Predicting 6 Vital Plant Traits using Feature Extraction and Regression

Sanjay Gupta
School of Computer Science
University of Waterloo
Waterloo, ON, N2L 3G1
s347gupta@uwaterloo.ca
report due: August 12

Abstract

Plant functional traits are important for assessing biodiversity and ecosystem processes, but measuring these traits can be cumbersome. With the advent of machine learning and computer vision, it is possible to predict these traits.The following paper describes an implementation to predict functional traits using images obtained from iNaturalist and corresponding ancillary information obtained from the TRY Plant Trait Database. The proposed model utilizes Convolutional Neural Networks (CNNs), specifically a pre-trained VGG16 model, to extract plant features from images. XGBoost is used to take these features and the ancillary information to predict plant functional traits such as leaf area (LA), growth height (GH), and others. The implementation and results are detailed in this report, with code available for repeatability on GitHub.

1 Introduction

Convolutional Neural Networks (CNNs) have been widely used in image classification tasks, and have been shown to be effective in extracting features from images. Resultingly, the implementation presented in this paper involves combining a pretained CNN (VGG16), leveraging transfer learning, to extract plant features and combines this with the ancillary information to perform regression to predict functional features.

ID	Trait
X4	Stem specific density (SSD) or wood density (stem dry mass per stem fresh volume)
X11	Leaf area per leaf dry mass (specific leaf area, SLA or 1/LMA)
X18	Plant height
X26	Seed dry mass
X50	Leaf nitrogen (N) content per leaf area
X3112	Leaf area (in case of compound leaves: leaf, undefined if petiole in- or excluded)

Figure 1: Mapping of trait names encountered within data (uploaded on Git) with their corresponding function trait name.

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

2 Related Works

In recent years, the application of Convolutional Neural Networks (CNNs) in plant trait prediction has garnered significant attention in the scientific community.

2.1 Deep learning and citizen science enable automated plant trait predictions from photographs

This study extensively explored the use of CNNs to predict plant functional traits from images. However, one of the limitations identified in the study is the exclusive reliance on image data, without incorporating additional contextual information that might improve prediction accuracy. In this case, an ensemble of CNNs were used, including models that processed only image data, as well as CNNs that also took as input world climate and plant plasticity data.

Like the ensemble model, the model presented in this paper incorporates ancillary data and utilizes CNNs for feature extraction, though the final methodology for predicting trait values is different. We also note that the normalization process used in our implementation for training images comes from Schiller's work.

2.2 VGG16 Feature Extractor with Extreme Gradient Boost Classifier for Pancreas Cancer Prediction

The proposed models (VGG16 as solely a feature extractor and utilizing XGBoost for regression) were adapted from this paper on classifying stages of pancreas cancer.

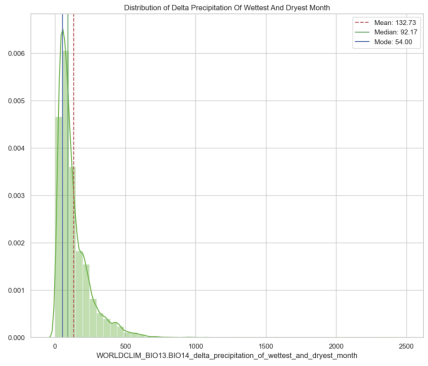
3 Main Results

3.1 Data Normalization

The role of image training-data transforms, such as resizing, flipping, and adjusting brightness, contrast, and saturation, is to augment the dataset by introducing variability, which helps the model generalize better to new, unseen data. These transformations simulate different conditions that the model might encounter, making it more robust. Normalizing color channels to the 0-1 range ensures that the pixel values are standardized, and helps in reducing the effects of lighting variations and improving convergence during training.

```python
def train_transform(image):
    image = tf.image.resize(image, (224, 224))
    # Apply other transformations (flipping, brightness, etc.)
    image = tf.[illegible]ight(image)
    image = tf.[function] image: Any [illegible]mage]
    image = tf.image.random_brightness(image, max_delta=0.1)
    image = tf.image.random_contrast(image, lower=0.9, upper=1.1)
    image = tf.image.random_saturation(image, lower=0.9, upper=1.1)
    # Normalize pixel values to the range [0, 1]
    image = tf.cast(image, tf.float32) / 255.0
    return image

def test_transform(image):
    image = tf.image.resize(image, (224, 224))
    # Normalize pixel values to the range [0, 1]
    image = tf.cast(image, tf.float32) / 255.0
    return image

# Normalize the CSV file
def normalize(df):
    # Convert all columns except ID (first) to float64 to prevent dtype issues
    df.iloc[:, 0] = df.iloc[:, 0].astype('int64')
    df.iloc[:, -6:] = df.iloc[:, -6:].astype('float64')

    # Apply log-10 transformation
    df.iloc[:, -6:] = np.log10(df.iloc[:, -6:] + 1e-4)

    # Determine outliers: rows that are more than 3 standard deviations away from the mean
    outlier_mask = (np.abs(df.iloc[:, -6:] - df.iloc[:, -6:].mean()) > (3 * df.iloc[:, -6:].std())).all(axis=1)
    # Retrieve IDs of outlier rows
    outlier_ids = df.loc[outlier_mask, df.columns[0]]
    # Remove outliers (more than 3 standard deviations from the mean)
    df = df[
        ~df[df.columns[0]].isin(outlier_ids)
    ]

    # Normalize the traits
    min_train = df.iloc[:, -6:].min()
    max_train = df.iloc[:, -6:].max()
    df.iloc[:, -6:] = (df.iloc[:, -6:] - min_train) / (max_train - min_train)
```

(a) Code snippet for applying transforms to the images to augment training dataset and improve the model's ability to generalize. We also note the normalization procedure that removes outliers and adjusts the target values



(b) Skewed distribution for a climate variable being sampled from. Highlights the need for normalizing distribution as seen in 2a

Figure 2: Data View/Normalization

3.2 Feature Extraction – VGG16

The VGG16 model, pre-trained on ImageNet, is used to leverage its extensive knowledge of recognizing various image features through transfer learning. To avoid overfitting, all layers of VGG16 are frozen, ensuring that the pre-trained weights remain unchanged while training on the smaller dataset. The model's final fully connected layer, classification, is removed to allow the extraction of a feature matrix—a high-dimensional representation of the images. This feature matrix is then combined with other dataset features and used as input for further modeling, such as with XGBoost, to predict continuous traits.

3.3 Boosting – XGBoost

For each trait, an independent XGBoost model is incrementally trained across batches and epochs, and the model's performance is tracked by monitoring the loss (RMSE) per epoch. This approach helps in fine-tuning the model parameters and selecting the best-performing models, while also providing a reliable estimate of the model's performance on unseen data.

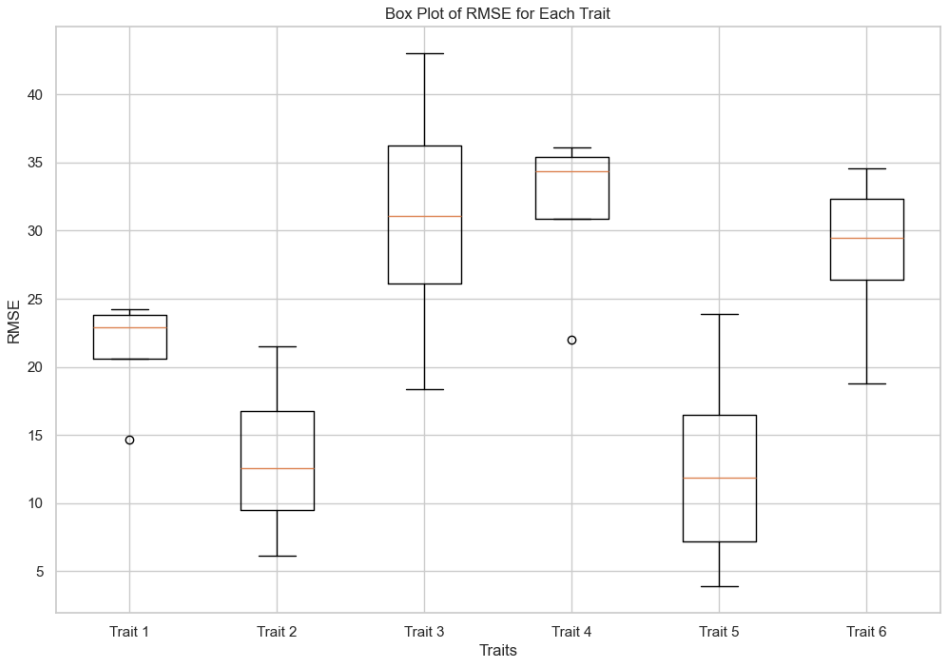


Figure 3: Box plots describing the deviation of RMSE over epochs for each predictor

4 Conclusion

Through this project, we have explored the application of transfer learning using a pre-trained VGG16 model, combined with ancillary data, to predict traits from plant images. By freezing the CNN layers, we leveraged the extensive feature extraction capabilities of VGG16, allowing us to focus on the prediction task using XGBoost models. This approach highlighted the importance of batching in managing computational resources, particularly when dealing datasets that can generate enough parameters to slow the computer to a crawl.

However, this implementation also has its limitations. The frozen CNN restricts the model from learning more task-specific features that could potentially improve performance. Future work could explore the impact of fine-tuning the CNN layers or replacing VGG16 with other models to compare their effectiveness. Additionally, there is an opportunity to devise custom CNN architectures that incorporate more domain-specific knowledge, potentially leading to better feature extraction

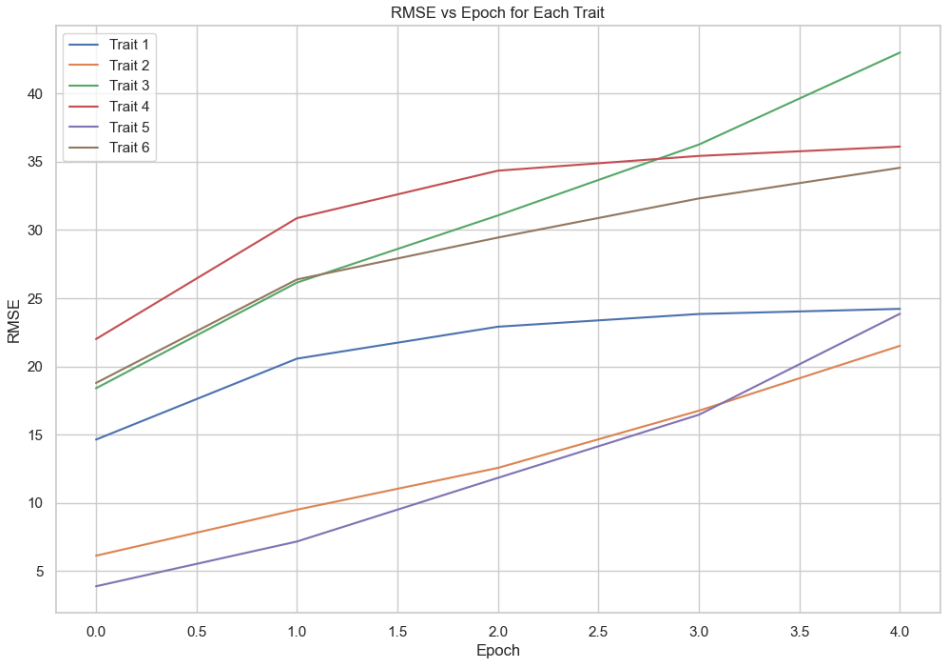


Figure 4: Plot of RMSE (per trait) against epochs for each predictor. We can see that the final model is not generalizing well and seems to over fit as the epochs increase

and improved prediction accuracy. For example, the paper from Schiller et al. meticulously describes building individual models that incorporates segments of the ancillary data features, and a mechanism to determine the corresponding weights of each model. These directions could provide valuable insights and further enhance the capabilities of the model in similar tasks – as seen from pancreatic model presented above.

In light of the scores received for this model, a point to improve upon is the testing/evaluation for XGBoost regression – methods like K-fold cross validation could have mitigated and provided a gauge on the model's ability to generalize on unseen examples. While the model did not perform to my expectations, I learned a lot for the future!

Acknowledgement

Thanks to Bakasa and Viriri (2023), Schiller et al. 2021 and Rodrigues 2020. Special thanks to Saber Malekmohammadi for clarifying many points that arose throughout this process.

References

Bakasa, W. and S. Viriri (2023). "VGG16 Feature Extractor with Extreme Gradient Boost Classifier for Pancreas Cancer Prediction." *Journal of Imaging*, vol. 9, no. 7.

Rodrigues, G. (2020). "VGG16+XGBoost, Python, SIIM-ISIC Melanoma Classification".

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, no. 16395.