# **RESEARCH PROPOSAL**

## **PLANT LEAF DISEASE DETECTION PROBLEM USING XGBOOST**

|  |
| --- |
| **Group name:**  DoubleSlash  **List of members:**  20120165 - Hồng Nhất Phương  19120522 - Phạm Quốc Hưng |

|  |
| --- |
| **Keywords:** Parallel Programming, CUDA, Plant Disease Detection, Machine Learning, Computer Vision, Ensemble Learning, Gradient Boosting  **List of references:**  [A Gentle Introduction to XGBoost for Applied Machine Learning](https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/)  [XGBoost for Classification](https://youtu.be/8b1JEDvenQU?si=kdWBRs1GRe8ZSbnb)  [Plant leaf disease datasets](https://www.kaggle.com/search?q=plant+leaf+disease+datasets)  [Computer vision explanation](https://cvexplained.wordpress.com/)  [Introduction to Numba: CUDA Programming (nyu-cds.github.io)](https://nyu-cds.github.io/python-numba/05-cuda/)  [Random Realizations – XGBoost from Scratch](https://randomrealizations.com/posts/xgboost-from-scratch/) |

|  |
| --- |
| **Content**   1. **Summary:**   Parallelizing the machine learning model is one way to speed up the training process, save time and be able to deploy the model on even larger data sets. For that reason, our team decided to parallelize a machine learning model, specifically the XGBoost model used for plant leaf disease detection problem. The team's plan for parallelization is to rewrite the XGBoost algorithm without calling its library, then parallelize the parallelizable code and make optimizations to increase performance. To complete this project, our team used numpy, numba library and google colab to run the model.   1. **Background**   Plant leaf disease detection using XGBoost predicts diseases from plant leaf images using the XGBoost algorithm and leverages parallel algorithms to speed up the process. The process involves in the following stages:  - Data collection: collect images of pre-labeled leaves, including both healthy and diseased samples  - Feature extraction: extract relevant features from image data to create representative features for each class. Use OpenCV image processing module.  - Install XGBoost model sequentially: install XGBoost model ourselves without using existing libraries  - Parallelize the XGBoost model and perform optimization: consider parts that take a long time and can be parallelized to improve model performance  - Train the model and compare results: train and predict results on two sequential and parallel versions, then compare the time and accuracy of the two versions  Time and performance are aspects that are benefited by using parallelization. Parallelization helps reduce training time, making it possible to use very large datasets or scale up the model's complexity   1. **The challenge**  * Regarding feature extraction, members have to spend time learning how to extract good features, making classification easier. * Regarding output, the team's desire is to create a multiclass classification model. This is a challenge because the model will be complex and our team don’t have much experience in dealing with multiclass problem * Regarding the installation of the XGBoost algorithm, members have to spend time learning in-depth about how to install the algorithm because of the lack of experience and just knowing the general implementing idea   After this project, our team hopes to know more about image processing, feature extraction for problems that use images as input as well as a deeper understanding of how XGBoost is built and deployed.   1. **Resources**  * Our team uses Google Colab as the environment to run the model since it already has CUDA installed in advance and provides a powerful GPU for parallel programming with no cost. * List of resources we uses as references to implement the XGBoost model:   [Random Realizations – XGBoost from Scratch](https://randomrealizations.com/posts/xgboost-from-scratch/)  <https://github.com/Ekeany/XGBoost-From-Scratch>  [Accelerating XGBoost using GPU](https://peerj.com/articles/cs-127.pdf)   * About data: data was searched by keyword "plant leaf disease detection", looking at popular datasets as well as other data sets used and uploaded by users in kaggle  1. **Goals And Deliverables**   **Partly achieved:**   * Successfully installed and run XGBoost sequential version as well as perfected the idea of ​​parallelization and started implementing the idea.   **Plan to achieve:**   * Being able to parallelize and have some optimizations for XGBoost implementation with a shorter execution time in comparison to the original sequential version while the model still maintains an acceptable accuracy * Create XGBoost model with binary classification and use binary classification algorithms for multi-classification problems   **Hope to achieve:**   * Applicable on larger and more complex datasets * Apply many other optimization techniques to reduce execution time * Build a multiclass classification model to classify the specified number of labels without relying on binary classification   **Demo plan:**   * Show some evaluation metrics of the model like accuracy or precision and execution time along with a few charts to visualize the differences in performance between non-parallel and parallel version |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Weekly schedule:**   |  |  | | --- | --- | | Week | Task | | Week 01 - 02  (26/02/2024 - 10/03/2024) | Find a suitable topic | | | Week 03  (11/03/2024 - 17/03/2024) | Data collection | | | Week 04  (18/03/2024 - 24/03/2024) | Feature extraction | | Week 05  (25/03/2024 - 31/03/2024) | Feature extraction + Handle imbalance data | | Week 06 - 08  (01/04/2024 - 21/04/2024) | Install the sequential XGBoost algorithm | | Week 09 - 10  (22/04/2024 - 05/05/2024) | Install the parallel XGBoost algorithm | | | Week 11 - 12  (06/05/2024 - 19/05/2024) | Applying optimized techniques to the parallel XGBoost algorithm | | |