Plant Traits: Leveraging pre-trained models

Sean Huh
School of Computer Science
University of Waterloo
Waterloo, ON, N2L 3G1
s5huh@uwaterloo.ca
report due: August 12

Abstract

Please note: This project makes a critical failing by using leaked test data in the pre-trained SWin model. Despite this, the effort will be to demonstrate the model's effectiveness as if the SWin model was pre-trained on solely training data.

The basis of this project's model was exploring how to learn from multi-modal data with limited compute resources. Tuning an image transformer can be resource intensive, so a pre-tuned SWin transformer was used for image feature extraction. These features were concatenated with tabular features to learn from both modalities. Other methods beyond concatenation were explored as well. The final model's hyperparameters were fine-tuned with random searching and predicted a score of $R^2 \approx 0.55$ on the public test data. See github for code.

1 Introduction

The problem is to predict 6 plant trait labels given photographs from the iNaturalist database and plant trait data. The training set has 43363 images with 164 columns of tabular data. Architecture of the model used to make predictions is given in Figure 1.

This model uses a pre-trained SWin model tuned on the PlantTraits2024 training images to extract image features. A single-layer network was used on the tabular data to extract tabular features. Both modalities' features were concatenated and processed through multiple layers to predict the 6 plant traits.

This is a multi-modal model that incorporates multiple data formats during training. In literature, many different methods exist between how to mix and learn from multi-modal data. In brief, models like ViLT as seen in Kim et al. (2021) often incorporate contextual-learning across different modalities. In ViLT, text-features help the model learn what and how to pay attention to the image features, and vice versa. Preliminary exploration on the plants dataset showed predicting with strictly image features resulted in decent performance, but a more accurate model would have to be multi-modal.

2 Related Works

The skeleton of the code was adapted from Kaggle user's HdJoJo's methodology for tuning a model on PlantsTraits2024 using just image data.

SWin's architecture as described in Liu et al. (2021) innovates on previous vision transformers by reducing self-attention heads' focus to 'windows' of the image. The model's strength is in learning characteristics of the image per-patch, and then informing other patches via cross-attention. Most relevant to the plant traits problem however is the use of multiple differently-sized windows. As

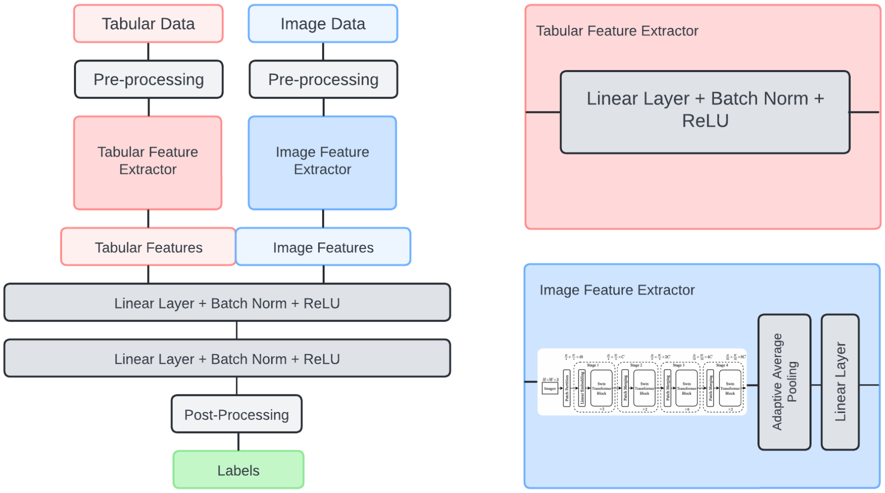

Figure 1: This multi-modal model is used to predict plant traits given tabular and image data.

outlined in the initial report by Shi et al. (2021), the image features of each plant are heavily dependent on the photographer. This necessitated classification heads for their model to estimate human variance such as distance from the plant or brightness. This paper postulates that SWin architecture partially evades this human variance due to differently-sized windows capturing plant features from different distances, brightness, etc.

To condense the output of SWin models to an image feature vector, adaptive average pooling (with stride 1) was used, following the Kaggle user's methodology. In a SWin model, each channel can be interpreted to represent some high-level 2D representation of a feature of the original image. Adaptive average pooling condenses each of these features into a single numerical value to create a one-dimensional feature vector.

3 Main Results

To pre-process the data, the test labels were augmented by log-transforming and normalizing following Shi et al. (2021). Due to the high variance of the test label values, the training data was clipped within a 0.005 and 0.985 percentile. The tabular data was also augmented by log-transforming any traits with high skewness (>1) then normalizing. This was done to handle heavily-skewed distributions and reduce scale sensitivity in preparation of training. Figure 2 demonstrates the effect both these augmentations have on the distribution of the six traits of interest and six randomly chosen features from the tabular data.

The images were augmented by applying the same augmentations used by the Kaggle user during pre-training and upscaled to (382x382) to match the SWin model's input dimensions. Applying random flips and contrasts along with image cropping was done to improve the model's generalization (recall discussion on human variance) and reduce overfitting.

The model feeds the augmented image and tabular data into feature extractors respectively. The tabular feature extractor is a simple neural network that is trained along with the rest of the model during training. The image feature extractor is a pre-trained SWin model tuned on the PlantsTraits2024 training images. The model predicted the final 6 traits directly, so the extractor removes the last prediction layer and performs adaptive average pooling (with stride 1), creating a feature vector. A final linear layer is applied to condense these features further.

Within multi-modal modelling, it is common practice to represent each modality with an equally-sized feature representation. This was emulated here but hypertuning of the tabular and image feature sizes indicated image features were weighted more in predicting the six plant traits. Because of the leaked test set, it is inconclusive to say if this is a reflection of the actual data. However, it

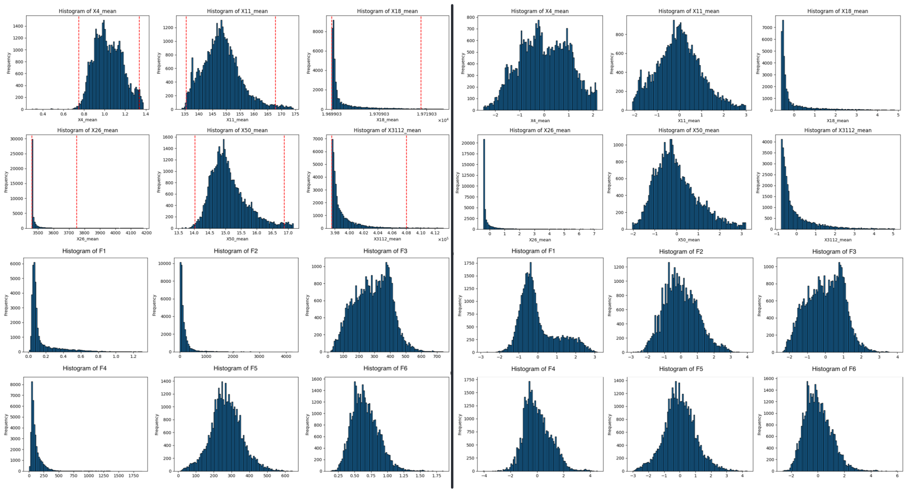

Figure 2: By augmenting the labels and features, performance improves due to normalization and reduced scale sensitivity.

does present an intriguing concept of balancing/condensing the dimensions of each modality features during hypertuning.

Regarding the pre-trained SWin model, all layers of the original model are frozen in-place to save computation resources. In abalation testing, the SWin model was used directly on the dataset with decent accuracy, see Table 1. After both the image and tabular features are extracted, they are then fused into a single layer by concatenating them. This is then brought through two more layers with batch normalization and ReLU as standard in-between.

As a trial, cross-attention mechanisms were also explored as fusion methods. Ideally, the tabular and image features would attend to one another to capture the most relevant information (per head) using the other modality as context. These contextualized features would then be concatenated and used to predict the final traits using a final layer. The model (partially depicted in Figure 3) was deployed and trained over three epochs. Ultimately, the model showed a degradation in performance versus simple concatenation as a fusion method.

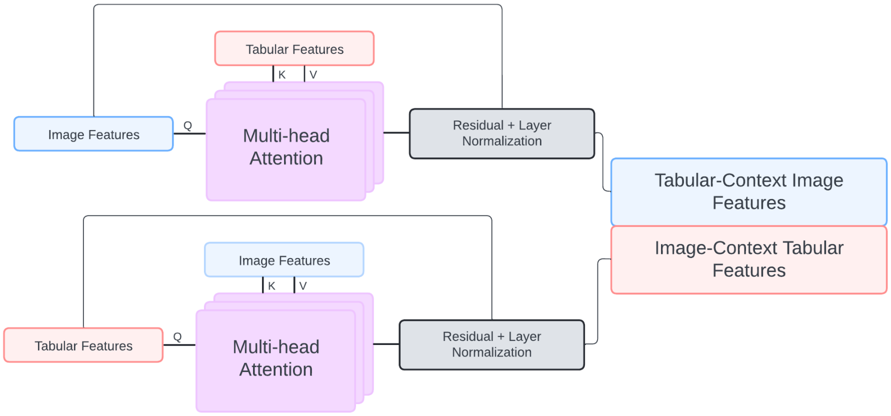
Tabular Features
K
V
Image Features
Q
Multi-head Attention
Residual + Layer Normalization
Tabular-Context Image Features
Image-Context Tabular Features
Image Features
K
V
Tabular Features
Q
Multi-head Attention
Residual + Layer Normalization

Figure 3: Alternative methods of fusion proved unsuccessful.

Table 1: Model Comparison on stand-alone SWin and Hyperparameter Tuning.
In order: [Learn Rate, Weight Decay, Image Features, Tab. Features, Combo. L1, Combo. L2].

Model	Loss average	MAE average	R^2
SWin-standalone	-	-	0.29
1e-3/1e-3/128/256/512/1024/128	0.207	0.469	0.47
1e-3/1e-3/256/64/256/2048/128	0.206	0.468	0.474
1e-2/1e-4/128/128/512/512/256	0.205	0.469	0.473
1e-2/1e-4/512/512/512/512/256	**0.203**	**0.466**	**0.481**

The model seen in Figure 1 was trained using a OneCycleLR scheduler with AdamW. Described in Smith (2017), the warm-up period during the beginning of training allowed the model to quickly converge to optimal values. The later cool-down helps prevent over-fitting to the training data. The model uses Huber loss due to concern over outliers disrupting the model's convergence. Pseudocode for how the gradient and learning rate is updated is available in Algorithm 1.

Algorithm 1: AdamW with OneCycleLR using SmoothL1Loss

```
for $\text{step} = 0, 1, 2, \ldots (\text{num\_epochs} \cdot \text{n})$ do
    $\hat{y} \leftarrow \text{model}(X)$
    $\frac{\partial l}{\partial \hat{y}} \leftarrow \begin{cases} y_i - \hat{y_i} & \text{if } |y_i - \hat{y_i}| \leq -1, \\ \text{sign}(y_i - \hat{y_i}) & \text{otherwise} \end{cases}$        // i=1,2,...,6
    $\hat{g} \leftarrow \frac{\partial l}{\partial \hat{W}} = \frac{\partial l}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \hat{W}}$        // for some arbitrary layer weights W
    $lr \leftarrow$
      $\begin{cases} lr_{min} + \frac{1}{2}(lr_{max} - lr_{min})(1 + \cos\left(\frac{\text{step}-\text{pct}\cdot\text{num\_epochs}\cdot\text{n}}{(1-\text{pct})\cdot\text{num\_epochs}\cdot\text{n}} \cdot \pi\right) & \text{if } \frac{\text{step}}{\text{num\_epochs}\cdot\text{n}} > \text{pct}, \\ lr_{max} + \frac{1}{2}(lr_{min} - lr_{max})(1 + \cos\left(\frac{\text{step}}{\text{pct}\cdot\text{num\_epochs}\cdot\text{n}} \cdot \pi\right) & \text{otherwise} \end{cases}$
    $\text{m}_{\text{step}} \leftarrow \beta_1 \cdot \text{m}_{\text{step}-1} + (1 - \beta_1) \cdot \hat{g}$
    $\text{v}_{\text{step}} \leftarrow \beta_2 \cdot \text{v}_{\text{step}-1} + (1 - \beta_2) \cdot \hat{g}^2$
    $\text{m}_{\hat{\text{step}}} \leftarrow \text{m}_{\text{step}}/(1 - \beta_1)$
    $\text{v}_{\hat{\text{step}}} \leftarrow \text{v}_{\text{step}}/(1 - \beta_2)$
    $W \leftarrow W - lr \cdot \text{m}_{\hat{\text{step}}}/\left(\sqrt{\text{v}_{\hat{\text{step}}}} + \epsilon\right) - lr \cdot \text{weight\_decay} \cdot W$
```

Several hyperparameters were tuned by performing three epochs of training using different configurations via Random Search, then testing against the test set. This was chosen over a Box Method due to limited compute resources. Results for the standalone Swin-model along with a sample of 5 hyperparameter configurations are summarized in Table 1.

The final training occurred over 6 epochs of training and dimensions using hyperparameters [1e-4, 1e-2, 512, 512, 512, 512, 256] that performed best while tuning. Running this model three times resulted in an R^2 standard deviation of 0.002256.

4 Conclusion

This model uses the idea of multi-part training for multi-modal models. By using a SWin model pre-tuned to the image dataset, the surrounding model was relatively simple and efficient. Despite the pre-tuned model using leaked test data, the model should still work with at least some effectiveness had the model been tuned on strictly training data. Actual testing would need to be done to confirm this, however.

Beyond this, there is interest in exploring different fusion mechanisms between modalities. Particularly, it would be intriguing to explore problems where a multi-modal model best performs with severely different dimension sizes for different modalities. There are multiple ways these features can be combined and may be problem-specific. Exploration would have to be done to see how fusion methods translate for different problems.

Acknowledgement

Thank you to TA Saber Malekmohammadi for helping me in the creation of this report. I made a mistake in using a model that was pre-trained on leaked test data. After I explained the mistake, he was supportive in understanding what had occurred and allowed me to still submit my model.

Thank you to professor Yao-Liang Yu for being foundational to my undertanding of many of the topics and techniques used for this project.

Figure 1 adopts from Liu et al. (2021).

References

Kim, W., B. Son, and I. Kim (2021). "ViLT: Vision-and-Language Transformer Without Convolution or Region Supervision". arXiv: 2102.03334 [stat.ML].

Liu, Z., Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo (2021). "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows". arXiv: 2103.14030 [cs.CV].

Shi, L., Q. Wei, L. Xie, W. Wang, and Y. Zheng (2021). "Growing deep learning-based image segmentation for bone age assessment: A systematic review and future perspectives". *Scientific Reports*, vol. 11, no. 1, p. 14553.

Smith, L. N. (2017). "Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates". *arXiv preprint arXiv:1708.07120.*