in image tasks (as it was pre-trained on ImageNet) and employ transfer learning to use these outputs to help us with our regression goal. We pass the outputs of the Resnet layer into another feedforward network, which contains Linear, LayerNorm, Dropout, and ReLU layers. The Dropout layer mitigates some potential overfitting risks, and the ReLU activation function helps mitigate the potential risk of a vanishing gradient during training. The last layer of this feedforward network outputs a tensor of length 4.

"BranchB" takes the 163 ancillary traits for the plant as input and feeds it into a feedforward neural network. This contains several linear layers and again uses ReLU for non-linear activation, as motivated earlier. It also outputs a tensor of length 4.

The two length-4 tensors from "BranchA" and "BranchB" are then concatenated to form a tensor of length 8. This concatenated tensor then gets fed through another feedforward neural network, which outputs a tensor of length 6, representing the model's prediction for the 6 plant traits.

StackedNN produced an R^2 score of around 0.21 for the validation set. For all experiments, the validation set contained 20% of the 43,363 labelled plant samples.

Training for StackedNN was done using gradient descent and the Adam optimizer with a learning rate of 0.001. The mean squared error was used to compute the loss between a predicted length-6 tensor and the true trait values. Training was done independently using 3, 4, and 5 epochs, and the loss is reported in Figure 3.

Figure 3: The training loss per training sample reported every 50 batches. The final loss value was 0.517 for the 3 epochs experiment, 0.491 for the 4 epochs experiment, and 0.500 for the 5 epochs experiment.

The R^2 score for each experiment (3, 4, and 5 epochs) on the validation set is shown in Figure 4.

Since the 5 epoch experiment produced the highest R^2 score on the validation set, I chose that model to use in the final ensemble instead of the 3 epoch or 4 epoch model.

The remaining 3 models in the final ensemble provide support for StackedNN by reducing the impact of its overfitting on training data.

3.2 Linear Regression, k-nearest Neighbours, Random Forest

All of the remaining three models use multi-output regression to produce a tensor/vector of length 6 like StackedNN. The k-nearest Neighbours model predicts the length-6 vector using the output targets of the nearest neighbours of the input sample. The 163 ancillary traits are used as input in this case, so the neighbourhoods are formed based on these traits during training. The hyperparameter for the number of neighbours in a neighbourhood was set to be 10. This is a relatively high value

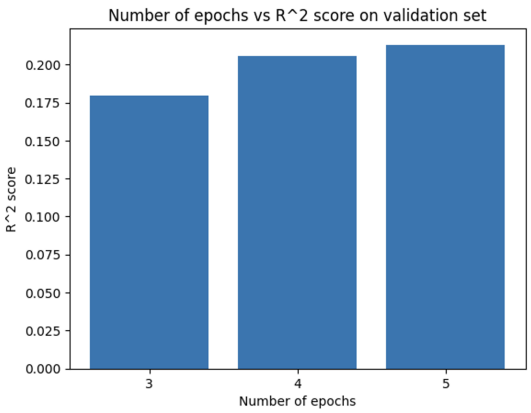

Figure 4: The R^2 scores for the 3, 4, and 5 epoch experiments were 0.180, 0.206, and 0.213, respectively.

that reduces the impact of overfitting. The Random Forest model uses multiple regression decision trees that are fit on subsamples of the training data.

Training for all three of these models was done using 5-fold cross validation.

The cross validation scores for the Linear Regression model for 5-fold cross validation were [0.16626419, 0.15951519, 0.16397977, 0.16191402, 0.16079204].

The cross validation scores for the k-nearest Neighbours model were [0.16305706, 0.15421981, 0.16143861, 0.16108168, 0.15522039].

The cross validation scores for the Random Forest model were [0.14889846, 0.14199186, 0.14081769, 0.13064541, 0.13606785].

3.3 Weighted Average Ensemble

The predictions of the StackedNN, Linear Regression, k-nearest Neighbours, and Random Forest models were multiplied by the weights 0.5, 0.18, 0.18, 0.16, in that order. These weights were chosen in order to put more emphasis on the robust StackedNN, while still allowing the other 3 models to influence the final outcome and reduce the impact of each individual model's overfitting.

This weighted average ensemble produced an R^2 score of 0.38 on the validation set, and 0.25 on the public tests. Both these scores were higher than the score obtained by just the StackedNN model itself on the validation set, which was 0.21.

4 Conclusion

Overall, I saw through my experiments that ensemble methods were quite helpful and boosted by overall performance, and these benefits reinforced my prior research (e.g. from Alhatemi and Savas' paper "A Weighted Ensemble Approach with Multiple Pre-Trained Deep Learning Models for Classification of Stroke". The benefit of ensembles can be seen in my case with the final weighted average ensemble, as using 3 additional regression models in addition to StackedNN helped boost its R^2 score.

In the future, I would like to explore methods involving attention and vision transformers for tackling this multi-output regression problem.

Acknowledgement

Special thanks to Saber Malekmohammadi for his great advice and help for this project, and to Yaoliang Yu for his great teaching and support throughout CS 480.

References

Borges, J. (2019). "The Power of Ensembles in Deep Learning". *Towards Data Science*.

Rusul Alhatemi, S. S. (2024). "A Weighted Ensemble Approach with Multiple Pre-Trained Deep Learning Models for Classification of Stroke".

Schiller, C. (2021). "Deep learning and citizen science enable automated plant trait predictions from photographs".

"What is the k-nearest neighbors (KNN) algorithm?" (N.d.) ().