Beyond Visible Spectrum: Developing Computer Vision Models for *Agricultural Hyperspectral Image Classification*

Todd Perkins, Saurav Aryal, PHD
AI4PC, Institute for Human-Centered Artificial Intelligence

INTRODUCTION

Crop diseases caused by fungal, bacterial, and other pathogens reduce agricultural productivity and threaten food security. Our task is a three-class classification that consists of Health, Rust, and Other. Through it, we distinguish healthy crops from rust-infected and other diseased or damaged plants. Remote sensing with RGB, multispectral, and hyperspectral imagery can support diagnosis early on, but the many spectral bands and complex relationships in these datasets make automated detection difficult. Through the Beyond Visible Spectrum competition, we aim to improve automated crop disease classification to support more precise and sustainable farming practices.

METHODS

1. **Multimodal feature extraction**
   Features were extracted from RGB, multispectral, and hyperspectral imagery. For RGB, we used GLCM texture, color indices (ExG, ExR, VARI), and HSV stats. For our multispectral bands, we used vegetation indices (NDVI, NDWI, MCARI, OSAVI, RENDVI). For our hyperspectral bands, we used spectral summaries (red-edge slope, entropy, band ratios). Cross-modal features combined RGB and MS signals to capture stress and disease cues.

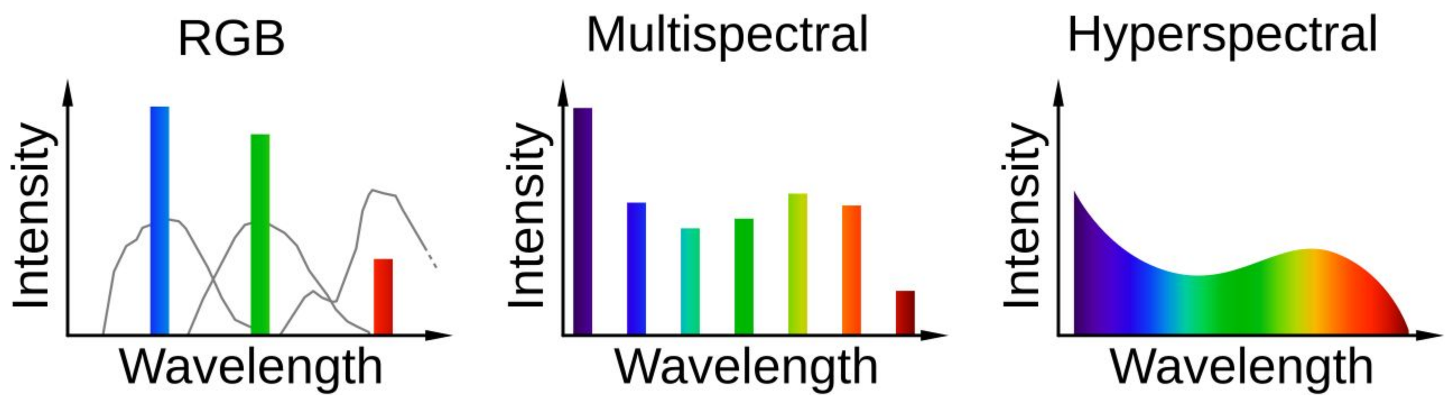

***Figure 1:** Spectral sampling for RGB, multispectral, and hyperspectral imaging.*

2. **Dimensionality reduction**
   Principal Component Analysis was applied to the hyperspectral spectral curves to reduce dimensionality and multicollinearity. RFECV (Recursive Feature Elimination with Cross-Validation) was an optional method for feature selection but was disabled in the final setup.

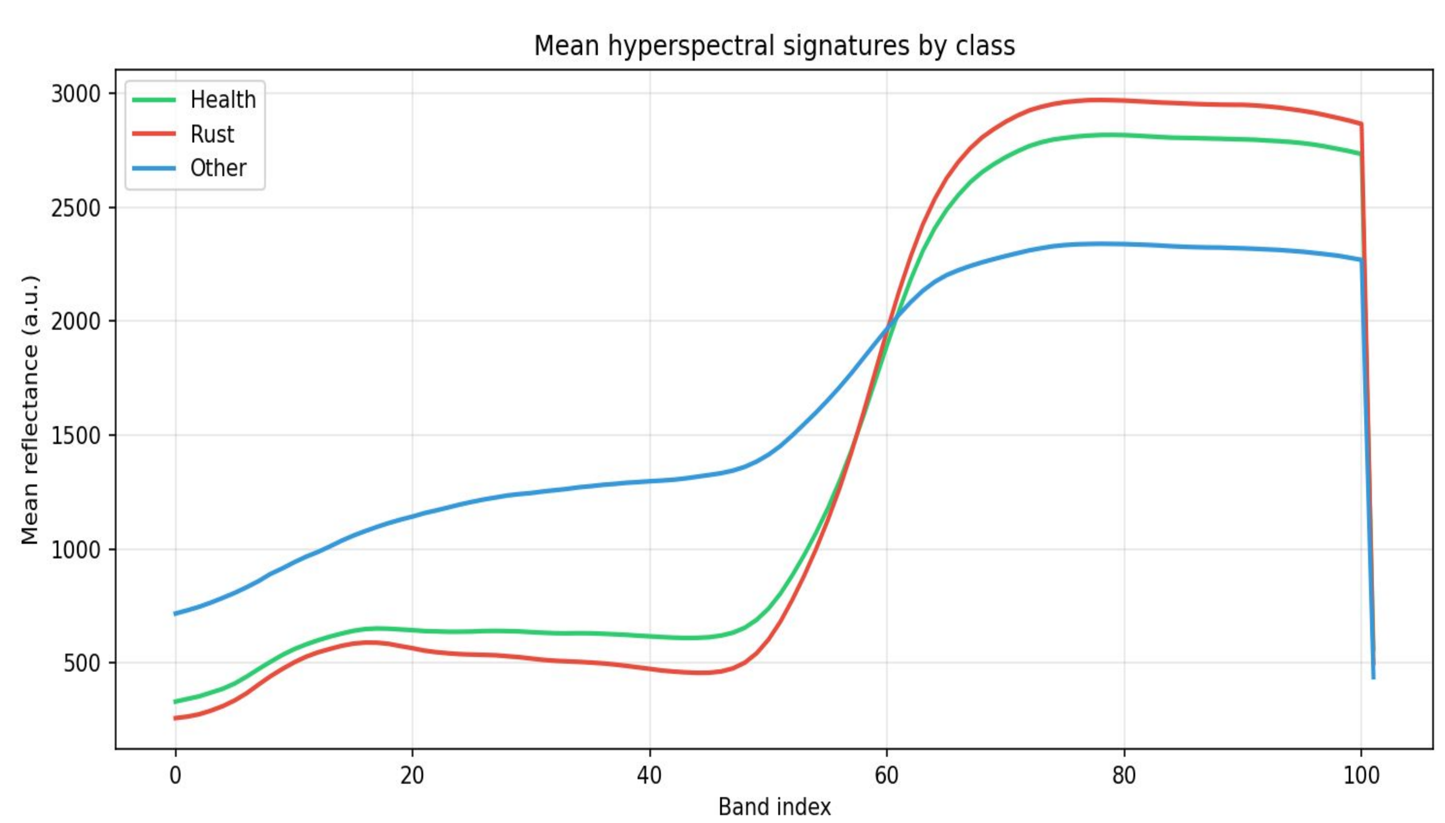

***Figure 2:** Mean hyperspectral reflectance by class*

METHODS

3. **FLAML AutoML model selection**
   Used FLAML with a time budget to search over LightGBM, XGBoost, CatBoost, Random Forest, Extra Trees, HistGradientBoosting, and logistic regression. Selected the best model and hyperparameters by maximizing macro-F1 with a 5-fold stratified CV and Ray for parallel trials.
4. **Class imbalance and post-processing**
   Applied class-balanced sample weights during training and tuned per-class decision thresholds on out-of-fold probabilities (utilizing a grid from 0.20 to 0.80) in order to improve macro-F1 for the imbalanced Health/Rust/Other classes.

Initial Results

In our initial phase, we utilized Optuna-tuned models (HistGradientBoosting, RandomForest, ExtraTrees, Stacking, etc.) on CV accuracy and macro-F1. We then received the result per FLAML model family (lrl2, rf, histgb, lgbm, extra_tree) and shows how FLAML explored these options.

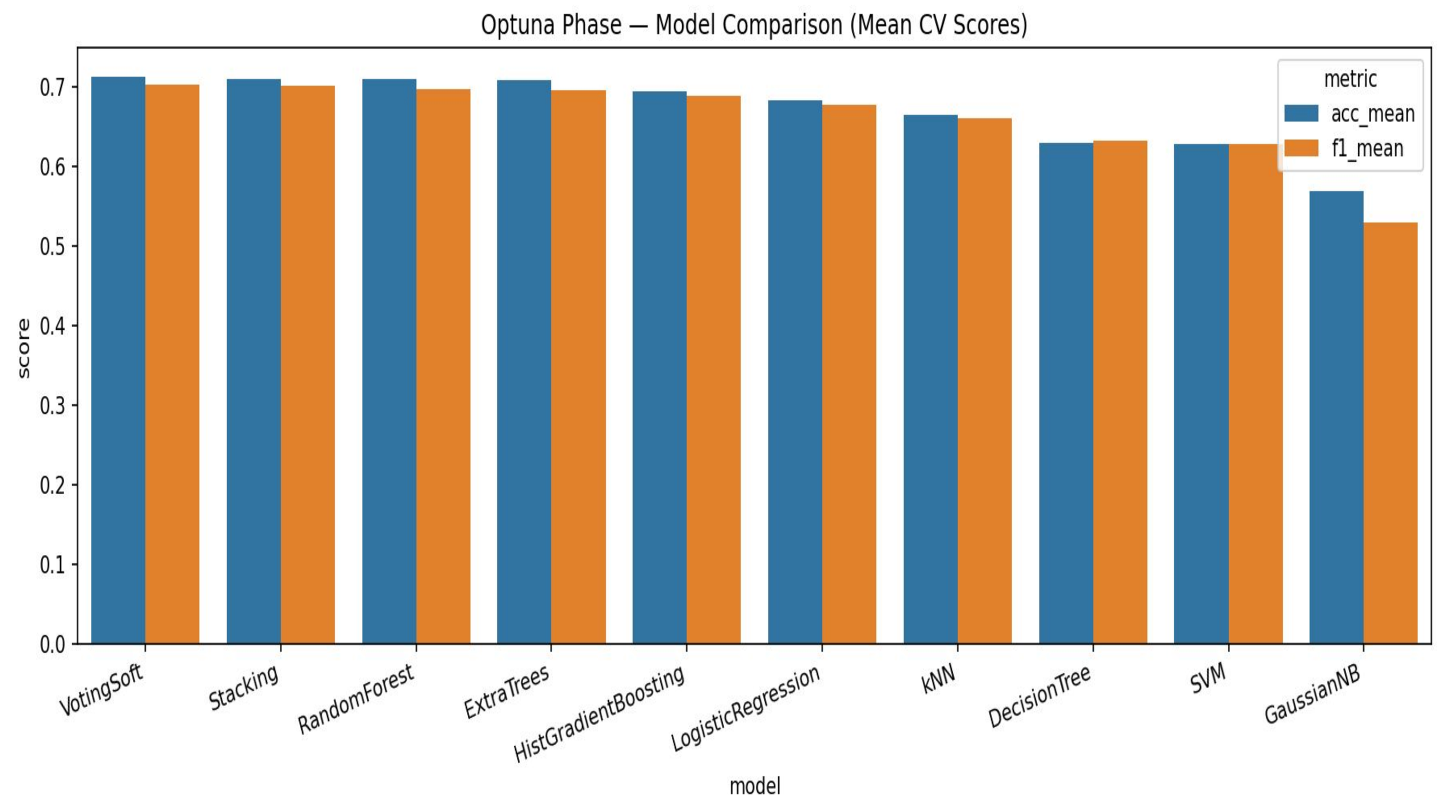

***Figure 3**: CV scores for Optuna-tuned model families and ensembles.*

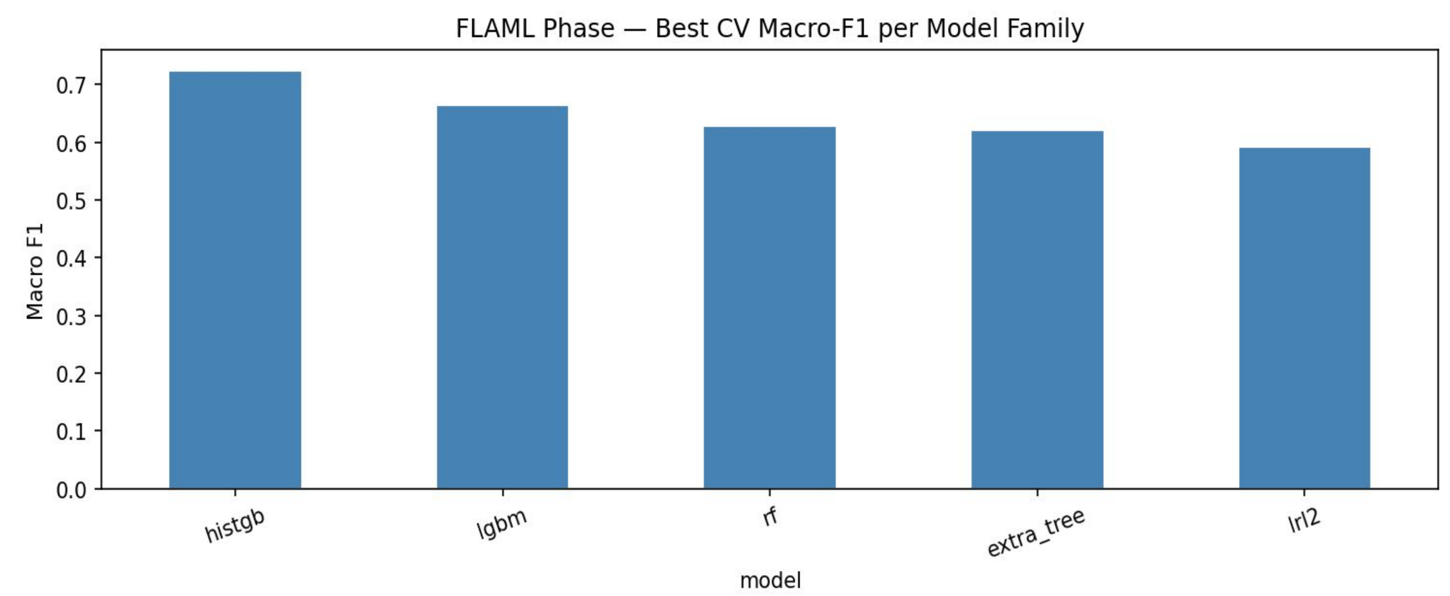

***Figure 4:** Best CV macro-F1 per FLAML model family*

Final Results

We chose to only utilize FLAML because it exceeded Optuna's performance with a simpler pipeline and let the full time budget be used for one search instead of splitting it with Optuna and ensembles. The learning curve shows macro-F1 improving over time as FLAML explored different model families. FLAML selected HistGradientBoosting as the best model and reached a best CV macro-F1 of 0.74. Per-class threshold tuning on out-of-fold predictions further improved performance for the imbalanced Health, Rust, and Other classes.

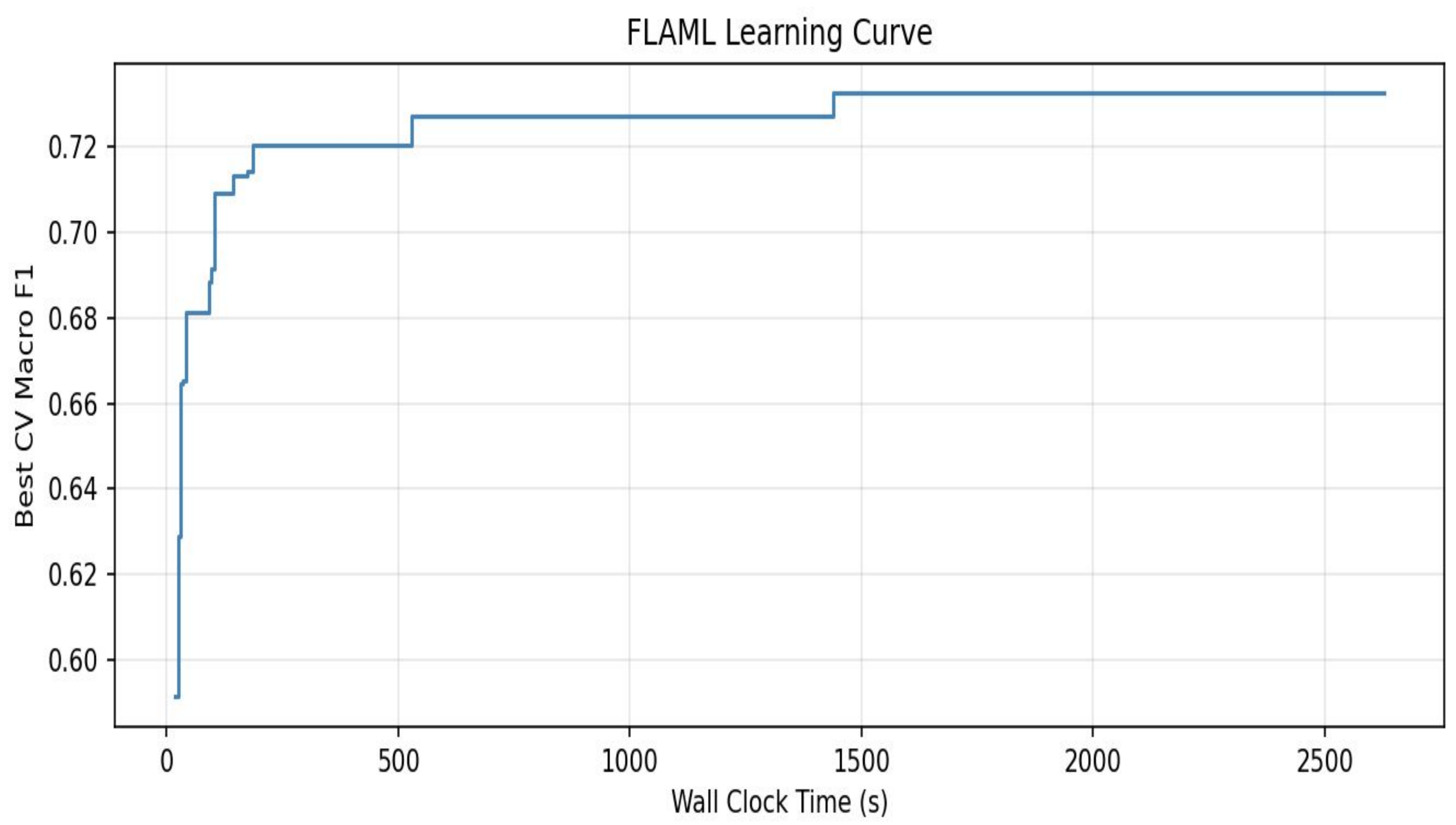

***Figure 5:** Best CV macro-F1 over FLAML search time*

REFERENCES

1. Microsoft. (n.d.). *FLAML: A Fast Library for Automated Machine Learning & Tuning*. https://github.com/microsoft/FLAML
   Medium. (n.d.). How to perform classification in Python with FLAML. https://medium.com/codex/how-to-perform-classification-in-python-with-flaml-f2cf7f68ef92
2. Medium. (n.d.). A conceptual explanation of Bayesian model-based hyperparameter optimization for machine learning. https://medium.com/data-science/a-conceptual-explanation-of-bayesian-model-based-hyperparameter-optimization-for-machine-learning-b8172278050f
3. Ozechi, S. (2021, August 30). Working with image pixels in OpenCV. *Towards Data Science*. https://towardsdatascience.com/working-with-image-pixels-in-opencv-fb03b1fe9439/
4. Alouini, Y. (n.d.). Working with TIFF files. *Kaggle*. https://www.kaggle.com/code/yassinealouini/working-with-tiff-files
5. Pedregosa, F., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.
   HistGradientBoostingClassifier: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.HistGradientBoostingClassifier.html
6. Pedregosa, F., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.
   ExtraTreesClassifier: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html

ACKNOWLEDGEMENTS

This research project was supported in part by the Office of Naval Research grant N00014-22-1-2714. The work is solely the responsibility of the authors and does not necessarily represent the official view of the Office of Naval Research.