Chair of Data Science in Earth Observation

School of Engineering and Design

Technical University of Munich

Data Science in Earth Observation Project Report

Case Study of Leveraging Optical Remote Sensing Data and ML Models for Tree Species Classification

Jiaqi Li (03796565)

Canberk Yalçıklı (03776764)

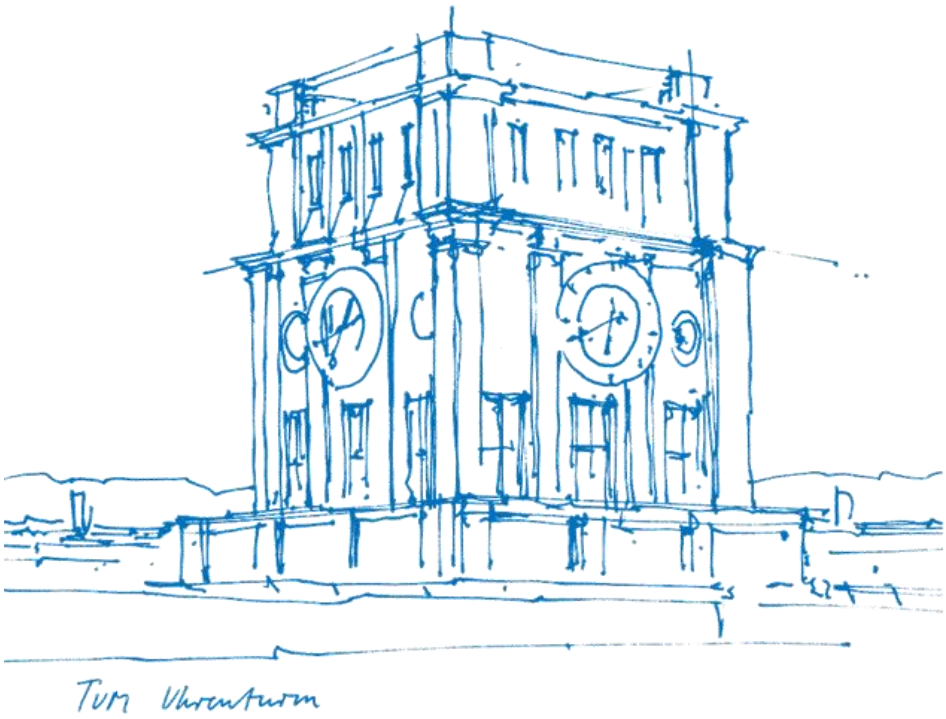
Supervised by

Prof. Dr.-Ing. habil. Xiaoxiang Zhu

Prof. Dr.-Ing. Muhammad Shahzad

Mr. Yang Mu

January 31st, 2025



Abstract

Accurate tree species classification is essential for precision forestry and biodiversity conservation. To address challenges such as cloud obstruction, limited spatiotemporal resolution, and data imbalance in remote sensing imagery, this study proposes a **hierarchical (leaf type–genus–species)** classification framework based on multi-temporal Sentinel-2 data. We first apply cloud masking and temporal interpolation to the raw imagery, compute multiple vegetation indices (NDVI, EVI, SAVI, etc.), and subsequently employ various machine learning and deep learning models, including XGBoost, Convolutional Neural Networks (CNN), and Vision Transformer (ViT), for tiered training and prediction.

Experimental results on three taxonomic levels demonstrate that ViT achieves the best accuracy at the species level (e.g., 65%, surpassing GNN and XGBoost by approximately 3% and 1%,), highlighting its advantage in capturing subtle spectral and temporal features. Moreover, leveraging time-interpolated multi-temporal data effectively mitigates the impact of cloud cover and enhances model robustness. The hierarchical label structure further improves generalization in large-scale, fine-grained species identification tasks. These findings offer valuable insights for forest resource assessment, ecosystem monitoring, and precision forestry applications.

Keywords: Tree Species Classification; Multi-temporal Remote Sensing; Sentinel-2; Hierarchical Learning; Deep Learning; Vision Transformer; XGboost; Convolutional Neural Network; Random Forest; Graph Neural Network

Contents

1. Introduction

Tree species classification using remote sensing data plays a crucial role in biodiversity monitoring and forest ecosystem management [1]. The accurate identification and mapping of tree species distribution is fundamental for multiple stakeholders, including forest managers and governmental bodies, to implement effective reforestation strategies, assess forest resilience against various environmental stressors, and guide ecosystem adaptation to climate change. This detailed species-level information enables better decision-making in sustainable forest management, particularly in determining site-specific species selections and identifying vulnerable areas requiring targeted interventions.

Remote sensing technologies, particularly the Sentinel-2 satellite platform, have revolutionized our ability to monitor and classify tree species at large spatial scales. The platform's high-frequency multispectral imagery provides rich data for vegetation analysis, though significant challenges remain in the field. These challenges include the complexity of converting pixel-level information to meaningful forest features, the difficulty in acquiring reliable reference data for model training, and the computational demands of processing large-scale satellite imagery effectively.

In our research, we developed a hierarchical classification approach using multi-temporal Sentinel-2 data to identify tree species across three taxonomic levels: leaf types (broadleaf/needleleaf), genus, and species. Our methodology incorporates comprehensive data processing steps, including cloud masking, seasonal composite creation, and the calculation of various vegetation indices (NDVI, EVI, SAVI). By utilizing multiple spectral bands (B2-B12) and temporal snapshots, our approach captures both the spatial and seasonal characteristics of forest vegetation. Different deep learning methods were used to classify the data. The selected methods include: XGBoost, Vision Transformer (ViT), Convolutional Neural Network (CNN), Random Forest(RF), Graph Neural Network(GNN). Fine-tuning was performed for these five methods.

2. Data Curation

a. Data Acquisition

- **Data extraction form Google Earth Engine:**

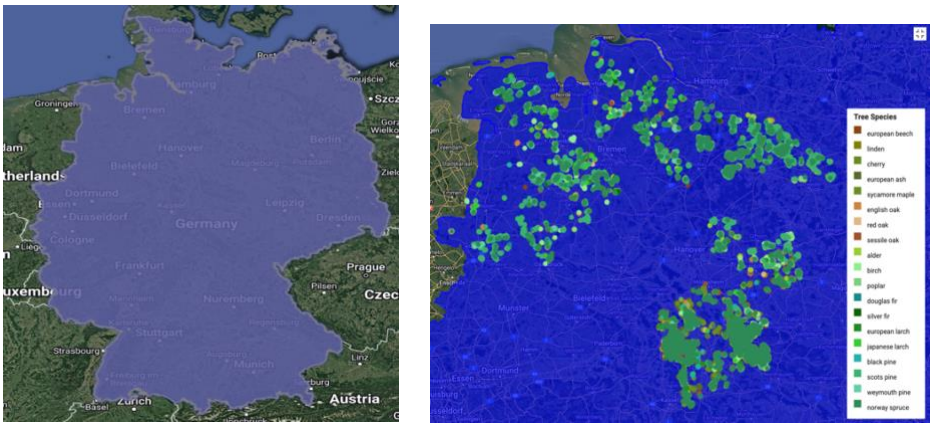


Figure 1: Region of Interest – Germany (left); Species distribution inside of Germany (right)

We obtained data and preprocessing workflow through the Google Earth Engine (GEE) platform to extract multi-temporal Sentinel-2 data features of tree species within Germany. First, we defined the study area and imported tree species distribution data. It can be shown in figure 1. Then performed cloud masking (threshold 20%) and monthly compositing on Sentinel-2 Level-2A imagery. We calculated multiple vegetation indices including NDVI and EVI, and acquired local features through 2×2 pixel neighborhood sampling. All data is organized according to a "leaf type-genus-species" hierarchical structure, ultimately creating a patch-level dataset containing multi-temporal, multi-band, and vegetation indices, providing a foundation for subsequent machine learning model training.

- **Getting to know the initial dataset.**

The dataset extracted from GEE is merged into single dataset. The extracted data has the shape 37907 x 125, which composes of sentinel-2 spectral bands, ratio indexes, labels and geometry coordinates.

The spectral bands are ranging from B1 to B12 and covering several spatial resolutions as it can be seen in figure 1.

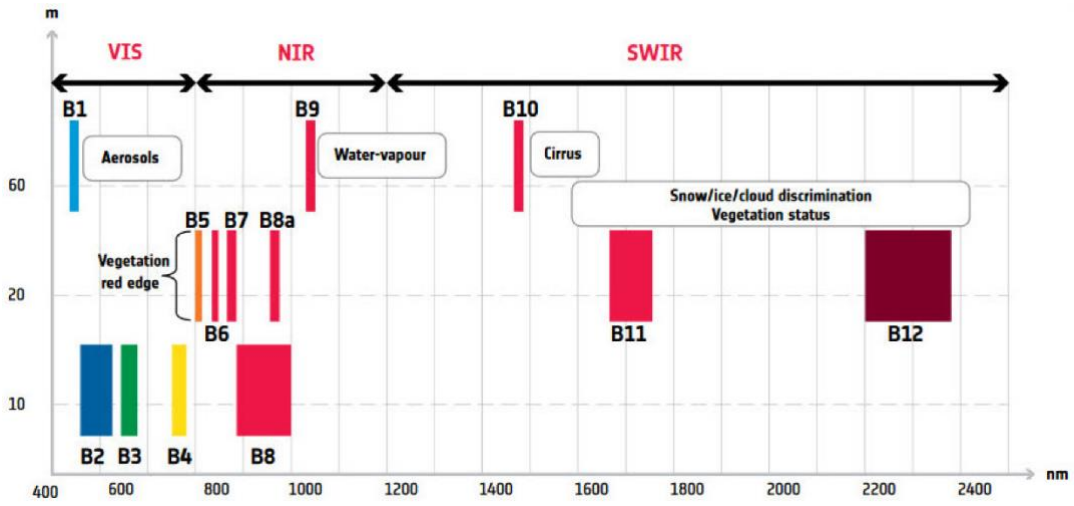


Figure 1: Spectral bands' resolution distances to wavelength distribution

In this study, we calculated several commonly used spectral indices (NDVI, EVI, EVI2, SAVI, NDWI) based on Sentinel-2 multispectral bands to comprehensively characterize vegetation and water features. These indices play a critical role in forest monitoring and tree species classification, as briefly described below:

1. **Normalized Difference Vegetation Index (NDVI)**: $NDVI = \frac{B8-B4}{B8+B4}$

   NDVI measures vegetation health based on the difference between red light (B4) and near-infrared (B8) reflectance. Healthy vegetation absorbs visible light strongly while reflecting near-infrared light significantly, making NDVI a reliable indicator of vegetation cover and vitality.

2. **Enhanced Vegetation Index (EVI)**: $EVI = 2.5\frac{B8-B4}{(B8+6\ x\ B4-7.5\ x\ B2+1)}$

EVI enhances NDVI by incorporating the blue band (B2) to correct for atmospheric scattering and soil background effects. It is particularly effective in high-biomass regions, such as dense forest canopies.

3. **Simplified Enhanced Vegetation Index (EVI2):** $EVI2 = 2.5\frac{\text{B8}-\text{B4}}{(\text{B8}+2.4\text{ x B4}+1)}$

EVI2 removes the blue band (B2) from the EVI formula, striking a balance between computational efficiency and atmospheric correction. It is suitable for scenarios where blue band data is unavailable or not needed.

4. **Soil-Adjusted Vegetation Index (SAVI)**: $SAVI = 2.5\frac{\text{B8}-\text{B4}}{(\text{B8}+\text{B4}+\text{L})}(1+L)$

Where L is the soil adjustment factor (set to 0.5 in this study). SAVI reduces the influence of soil brightness on vegetation indices, making it effective in sparsely vegetated areas.

5. **Normalized Difference Vegetation Index (NDVI):** $NDVI = \frac{\text{B8}-\text{B11}}{\text{B8}+\text{B11}}$

NDVI identifies water bodies and wetlands by comparing near-infrared (B8) and shortwave-infrared (B11) reflectance. Water bodies strongly absorb shortwave-infrared light, creating distinctive characteristics in this index.

Combining these indices with the original multispectral bands, we train models to differentiate tree species based on growth patterns, canopy water content, and soil background. NDVI and EVI assess vegetation health and coverage, SAVI adjusts for sparse vegetation, and NDVI monitors moisture changes. This integration enhances forest ecological classification, supporting robust tree species identification and health monitoring.

The dataset also includes hierarchical labels—from broadleaf/needleleaf types to genus and species—stored as strings (e.g., “broadleaf”, “needleleaf”) that are one-hot encoded for training. Such structured labeling enables gradual, efficient learning, especially with limited or imbalanced data, and facilitates feature or input-level fusion.

b. Data Preparation

The produced dataset had to be investigated further both in Google Earth Engine and in the Python code space. The investigation necessarily focused on data noise handling, imbalance adjusting, null value reasoning and curing.

First observation is that the certain band columns have null values in a repeatitive manner. The band numbers ranging from 1 to 8 refers to months selected in the Google Earth Engine code (see GEE code). This suggest accordingly with the figure 3, the 1st, 4th ,6th and 7th months might have been severed from cloud mask application. Meaning, due to high cloud coverage, the filter directly nulls the band readings that are positioned at cloud interfered locations. The high number of missing values in these band readings also causes null valued index results e.g. NDVI_6, EVI_6.

Before cleaning the data with python, it is better to reproduce these specific months with different cloud coverage values to see the differences. The correspoding months are March, April, June, August and September. An example of relating Google Earth Engine code is as follows.

```
var s2_Aug = processMonthlyData('2022-08-01', '2022-08-31', 30);
```

The situation can be solved by either adusting the cloud masking value to a higher index or extending if not completely changing the timeline or with interpolation techniques.

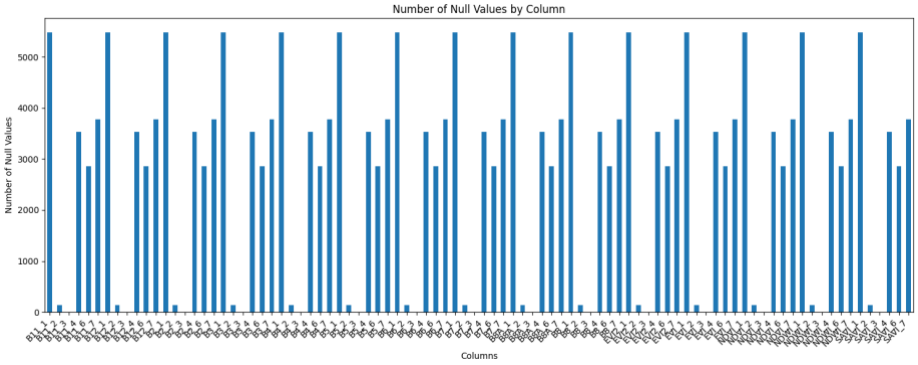


Figure 2. Demonstrates the specific timelines of band data has high number of null values.

The null value including bands with their respective timelines are visualised on Google Earth Engine and the points that are coinciding with empty/null areas are spotted which strengthen the suggestion of cloud masking resulted null values.

To address missing values in Sentinel-2 data, we implemented a two-stage spatiotemporal interpolation strategy:

1. Monthly Spatial Interpolation
    - First performed median composite processing for each month's data
    - Applied focal mean spatial interpolation using a circular kernel with 3-pixel radius
    - Utilized two iterations to fill small-scale data gaps
2. Cross-temporal Spatiotemporal Joint Interpolation
    - Spatial dimension: Applied a larger circular kernel with 5-pixel radius for broader spatial interpolation
    - Temporal dimension: Utilized the average of adjacent months (previous and next) for interpolation
    - Employed three iterations to ensure robust gap-filling

The result after interpolation:

- **The handling of the null values (interpolation)**

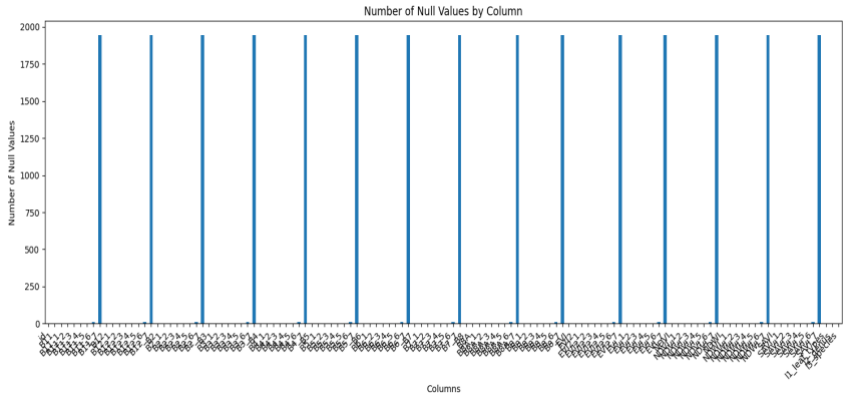


Figure 4: Null value distribution across columns after interpolation. The previously significant data gaps due to cloud masking were effectively reduced using a two-stage spatiotemporal interpolation strategy, ensuring higher data completeness and consistency.

c. Data Preprocessing & Augmentation

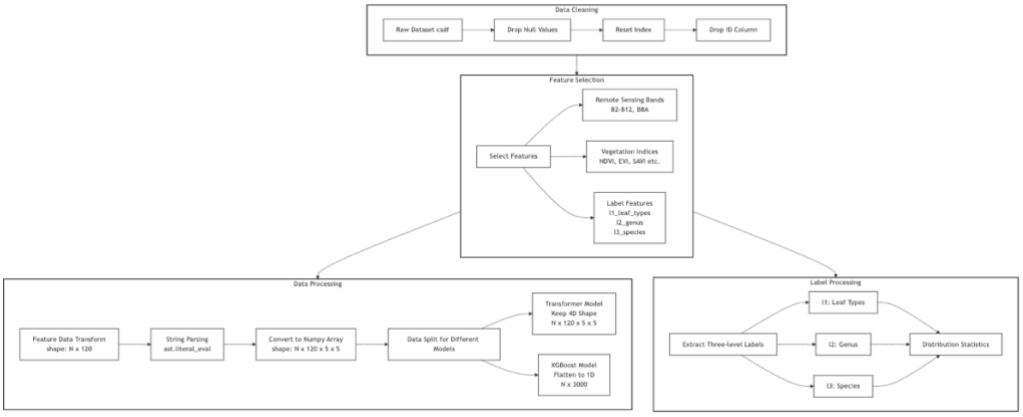


Figure 5: General data flow for preprocessing and augmentation, We took place in Python space. This flow diagram illustrates the steps involved in handling raw data, filling missing values, balancing imbalanced datasets, and preparing the data for machine learning models.

At this stage, Data Argumentation, imbalances, and noises addressed, and data set is separated to sentinel 2 data (X) and labels (Y) datasets.

At the first glance the l1 leaf types has shown in figure 6 (left), acceptable size difference of around 17500 needleleaf to 20000 broadleaf samples. However, the imbalances in genus and tree species classes have shown unacceptable imbalance of around 7:1 difference ratio in both genus and tree species.

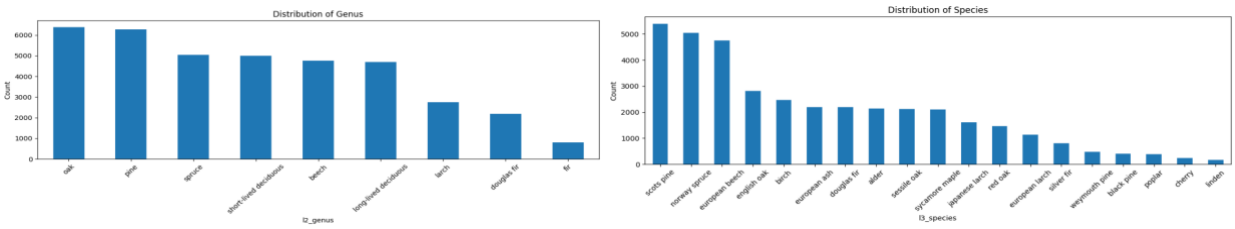


Figure 6: Distribution of L1 (Leaf Type) categories; Distribution of L2 (Genus) categories.

Theoretically there are various ways to handle this imbalance, such as using data augmentation methods. In our case, SMOTE as a synthetic data augmentation method is used for increasing the sample sizes of the lacking classes. However, the data augmentation and sampling must also align with the ml model and its data handling skills. A class weighting technique is implementable if all the classes have adequate level of samples for training but still some lacks in numbers or some ensemble techniques uses combined and altered training methods to use the dataset classes in more efficient ways. Lastly, algorithm specific methods are possible as CNN, XGBoost, random forest and more models have parameters that allows adaptation to the imbalanced dataset training.

In our case, the imbalance is too severe that sample size increasing is a must especially where the difference ration is higher than 1/3. However, the not too severely imbalanced classes are open to exploration and trail error kind of investigation.

One other important logic is considering the variations within the classes. The non new sample added classes might already have a good variation of data, whereas too much augmented classes might have values too repetitive of each other. This would result in slight overtraining of augmented classes and oblique boundary definitions between un resampled classes.

Graph Data Processing

Although GNN application is highly experimental for us, theoretically, the dataset of sentinel – 2 remote sensing band values hierarchically associated with 3 label types is a highly suitable study case. Compared to the logic of Convolutional networks that operates on grid or image-based logic, the graph dataset is free from such restriction and allows models to learn from a semantically enriched graph of nodes and edges. The graph data is prepared by manually creating nodes of trees, labels and using edges to connect tree nodes to their respective label nodes. While the label nodes we one hot coded with each node representing one specie, this procedure is done for all the labels. At the end, our labels represented as 19 specie nodes, 9 genus nodes and 2 leaf type nodes. The catch here was to use the directed edges for mapping the hierarchical labelling structure. The illustration below in figure 7 is representing this graph structure.

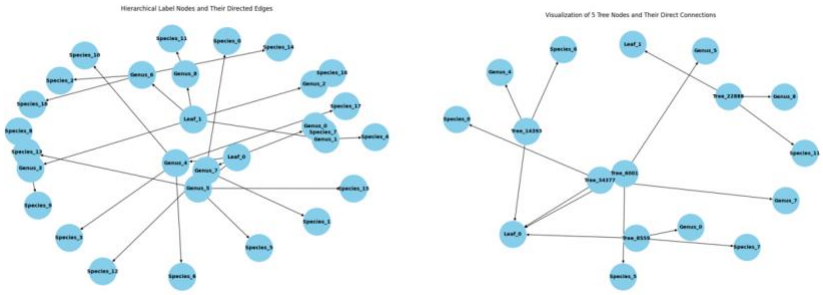


Figure 7: Graph structure visualization for hierarchical tree species classification showing label node connections (left) and tree-to-label node relationships (right).

The graph structure implements a single-direction directed graph, using unweighted edges to ensure equal importance across all connections. By encoding hierarchical relationships through directed edges between labels, the model learns taxonomic relationships alongside species classification. This integrated approach allows for species prediction without explicitly fusing leaf type and genus labels, as was necessary in traditional tabular models. Current limitations of the graph structure include, class imbalance in the

dataset, random shuffling of data during training, direct embedding of all Sentinel-2 features in tree nodes.

3. Model Selection

In this project, the machine learning methods: **Random Forest (RF)** and **XGBoost** are used as benchmark models, where their results and performance is compared against **Convolutional neural network (CNN), Vision Transformer based model(ViT)** and **Graph neural networks (GNN).**

Random Forest Classifier is an ensemble model combining bagging with feature randomness, distinguishing it from standard decision trees by focusing on feature subsets rather than all splits. Three key hyperparameters control model performance: node size, number of trees, and sampled features count, directly impacting training and over/underfitting behavior.

XGBoost[2] has emerged as a leading tool within the machine learning community. During CPU execution, the XGBoost algorithm utilizes parallel computing to construct trees sequentially. Instead of relying on the traditional stopping criteria initially, it applies the 'maxdepth' parameter and begins the tree pruning process from the bottom up. This method significantly enhances XGBoost's computational efficiency compared to other frameworks. Furthermore, XGBoost excels in automatically determining the best missing values through its training loss function, enabling it to effectively handle various sparsity patterns in the input data.

Convolutional Neural Networks [5] are well suited for tree classification while detecting the texture of patches and spectral features at the same time. They use convolutional layers to scan the image and extract local features, and pooling layers to reduce the spatial dimensions of the features. This allows CNNs to learn hierarchical representations of the image that are invariant to translations, rotations, and scaling's

Graph Neural Networks [3] The primary difference between CNNs and GNNs is that CNNs are made to operate on regular (Euclidean) ordered data. GNNs, on the other hand, are a generalized version of CNNs with different numbers of node connections and unordered nodes (irregular on non-Euclidean structured data).

Vision Transformer based model [4] is structured with an encoder and decoder framework that allows simultaneous processing of sequential information. One of the key factors behind its effectiveness is the multi-head self-attention (MHSA) component. This mechanism gives the model the ability to identify and understand complex relationships between different elements across the entire sequence, even when they are far apart from each other. This design enables ViT to process and analyze data more comprehensively than traditional sequential approaches.

4. Model Training

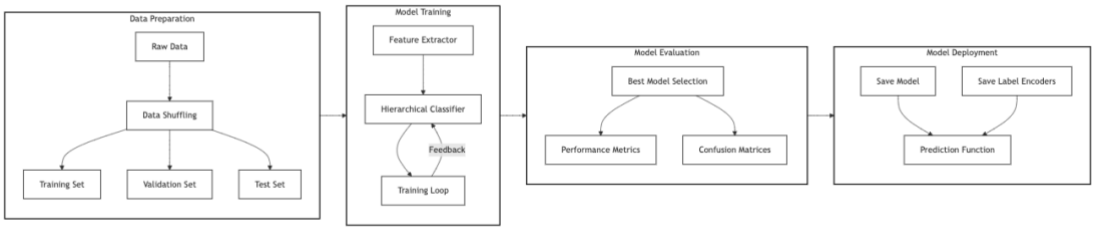


Figure 8: Workflow diagram of the tree species classification system, showing four main stages: data preparation (dataset splitting), model training (feature extraction and hierarchical classification), model evaluation (metrics and ensemble analysis), and model deployment (tree and save label functions).

The first step in implementing **XGBoost**, **Vision Transformer**, and other multilevel classification models was dataset division. We adopted a 70:30 train-test split ratio, considering our dataset size and training requirements. This ratio was chosen over the common 80:20 split to ensure sufficient test data while maintaining adequate training samples to prevent model underfitting.

Next, we describe the model architecture design. For the **Transformer** model, we built a Vision Transformer-based multi-level classifier consisting of a feature extraction module (120-channel input, 256-dimensional embedding, 4 Transformer layers) and sequential classifiers for leaf type, genus, and species. Training used AdamW (learning rate 5e-5), CosineAnnealingLR, and early stopping (patience=15), with hyperparameters tuned via iterative experiments.

For **XGBoost**, we designed a three-layer cascade classifier. The first layer (leaf type) uses conservative parameters (max depth 5, learning rate 0.01); the second layer (genus) increases complexity (max depth 6, learning rate 0.05); and the third layer performs fine-grained species classification while incorporating the predictions from the first two layers as additional features. Each layer employs early stopping (patience of 50, 20, and 5, respectively) and cross-validation, with SMOTE applied in the second and third layers to address data imbalance. Hyperparameters were determined through repeated experiments to balance performance and efficiency.

For the hierarchical classification task, the loss function design also differs. The Transformer model uses a hierarchical weighted cross-entropy loss ($0.2 \cdot \text{loss}_1 + 0.3 \cdot \text{loss}_2 + 0.5 \cdot \text{loss}_3$) with label smoothing and KL divergence to improve generalization, optimizing all layers end-to-end. In contrast, XGBoost uses a multiclass log-likelihood loss (objective='multi:softprob'), with each layer training its loss independently and using the previous layer's prediction probabilities as inputs, thereby transferring information between layers.

Random Forest Classifier: Initial data preprocessing involved augmentation, label alignment, and dataset integrity verification. Key features included bands B2-B8A, NDVI, and EVI. While temporal data (months 1-7) was less relevant for Random Forest compared to CNN and ViT architectures, experiments with time-series data showed slightly improved results. Although Random Forest performance could be enhanced through further optimization, XGBoost demonstrated superior results, leading us to focus on XGBoost and deep learning approaches (CNN, GNN, Transformers). The Random Forest implementation served as a baseline benchmark and helped deepen our understanding of the dataset.

Convolutional Neural Network: Three separate models were developed for classification tasks, implementing a progressive approach where previous label predictions are concatenated to aid training of subsequent, more complex models.

Model 1, focused on leaf type binary classification, uses sigmoid activation and binary cross-entropy loss. All three models share fundamental Keras components: convolutional layers, pooling, dropout, and dense layers. During training, Keras handles backpropagation while Adam optimizer adjusts weights to minimize loss. 'Same' padding maintains spatial dimensions across feature maps.

Common architecture across models includes three convolutional layers with increasing filter sizes (32, 64, 128) for complex feature capture, dropout layers for overfitting prevention, and globalAvgPooling to reduce 5x5 remote sensing data dimensions. Dense layers enable complex decision boundary learning. For multi-label tasks, models use categorical cross-entropy loss and softmax activation instead.

Graph Neural Network: The choice of a GNN model depends on the prediction task. Since our task required predicting node labels and the graph dataset is straightforward, we selected a simple Graph Convolutional Network (GCN) implemented with PyTorch Geometric. The model starts with a Conv1D layer that convolves node features, updates embeddings, and maps the input to a hidden dimension. A Conv2D layer repeats this process, and a fully connected layer produces the final output. ReLU activations are applied between layers for non-linearity. We used the Adam optimizer for its dynamic learning rate adjustment and initially applied cross-entropy loss—later to be replaced by a label hierarchy-aware custom method.

Training runs for 80 to 100 epochs (adjustable via manual learning rate settings) with dropout and batch normalization. Each epoch resets the gradients before backpropagation and parameter updates.

5. Network Tuning

Early on, we manually tuned hyperparameters for each model but found that, aside from the number of epochs—which notably affected training times—other adjustments had little impact without improved data preprocessing.

For hierarchical multilevel classification, we used different tuning strategies for Vision Transformer and XGBoost. In our cascaded **XGBoost** model, hyperparameters were optimized per layer according to task complexity. Key strategies included:

1. Stratified 5-fold cross-validation to ensure robustness,
2. Layer-specific early stopping using a separate validation set, and
3. Adaptive settings assigning higher learning rates and tree depths for more complex tasks.

We maintained efficiency (using SMOTE for imbalanced classes and fixing subsample/colsample_bytree at 0.8) while continuously monitoring mlogloss; each layer's predictions were validated before serving as features for the next.

For the **Vision Transformer**, we pruned parameters to reduce complexity by lowering the embedding dimension (512 to 256), depth (8 to 4), attention heads (8 to 4), and MLP layers (4 to 2). Additionally, we adjusted the hierarchical weighted loss from 0.3:0.3:0.4 to 0.2:0.3:0.5 to place greater emphasis on deeper layers. Figure 9 shows steadily declining and stabilizing training and validation losses, and converging accuracy curves for species classification, indicating effective optimization with minimal overfitting.

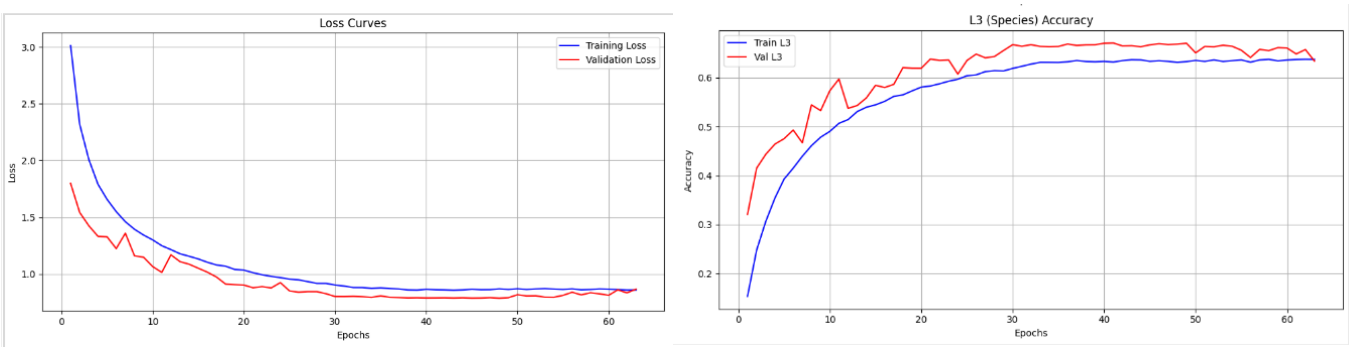


Figure 9: Training dynamics of the Vision Transformer model. (a) Loss curves showing training and validation loss convergence over epochs. (b) L3 (Species) level classification accuracy curves demonstrating model performance on training and validation sets.

Random Forest Classifier: Model showed imbalance in l2_genus labels. l1 labels performed moderately, hierarchical fusion underperformed with 85% target. l3 classification (19 classes) optimized via GridSearchCV 4-fold: max_depth=12, max_leaf_nodes=30, n_estimators=100.

Convolutional Neural Network: Three models optimized separately. Model 1 (3 conv layers) implemented 2 dropout layers after testing, achieving 65-85% accuracy on l1 test data. Manual class weights failed to resolve prediction bias. L2 predictions remained lowest performing, impacting species accuracy. Due to severe class imbalance, weights manually adjusted based on test results rather than misleading validation metrics. Learning rate tuned via Adam optimizer to address overfitting and imbalance. Model 3 features 2 conv blocks (3x3 kernel), ReLU activation, batch norm, and max pooling. Filter sizes: 32→64→128. L1 and L2 predictions concatenated with global avg pooled output, feeding into 128-neuron ReLU layer before final classification.

Graph Neural Network: GNN network's tuning is made in loss function part. Since the graph data has directed edges connecting the labels in a hierarchical way, this can be used to make the training process dynamically tuning the loss values not only depending on cross entropy but also the hierarchical structure of the labels.This small code snippet of dictionary shows the hierarchical structure of which the classification penalties must be applied.

```
hierarchy = {
  "species": {"genus": 0.5, "leaf": 0.2},
  "genus": {"leaf": 0.3},
  "leaf": {}
}
```

This way, each epoch is more aware of classification mistakes whereas, backpropagation and optimization processes are more effective.

One last touch was to add a manual learning rate input to Adam optimizer which seemed suffice at 0.01.

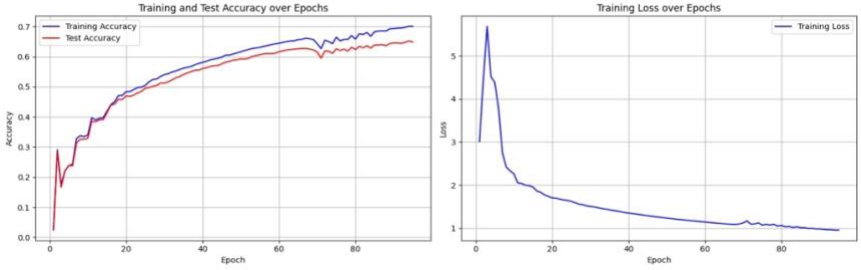


Figure 10: The changes in accuracy (left) and loss values (right) of the Graph Neural Network (GNN) during the training process across epochs.

At the end, it is observed from figure 10 that the model as a foundation is solid and with more regularization and more fine tuning, it can be trained further without overfitting. Our model started to show overfitting behaviour after 60 epochs, however at the earlier runs this number was around 80. Loss function also indicated a slowing but somehow gradual learning even after 60 epochs, still subject to overfitting

6. Experimental Results

In the test evaluation of tree species recognition, we compared methods including XGBoost, ViT, CNN, RF (Random Forest), and GNN. Overall, ViT performed best in fine-grained tree species recognition(shown in figure 11), thanks to its powerful modeling capabilities for multi-band and cross-seasonal features. XGBoost followed closely behind; while slightly inferior to ViT in some categories, it showed unique advantages in speed and handling missing values. CNN performed well in texture and local feature extraction but was relatively weak in modeling cross-seasonal time series features.

Experimental results show that temporal interpolation is crucial for improving overall model accuracy. Taking XGBoost and ViT as examples, after temporal interpolation processing, their test set accuracy (OA) improved significantly (XGBoost from 0.61 to 0.64, ViT from 0.56 to 0.65). This confirms that information loss due to cloud masking can be compensated to some extent through interpolation techniques, while also indicating that rich time series information aids in distinguishing between fine-grained categories in large-scale remote sensing scenarios.

However, under the multi-level (L1, L2, L3) label system, Random Forest (RF) consistently failed to achieve test accuracy above 20% for L2 categories and was therefore not continued in subsequent fusion experiments. Its final results on the test set only reached L1: 0.62, L2: 0.16, L3: 0.40. CNN showed promising predictions for L1 and L3 but still had notable difficulties distinguishing L2 categories, possibly due to data imbalance at this level. Even with fusion techniques, CNN's accuracy on the test set reached 80%, 0.20%, and 55% for L1, L2, and L3 respectively, still underperforming compared to ViT and XGBoost's overall performance.

On the other hand, GNN (Graph Neural Network) was designed to directly predict L3 categories in one step without requiring hierarchical prediction and concatenation, taking into account the characteristics of remote sensing images and label systems. It achieved approximately 65% L3 classification accuracy on the training set and about 63% on the test set, showing considerable potential. Overall, while each method has its advantages, ViT's global attention mechanism remains most competitive for fine-grained recognition with large-scale data and high category imbalance, while XGBoost deserves attention for its efficiency and handling of missing values. GNN provides a viable path for further mining data relationships, showing promising performance especially in scenarios not requiring strict hierarchical training.

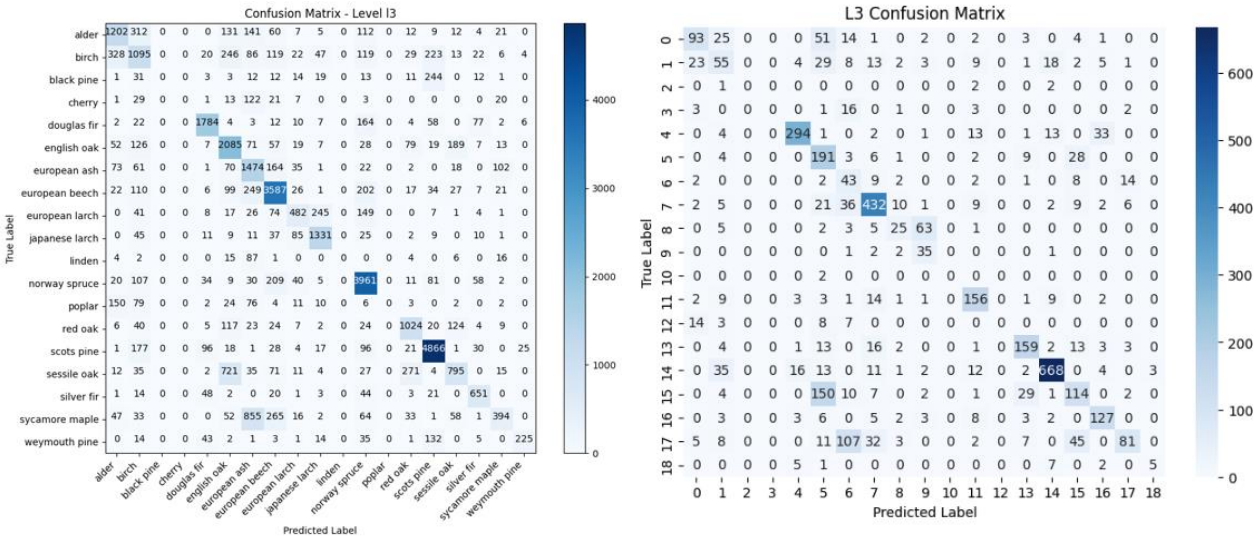


Figure 11: Confusion matrix for validation after applying time interpolation(left); Confusion matrix for the same model on test(right).

7. Discussion

From an overall perspective, the **Hierarchical** classification framework is effective for tree species identification, especially when the number of categories increases and the spectral and temporal differences between categories become more subtle. The multi-level output (L1, L2, L3) can fully utilize information from coarse to fine levels, helping the model focus on more precise feature spaces.

- **Potential reasons why ViT outperforms traditional machine learning and CNN**: The multi-head self-attention mechanism can simultaneously focus on important differences in both temporal and spectral dimensions. Compared to the problems of feature sparsity or insufficient convolution receptive fields that CNN or XGBoost encounter with large-scale feature inputs, Transformer has better global modeling capabilities.
- **XGBoost's advantages in speed and usability**: Compared to deep networks, XGBoost offers faster training speed and superior interpretability. For scenarios requiring rapid iteration or feature importance tracing (such as temporary monitoring or forest disaster emergency response), XGBoost remains a highly valuable model.
- **GNN's potential and limitations**: Converting hierarchical labels into nodes and utilizing hierarchical relationships for graph construction is an innovative approach. However, for large-scale data with extremely unbalanced class distributions, further graph structure optimization and regularization techniques, such as adaptive edge weights or hierarchy-based graph contrastive learning, are needed to fully leverage GNN's advantages.

8. Conclusions

In this project, we combined multiple deep learning and traditional machine learning methods to classify a hierarchy of tree species through multi-temporal phase Sentinel-2 data, demonstrating the effectiveness of the hierarchical learning framework in the complex task of tree species recognition. Vision Transformer obtains the optimal accuracy at the L3 level, indicating that the global self-attention mechanism has significant advantages in dealing with multi-band and multi-temporal remote sensing data; while XGBoost provides competitive results at a lower training cost, which is suitable for scenarios requiring high classification speed and interpretability. Realistic problems such as data imbalance and

cloud occlusion also demonstrate that their impact on the model is large, emphasising the need for data preprocessing and fusion of multi-temporal data. In the future we adjust the parameters of the vision transformer by continuing to change them in order to further improve the accuracy and stability of the tree classification.

Contributions

Since we are only 2 people in this group project, we had to work on the each task together. Especially, the data processing and model tuning tasks required discussions and cooperative working in every step.

1. Data extraction + interpolation (Jiaqi Li)
2. Data augmentation (Canberk Yalçıklı)
3. XGBoost + ViT (Jiaqi Li)
4. Random Forest Classifier + CNN + GNN (Canberk Yalçıklı)
5. Report (Jiaqi Li + Canberk Yalçıklı)
6. Presentation (Jiaqi Li + Canberk Yalçıklı)

References

[1] Phiri, D., Simwanda, M., Salekin, S., Nyirenda, V. R., Murayama, Y., & Ranagalage, M. (2020). Sentinel-2 data for land cover/use mapping: A review. *Remote Sensing*, *12*(14), 2291.

[2] Kwenda, C. G. M. F.-D. J. (2023). Forest Image Classification Based on Deep Learning and XGBoost Algorithm.

[3] Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., & Sun, M. (2020). Graph neural networks: A review of methods and applications. *AI Open*, *1*, 57–81. https://doi.org/10.1016/j.aiopen.2021.01.001

[4] Marjani, M., Mohammadimanesh, F., Mahdianpari, M., & Gill, E. W. (2025). A novel spatio-temporal vision transformer model for improving wetland mapping using multi-seasonal Sentinel data. *Remote Sensing Applications: Society and Environment*, *37*, 101401. https://doi.org/10.1016/j.rsase.2024.101401

[5] Sharma, N., Jain, V., & Mishra, A. (2018). An analysis of convolutional neural networks for image classification. *International Conference on Computational Intelligence and Data Science*, *132*(1), 377–381.

[6] Demonstrates the spectral bands' resolution distances to wavelength distribution. Cited from: Lolli, S., Alparone, L., Arienzo, A., & Garzelli, A. (2024). Characterizing Dust and Biomass Burning Events from Sentinel-2 Imagery. Atmosphere, 15(6), 672. https://doi.org/10.3390/atmos15060672