CS480 Kaggle Competition

Mitchell Dolny
School of Mathematics
Data Science Major
University of Waterloo
Waterloo, ON, N2L 3G1
mdolny@uwaterloo.ca
report due: August 13 8AM EST

Abstract

The task is to predict plant properties - so called plant traits - from citizen science plant photographs, and their corresponding ancillary information (Yu, 2024). The Best R2 score I was able to achieve was 0.37664 on the kaggle submission, before the final test results were released. I used a dino transformer and a basic CNN for tabular data. Click Here for Github repo

1 Introduction

This project aimed to predict the following plant traits:

- **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)

The goal of this project was to maximize our R2 Value which is calculated by:

$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}}$ where SS_{RES} is the sum of squared of our residuals and SS_{TOT} is the sum of the values compared to the mean. To maximize our R^2 we need to ensure our predictions are as close as possible to the actual y value.

2 Related Works

I had two main approaches for this project. The first approach was to get an understanding of building my own mode;. Firstly, I used the tabular data and a simple CNN to make predictions of the output. The second approach was to use Transformers and a CNN to build a combined model that received text and images, and make a prediction. The first approach was rather basic, but there is a lot of research into the field of transformers for my second approach.

I Pre-Processed my data similarly as done in a report by Schiller, C. et al (2011), where the tabular data is reduced by log10. This allows the data to be less skewed, then using this transformed data the outliers were removed if any of the 6 target columns were +/- 3 std. away from the mean. Then the targets are normalized by using the the following formula $f(x) = \frac{x - min_{train}}{max_{train} - min_{train}}$ allowing

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

us to have all values between 0 and 1. This allows for better convergence, consistent learning rates, and more (Schiller et al., 2011).

A study conducted by Dosovitskiy, A. et al. (2021), where they experimented with applying a standard transformer directly to images, with the fewest possible modifications. They split the image into patches that are the fed into the transformer. Similar to what I did in my final implementation of my model (Dosovitskiy et al., 2021).

A study showed that the transformer DINO_V2 self-supervised features outperformed the current state of the art by a large margin (Oquab et al., 2024). Because of this out-performance I chose to use this as my transformer of choice.

3 Model Choice

I used two approaches for this project, where the first approach was building a simple linear layer for the tabular data. Then the second approach was to build a combined model where I used a transformer for the images and a simple CNN for the tabular data. I would then combine the outputs of the 2 models and create our respected 6 estimates we are looking to predict.

I used many approaches for creating CNN's where I first started off with a linear decrease, and then ramping up the complexity by increasing the output size of layers and then decreasing. I added BatchNorm's, Dropout's, and ReLU's to reduce overfitting. From this I was able to reach a value of $R^2 = 0.158$ with just tabular data.

Once my tabular model was done, I transitioned to creating a combined model where I would use a transformer or a CNN for the images, my (or very equivalent) CNN for tabular data and then a simple network to combine the two predictions together from both models. I tried pretained CNNs including Resnet 50 and 150, and also transformers including SWIN-S, SWIN-T and VIT-b-16. Using these I could not get above a R2 value of 0.18 on validation data and a 0.155 on the test set when submitted to Kaggle.

The transformer that gave me the most success was the dinov2_vitb14 transformer in conjunction with a simple CNN for the tabular data. As stated earlier, Dino_V2 was out performing in the supervised-learning features, and since this project is supervised-learning this gave me the best R2 score out of everything else I tried.

4 Experimental Details

Tabular Model

As this was my first ML project I decided to start simple, and build a simple CNN network with a couple of linear layers. I was able to get it training and get an output. From there I started learning about what schedulers to use and what optimizers to use that would be best suited for regression tasks. The optimizer I went with was MSELoss due to this being a regression problem. I also went with AdamW as my optimizer with a learning rate of 1e-3. I chose AdamW over Adam as AdamW gave me consistently better results. In my best attempt I did not use a Scheduler as I found I got consistently worse results.

Combined Model

From there I created a combined model where I was using a pretrained vit-b-16 with tabular data and I was getting a R2 score of 0.15, which was not good enough, and I knew it was due to my image model. Utilizing Dino here drastically increased my performance and gave me a submission of R2 valued at 0.37764.

Over-fitting and Under-fitting

This was a huge problem for this project as I could never get this just right, either the model was over-fitting the results or under-fitting. If the model was over-fitting I would make the model less complex, adding dropouts, add transformations, weight decay, and more. If it was under-fitting I would make the model less complex, fine tune the optimizer, and decrease the transformations.

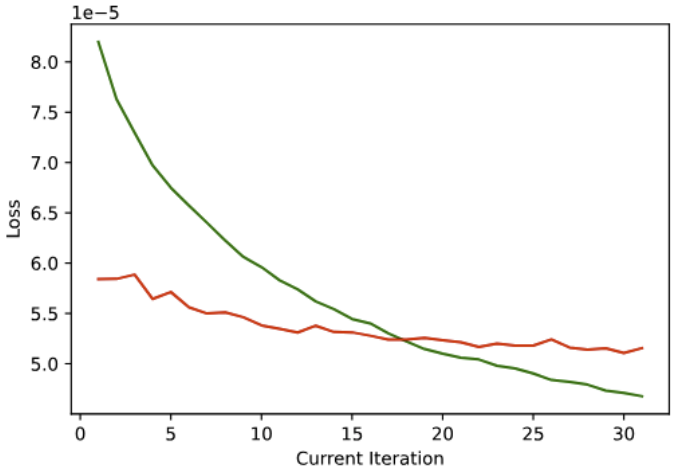

Figure 1: The training loss of an attempt vs the validation loss, where in this attempt the model is under-fitting and needs to be run longer or it needs to be fine-tuned.

One of the main ways I tested overfitting or underfitting was through graphing val loss vs train loss. As seen in Figure 1, the model is currently under-fitting due to the red line (val_loss) being underneath the green line (train_loss). This shows that either I should run for more epochs which is very computationally expensive due to my limitied resources, or I need to make the model more complex.

Generating the Best Submission for a model

For this I used checkpoints where I would store the R2 Score, Model, and Epoch number which gave me the best score. On completion of training I would load the best model given the R2 score and use that to generate my predictions.

5 Building my Final Model

How This Was Run Due to limitations I had locally for computational power I had to seek external places to run my code. I used lambda labs A6000 to train my models using Dino.

Pre-Processing Data

The best method that I found was to not remove outliers. I believe removing outliers removed too many data points from the model 3k and this caused the model to perform worse. Furthermore, I also found that adding transformations besides Resize(224), and Normalize caused it to also perform worse. So my final pre-processing include the log normalizing the targets and converting them into a 0,1 scale, applying basic transformations to pictures, and also standard scaling my tabular data.

Model Selection

I decided to go with a dinov2_vitb14 pretrained transformer and a simple yet effective CNN for the tabular data. The Dino transformer would produce an output of size 64 with the tabular model producing an output of 64. Combining these together I would reduce this to the 6 predicted traits we need.

Training the Model

To train the model I only used a criterion and an optimizer. My criterion was MSELoss and my optimizer was AdamW(1e-3). I tried many different schedulers and learning rates, but this performed the best.

Comparing R2 Scores

To check the trend of each model I trained I would generate an R2 plot to see the trend of R2 scores in each epoch. This would allow me to see if it was bottlnecking and try to improve the model as seen in Figure 2.

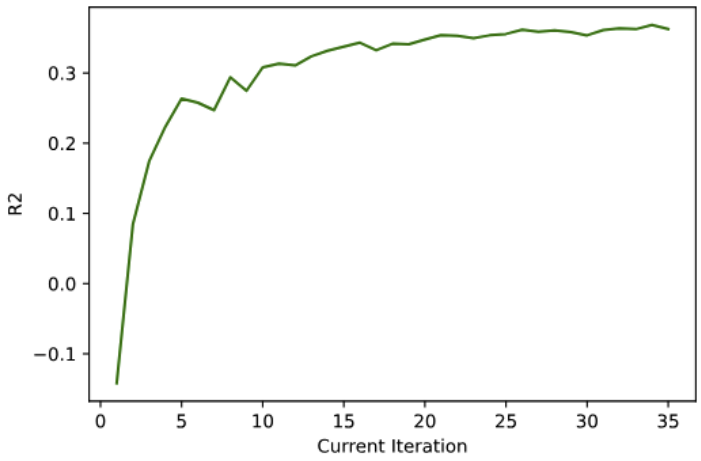

Figure 2: The output at each epoch of the R2 score, showing drastic increases of performance in the beginning and as it reaches epoch 6 it starts ti bottom off

Generating Predictions

To create the predictions I applied the same transformations to the test images and tabular data, by using standard scalar again on the tabular data. Once the predictions were generated I have to denormalize them by the following formula: f(x) = $10^{(x \cdot (\text{max\_train} - \text{min\_train}) + \text{min\_train})}$. Then the data is formatted for a csv file and outputted for submission.

6 Conclusion

I learned that creating and finding the right model for a task is very difficult, with the amount of fine tuning you must do, to see if you even have the right model. I also learned how important GPUs are, as I never really understood how powerful they really are. They sure made me realize that NVIDIA is good at what they do.

Limitations I faced during this project was computational power, specifically training models on images. Since I did not have access to a decent GPU that would run (google colab didnt work) I was using kaggle notebooks which often failed or took a very long time to run.

Next time I would dive earlier into research papers or existing kaggle projects right at the beginning before trying to "produce a submission." This wasn't a good way to start and it definitely made me take a lot longer to make my project and report.

This was a great learning experience, and I am looking forward to doing more ML Projects in the future without a leaderboard staring directly at me!

References

Dosovitskiy, A. et al. (2021). “AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE”.

Oquab, M. et al. (2024). “DINOv2: Learning Robust Visual Features without Supervision”. arXiv: 2304.07193 [cs.CV].

Schiller, C., S. Schmidtlein, C. Boonman, A. Moreno-Martínez, and T. Kattenborn (2011). “Deep learning and citizen science enable automated plant trait predictions from photographs”. *Sci Rep*, vol. 11, no. 16395, p. 16395.

Yu, Y. (2024). “Predicting 6 Vital Plant Traits”.