Tile Compression and Embeddings for Multi-Label Classification in GeoLifeCLEF 2024

Anthony Miyaguchi[1,*], Patcharapong Aphiwetsa[1] and Mark McDuffie[1]

[1]Georgia Institute of Technology, North Ave NW, Atlanta, GA 30332

Abstract

We explore methods to solve the multi-label classification task posed by the GeoLifeCLEF 2024 competition with the DS@GT team, which aims to predict the presence and absence of plant species at specific locations using spatial and temporal remote sensing data. Our approach uses frequency-domain coefficients via the Discrete Cosine Transform (DCT) to compress and pre-compute the raw input data for convolutional neural networks. We also investigate nearest neighborhood models via locality-sensitive hashing (LSH) for prediction and to aid in the self-supervised contrastive learning of embeddings through tile2vec. Our best competition model utilized geolocation features with a leaderboard score of 0.152 and a best post-competition score of 0.161. Source code and models are available at https://github.com/dsgt-kaggle-clef/geolifeclef-2024.

Keywords

GeoLifeCLEF, LifeCLEF, remote sensing, contrastive learning, multi-label classification, tile2vec, discrete cosine transform, locality-sensitive hashing

1. Introduction

GeoLifeCLEF [1] is a task organized within the LifeCLEF lab [2] at the CLEF 2024 conference, with the goal of predicting which plant species are present and absent at specific locations given spatial and temporal remote sensing data. Modeling species density distributions can be helpful in biodiversity management and conservation.

We explore methods to solve the posed multi-label classification task and incorporate unsupervised methods to build useful representations from the data. We propose a pipeline that pre-computes tiles from raw GeoTIFF images and stores a compressed version on disk to speed up the training process. We utilize metadata to build baseline geolocation models and indices for nearest neighbor queries. Our models utilize convolutional neural networks to exploit spatial information, spectral representations, and co-located bands of remote sensing data. We also explore the use of unsupervised methods to learn representations for knowledge transfer between two different datasets with similar semantics.

2. Related Works

The GeoLifeCLEF 2023 had seven submissions along with baseline results by the organizers [3]. Most participants focused on bioclimatic rasters and satellite imagery, leveraging Convolutional Neural Networks (CNN) like ResNets [4] for feature extraction. Participants combined rasters and trained separate models for prediction [5]. Spatial coordinates (longitude/latitude) were commonly used with models like K-Nearest Neighbors (KNN) and Random Forest, yielding surprisingly good results. However, the combination of diverse modalities provided in the dataset was rare, with only one participant utilizing time-series data with a 1D Convolutional Network.

CLEF 2024: Conference and Labs of the Evaluation Forum, September 9-12, 2024, Grenoble, France

[*] Corresponding author.

acmiyaguchi@gatech.edu (A. Miyaguchi); paphiwetsa3@gatech.edu (P. Aphiwetsa); mmcduffie8@gatech.edu (M. McDuffie)

0000-0002-9165-8718 (A. Miyaguchi)

3. Overview

```
{
  "type": "Polygon",
  "coordinates": [
      [
          [-32.26344, 26.63842],
          [-32.26344, 72.18392],
          [ 35.58677, 72.18392],
          [ 35.58677, 26.63842],
          [-32.26344, 26.63842],
      ]
  ],
}
```

Figure 1: GeoJSON polygon definition.

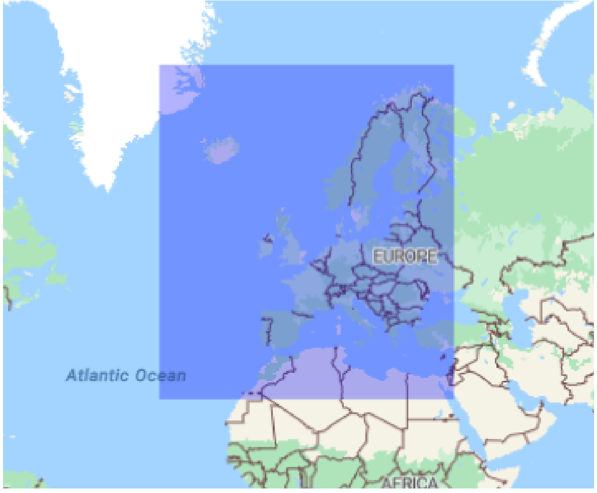

Figure 2: Polygon region overlayed on a map.

The competition has three main components to the dataset. The first are the metadata associated with the competition comprising a presence-only training, presence-absence training, and presence-absence test set. The metadata provides a mapping between location and the species labels available for supervised training. The second are the remote-sensing and raster data provided in pixel format. The final component is time series data containing quarterly environmental data over a 20 year period.

The presence-only training dataset comprises 5,079,797 examples over 3,845,533 survey sites distributed across Western Europe. The dataset is drawn from crowd-sourced data with potential gaps in the reported species. The presence-absent dataset has stricter semantics – species not included in the survey are presumed absent. The training set has 1,483,637 examples over 88,987 sites, while the test set has 4,716 sites. The datasets includes an identifier for the survey site alongside latitude and longitude, as per the schema in Listing 1. We compute a projection into EPSG 3035, which allows for Euclidean distance between sites in units of meters.

The majority of available data are raster and satellite imagery. The provided GeoTIFF files contain various measures such as elevation, roads, population, and soil. The GeoTIFF files are bounded by a GeoJSON polygon that covers Western Europe as seen in Figures 1 and 2. RGB-NIR satellite imagery is directly available as 128 × 128 pixel tiles associated with each survey site.

```
 |-- dataset: string (nullable = true)
 |-- surveyId: integer (nullable = true)
 |-- lat_proj: double (nullable = true)
 |-- lon_proj: double (nullable = true)
 |-- lat: double (nullable = true)
 |-- lon: double (nullable = true)
 |-- year: integer (nullable = true)
 |-- geoUncertaintyInM: double (nullable = true)
 |-- speciesId: double (nullable = true)
```

Listing 1: Metadata schema for the competition.

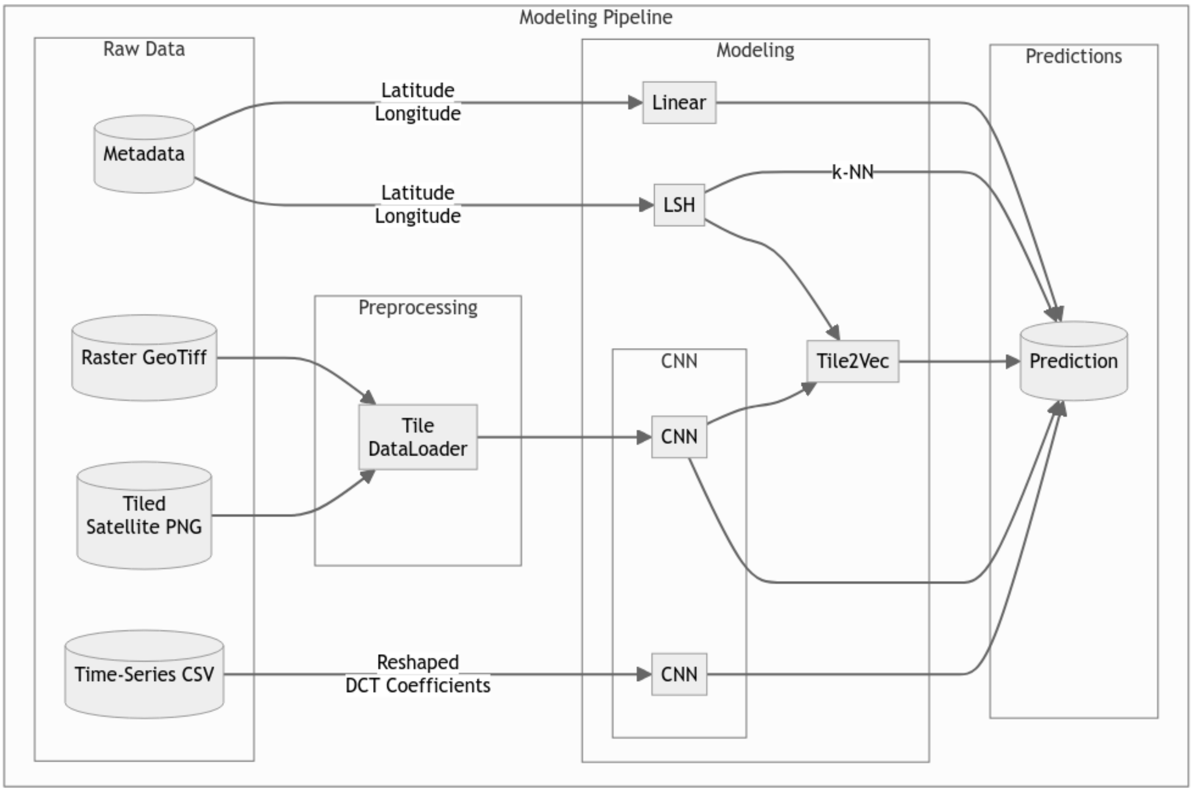

Figure 3: Overview of the data and modeling pipeline. Raw data is pre-processed to maintain survey site per row semantics, with 2D DCT coefficients as features. Data is cached as columnar parquet files in cloud storage for efficient access.

4. Processing Pipeline

We explore several solutions for the multi-label classification problem. We use Luigi [6] as our workflow management tool, which provides idempotent directed acyclic graphs (DAGs) of tasks. We use Spark [7] to perform data extraction, transformation, and loading (ETL) from tarred images and CSV files to columnar parquet files. We use PyTorch as our deep learning framework and use PyTorch-Lightning to simplify the training and inference process. We use Petastorm to preprocess and load data into Torch. We use Weights and Biases to log hyperparameters and metrics.

4.1. Satellite and Raster Data

The competition organizers provide point data for each survey site in a pre-computed train CSV file. Our experiments focus on 128x128 pixel tiles extracted from provided GeoTIFF files for use in a supervised learning setting. Significant in-memory overhead tiling exists because we need to store a 128x128 matrix of integers or floats for each survey site. Memory access patterns can cause significant slow-downs if we need to fetch them from disks often.

We fork the official `plantnet/GeoLifeCLEF` data loaders to pre-compute tiles for each of the provided GeoTIFF under 1GB. Certain rasters do not fit into memory (e.g., elevation raster at 11GB); therefore, we omit them from our experiments. We only compute the tiles associated with the survey identifiers in our metadata, which helps limit the size of the resulting dataset.

4.2. Tile Compression via Discrete Cosine Transform (DCT)

We compute the 2D-DCT on the resulting tile images and keep low-frequency coefficients as features in downstream modeling. We implement a PySpark wrapper around the ND-DCT to supplement

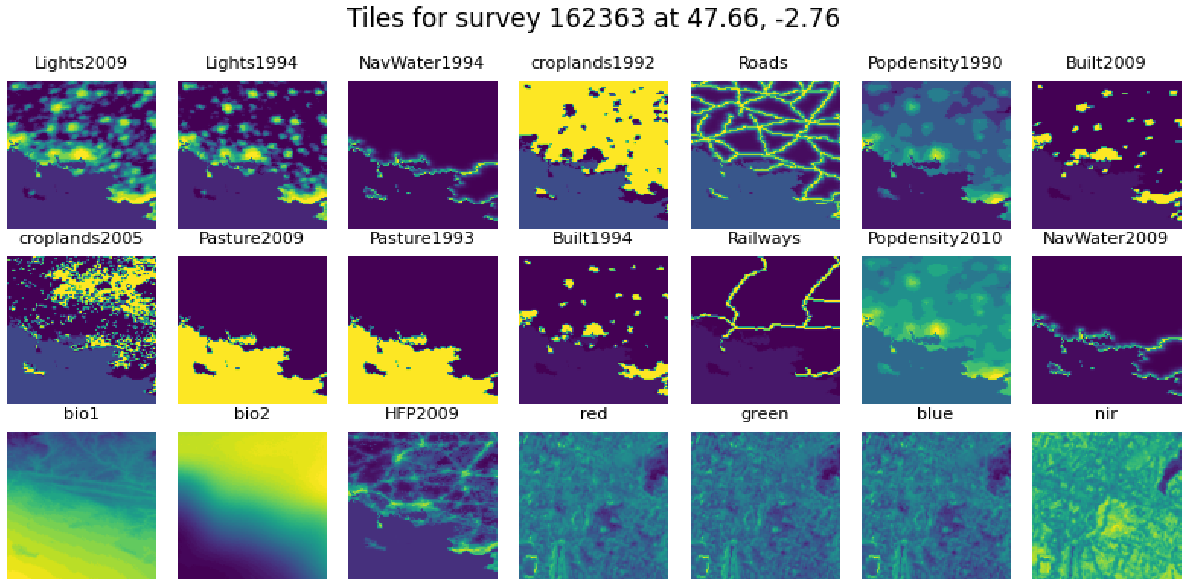

Figure 4: Example of a tiled raster image. The image is a 128x128 tile of the RGB-NIR satellite imagery. The image is associated with a survey site and is used as input to the model.

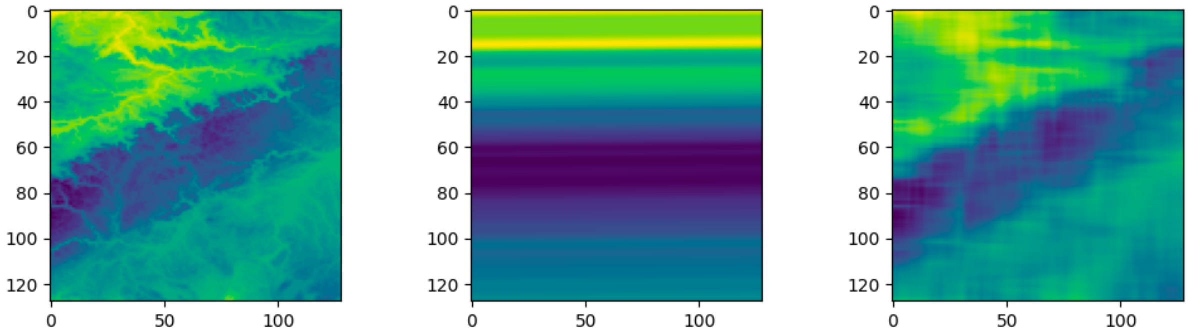

Figure 5: Example of low-pass filtering using the DCT. (a) Original bio1 raster image. (b) Low-pass filter using the first 50 coefficients of the 1D-DCT, reshaping on the first axis (row-major order). (c) Low-pass filter using the 2D-DCT using the top-left 8x8 coefficients.

the standard library 1D-DCT implementation for feature pre-processing. We lose significant spatial information if we perform filtering in 1D coefficient-space as seen in Figure 5.

4.3. Time-Series Data

Time series data is treated as another layer in the network by pre-processing the data to obtain DCT coefficients. We have access to quarterly time-series data for each survey site over 20 years. Some sites have missing data, which are padded with zeros. We compute the 1D DCT on the time series data and keep the first 64 coefficients in the transformed space, which parallels the 8x8 2D-DCT coefficients extracted from the raster data. The original time-series data and its DCT are displayed side by side in figure 6

4.4. Data Augmentation

We apply augmentations to our data to encourage model invariance to rotation and reflection. Some such transforms include rotating and flipping images before sending them into a model. Equivalent augmentations exist in frequency space. For example, a 90-degree rotation in pixel space is equivalent to the transpose of the 2D-DCT coefficients. We can flip the image in pixel space by alternating the

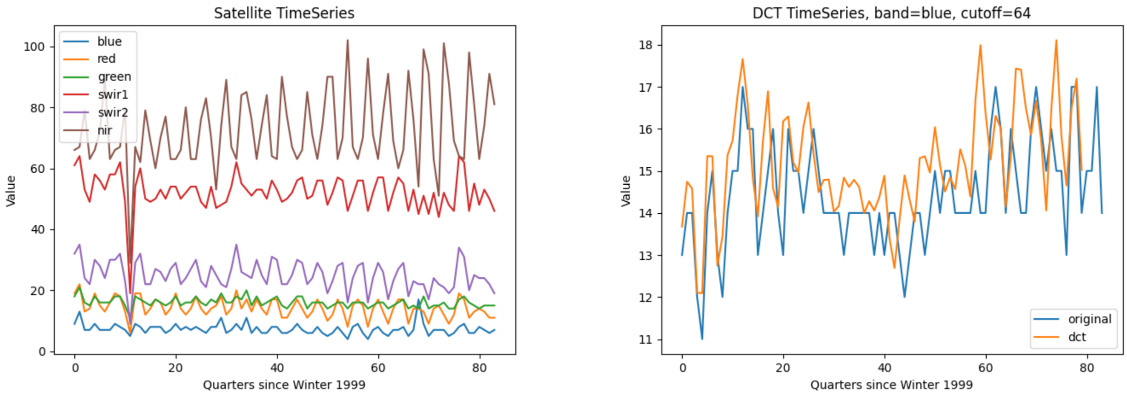

Figure 6: Example of time-series data and its DCT. (a) Original time-series data. (b) Blue band time-series approximated using DCT.

```
class DCTHorizontalFlip:
    def __init__(self, k=8):
        self.odd_factor = -torch.ones((k, k))
        for i in range(0, k, 2):
            self.odd_factor[i, :] = 1

    def forward(self, X):
        return X * self.odd_factor
```

Figure 7: Augmentation of 2D-DCT coefficients that flips the image across the horizontal axis in the spatial domain.

signs of the 2D-DCT coefficients along a given axis. Rotations and flips along the axis give us enough flexibility to implement useful symmetries in the data to improve model generalization.

4.5. Locality-Sensitive Hashing for Nearest Neighbor Queries

Species at sites that are close together should intuitively have similar distributions of plants. The projected latitude and longitude have physical meaning via euclidean distance, so we can build a nearest-neighbor model and rank species per survey site by frequency in a neighborhood. We use locality-sensitive hashing (LSH) with random hyperplane projections to build a k-NN model [8], with the hyper-parameters for bucket length and number of hash tables set to 20 and 5, respectively. We can find the top-k nearest survey sites in linear time for any given survey site using the LSH model. The approximate nearest neighbor self-join is performed with a 50km cutoff, and the results stored on disk for downstream modeling.

5. Models

5.1. Nearest Neighbor Model

We generate predictions by querying the LSH model built on survey site locations across presence-only and presence-absent datasets in a 50km radius. For each survey site, we limit either the number of neighbors or the distance to the nearest neighbor. Our nearest neighbor models (NN) consider all neighbors within a 5km, 10km, and 50km radius. The k nearest neighbors model (k-NN) considers the top 10 neighbors for each survey site and all of the species reported in the neighbors.

5.2. Geolocation Model

We model the relationship between geo-spatial metadata (i.e. projected latitude and longitude) and the species labels. We learn a linear and non-linear multi-label classification problem, and treat this model as a baseline for remote-sensing based models. The linear model uses a single linear layer that maps the feature space into the output space. The nonlinear model uses a two-layer model, with a $\mathcal{R}^{256}$ latent space in the first layer and a linear layer to map to the output space. We add random noise to latitude and longitude with a mean of 5km to increase generalization.

5.2.1. Tile CNN Model

The processed remote-sensing data is represented by a multi-dimensional array with one indexing dimension for the layer type and two spatial dimensions, making this a natural fit for CNNs. We pre-process satellite and raster data by generating 128x128 pixel tiles, then applying the 2D-DCT to filter the 8x8 coefficients representing the lowest spatial frequencies. We use the coefficients as inputs to a CNN model that learns a mapping to a multi-label classification task. We additionally run a few experiments where we apply the inverse 2D-DCT to the coefficients to recover a multi-layer 128x128 pixel image. We learn models on the presence-absent dataset, since the dataset is representative of the test.

We build CNN models convolve input layers with a 3x3 kernel with padding, followed by a 1x1 convolution and linear layer to map into latent space. We map latent space to a linear layer with the number of classes as the output space. Batch normalization is applied at every convolution for numerical stability, and ReLU activation for non-linearities. We also experiment with alternative parameterizations of the network, notably replacing the custom CNN with a pre-packaged efficientnetv2 backbone.

We experiment with several models that take in different input data. The simplest model uses RGB-NIR satellite imagery in four channels. We then incorporate the 13 MODIS landcover layers and 19 bio-climatic rasters. We then hand-choose specific layers from the rasters to remove redundancy, specifically layers 9, 10, and 11 from MODIS and the years 2001, 2010, and 2019 from the bio-climatic rasters. The other layers of the MODIS dataset correspond to legacy classification schemes and confidence bands, which are likely not useful to our model. We choose three bands from bio-climatic rasters since the years are evenly spaced across the set.

We build a model using the provided time-series RGB-SWIR data. We reuse the same architecture as the satellite imagery data by reshaping the first 64 coefficients of the time-series DCT into a 8x8 tensor and then applying the same convolutional layers. The semantics are not necessarily the same as the 2D-DCT coefficients, but we hypothesize learned structure from this basis despite the lack of spatial symmetry.

5.2.2. Tile2Vec Model

Tile2Vec [9] is a self-supervised learning technique that learns embeddings of tiles of satellite imagery. The Tile2Vec model utilizes a spatially-aware sampling procedure and triplet loss to learn a low-dimensional embedding that preserves metric distances via the triangle inequality. The triplet loss is $L(t_a, t_n, t_d)$ with a margin m where f_θ maps data to a d-dimensional vector of real numbers using a model with parameters θ.

$$L(t_a, t_n, t_d) = [||f_\theta(t_a) - f_\theta(t_n)||_2 - ||f_\theta(t_a) - f_\theta(t_d)||_2 + m]_+ \tag{1}$$

We obtain triplets from the presence-only datasets by querying the LSH model to sample one million pairs of tiles within 100km in the presence-only dataset. We generate a distant neighbor by randomly selecting a tile from dataloader batch. We train a tile2vec model using the CNN architecture described in section 5.2.1 without a classifier head in $\mathcal{R}^{256}$ latent-space. We experiment with a multi-objective loss incorporating the sum of triplet and ASL losses, using labels for each survey site by aggregating all species within each site's radius. The classifier adds a linear layer to the learned latent space and training with the ASL loss on the presence-absent dataset.

5.3. Model Evaluation and Loss Functions

The competition uses the F1-micro score to evaluate models, and we use the same metric in training. We utilize to compute the F1 score. We evaluate the model during training and validation using `MulticlassF1Score(average="micro")` from the `torchmetrics` library, with a 90-10 train-validation split of the presence-absent dataset.

5.3.1. Binary Cross-Entropy

Binary cross-entropy is a loss function used for binary classification. In a multi-label setting, each label as an independent binary classification problem. We use the loss function that accepts logits as input for numerical stability, which is necessary to achieve acceptable convergence.

$$L = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{2}$$

5.3.2. Asymmetric Loss (ASL)

The asymmetric loss [10] penalizes false positives and false negatives differently than the binary cross-entropy loss. The loss is defined in terms of the probability of the network output p, and hyper-parameters γ_+ and γ_-. Setting $\gamma_+ > \gamma_-$ emphasizes positive examples, while setting both terms to 0 yields binary cross-entropy. Easy negative samples are dynamically down-weighted, and hard thresholded samples are ignored.

$$ASL = \begin{cases} L_+ = (1-p)^{\gamma_+} \log(p) \\ L_- = (p_m)^{\gamma_-} \log(1-p_m) \end{cases} \tag{3}$$

We sweep over parameters $\gamma_+ \in \{0,1\}$ and $\gamma_- \in \{0,2,4\}$. The default values are $\gamma_+ = 1$ and $\gamma_- = 4$.

5.4. Hill Loss

The Hill loss [11] is a loss function designed for robust multi-label classification with missing labels. The loss is defined as a weighted mean-squared error (MSE), where the weight modulates potential false negatives.

$$\begin{aligned} \mathcal{L}^-_{\text{Hill}} &= -w(p) \times MSE \\ &= -(\lambda - p)p^2. \end{aligned} \tag{4}$$

The implementation provided by the authors provides the following form with default values of $\lambda = 1.5$ and $\gamma = 2$.

$$\mathcal{L}^-_{\text{Hill}} = y \times (1-p_m)^{\gamma} \log(p_m) + (1-y) \times -(\lambda - p)p^2 \tag{5}$$

5.4.1. sigmoidF1

The sigmoidF1 loss [12] optimizes the F1 score directly by creating a differentiable approximation of the F1 score. We first define the terms true positive, false positive, false negative, and true negative as a function of the sigmoid function.

$$\begin{aligned} \tilde{tp} &= \sum \mathbf{S}(\hat{\mathbf{y}}) \odot \mathbf{y} \quad \tilde{fp} = \sum \mathbf{S}(\hat{\mathbf{y}}) \odot (1-\mathbf{y}) \\ \tilde{fn} &= \sum (1-\mathbf{S}(\hat{\mathbf{y}})) \odot \mathbf{y} \quad \tilde{tn} = \sum (1-\mathbf{S}(\hat{\mathbf{y}})) \odot (1-\mathbf{y}) \end{aligned} \tag{6}$$

where $\mathbf{S}(\hat{\mathbf{y}})$ is the sigmoid function applied to the model's output $\hat{\mathbf{y}}$.

$$S(u;\beta,\eta) = \frac{1}{1+\exp(-\beta(u+\eta))} \tag{7}$$

Then we define the F1 score as a function of the true positive, false positive, and false negative terms.

$$\mathcal{L}_{\widetilde{F1}} = 1 - \widetilde{F1}, \quad \text{where} \quad \widetilde{F1} = \frac{2\widetilde{tp}}{2\widetilde{tp} + \widetilde{fn} + \widetilde{fp}} \tag{8}$$

We are given two hyper-parameter $S = -\beta$ and $E = \eta$. For tuning we sweep over parameters $S \in \{-1, -15, -30\}$ and $E \in \{0, 1\}$ as suggested in the author's experiments. The default values are $S = -1$ and $E = 0$.

6. Results

We report the performance of our models using the hidden test set on the Kaggle competition leaderboard. For torch-based models, predictions are made in two ways: top-k and threshold. The top-k method selects the top-k species with the highest probability, while the threshold method selects species with a probability greater than a threshold. We set k to 20 and the threshold to 0.5 for all relevant models.

6.1. Nearest Neighbor Model

We report nearest neighbor models in table 1. The k-NN models perform better than our NN models, likely due to filtering out noise from different thresholds. We observe that when we do not limit by the number of neighbors, larger distance thresholds lead to worse performance, with a difference of 0.10 to 0.08 when going from 5km to 50km. The opposite is true once we keep the top 10 neighbors, where there is a small improvement in score as we increase the distance threshold.

Table 1
Nearest neighbor Kaggle leaderboard scores using the LSH model on survey site distances in kilometers. We use cutoffs of 5km, 10km, and 50km to limit the number of neighbors. The k-NN model limits the number of neighbors to the top 10.

model	private	public
NN 50km	0.08638	0.08785
NN 10km	0.10719	0.10149
NN 5km	0.11059	0.10718
k-NN 50km	0.12919	0.1251
k-NN 10km	0.12545	0.11998
k-NN 5km	0.12219	0.11746

6.2. Geolocation Model

The projected latitude and longitude provide a relatively strong signal for the multi-label species classification tasks, with a score of 0.161 on the public leaderboard when using the top-k results in table 3. Our experiments show that the linear model performs poorly as per table 2. However, adding a non-linear layer to the model increases the performance by a large margin. For example, models trained with BCE loss without class weights go from 0.03 to 0.14 when adding a non-linear layer.

During training of BCE models, we observed that while the loss was decreasing between training and validation sets, the validation F1 score would peak on the first epoch and then decrease over time. We suspect that BCE loss has difficulty with the class imbalance even when given explicit class weights. ASL increases the validation scores by a wide margin, likely because of the dynamic weighting behavior. While the Hill and sigmoidF1 losses report improvements over ASL in the experimental setting in literature, we find that default hyper-parameters perform worse than BCE loss. It's possible hyper-parameter tuning could increase performance over ASL. For the rest of the experiments, we focus on ASL as the primary loss due to its performance and robust parameterization.

Table 2
Scores for different losses on geolocation based models. The scores provided are collected from late submissions on the public leaderboard using the threshold method of prediction.

Loss	**linear**	**nonlinear**
BCE (weights= True)	0.0259	0.14523
BCE (weights=False)	0.03011	0.14398
ASL ($\gamma_- = 4, \gamma_+ = 1$)	0.00042	0.15768
ASL ($\gamma_- = 4, \gamma_+ = 0$)	0.00169	0.15619
ASL ($\gamma_- = 2, \gamma_+ = 1$)	0.00064	0.14963
ASL ($\gamma_- = 2, \gamma_+ = 0$)	0.0008	0.15414
ASL ($\gamma_- = 0, \gamma_+ = 1$)	0.00635	0.14873
ASL ($\gamma_- = 0, \gamma_+ = 0$)	0.00029	0.15793
Hill ($\lambda = 1.5$)	0.0048	0.14867
sigmoidF1 (E=1 S=-30)	0.00046	0.0014
sigmoidF1 (E=1 S=-15)	0.00404	0.00106
sigmoidF1 (E=1 S=-1)	0.00015	0.05031
sigmoidF1 (E=0 S=-30)	0.00146	0.03082
sigmoidF1 (E=0 S=-15)	0.00047	0.03917
sigmoidF1 (E=0 S=-1)	0.00177	0.0439

For BCE models, we find that adding a class weight that is proportional to the normalized frequency in the dataset does not improve performance of the model. It's possible that these weights are not computed correctly, but ASL provides a dynamic weighting mechanism that is far more effective when the number of classes is large.

For our ASL models, we find that the best hyper-parameters are $\gamma_- = 0$ and $\gamma_+ = 0$ on the public leaderboard, but the default value of $\gamma_- = 4$ and $\gamma_+ = 1$ works just as well. This score performs better than the BCE model, despite claims that setting values of γ_- and γ_+ to zero would lead to a model that is equivalent to the BCE model. We note that if we had to choose hyperparameters for ASL through a validation set, it's possible that we could choose one that would be sub-optimal for the test set. We choose to use the default values for the rest of our experimentation.

The Hill loss performs between BCE and ASL in the non-linear model, and so we do not consider it further given the performance of ASL. The sigmoidF1 loss performs the worst out of all of the losses, despite the tuning in the ranges provided by the literature.

Table 3
Kaggle private and public leaderboard scores for torch-based models. The models are trained on the presence-absent dataset.

model	**private**	**public**
Geolocation ASL Top-k	0.16047	0.16128
Geolocation ASL Threshold	0.13395	0.13734
RGB-IR Top-k	0.16229	0.16108
RGB-IR Threshold	0.15217	0.15096
RGB-IR Landcover Top-k	0.01986	0.02096
RGB-IR Landcover Threshold	0.02545	0.0255
Time-series Top-k	0.10497	0.10392
Time-series Threshold	0.08444	0.0832
Tile2Vec RGB-IR Top-k	0.14071	0.13943
Tile2Vec RGB-IR Threshold	0.12989	0.12701
Tile2Vec RGB-IR Top-k	0.14448	0.13832
Tile2Vec RGB-IR Threshold	0.13318	0.12438

6.3. Tile CNN Models

We report minor improvements in the performance of the geolocation model, with our best model utilizing satellite imagery at 0.161 on the public leaderboard in table 3. However, training a model on the presence-absent dataset is difficult due to model convergence to a minima. This behavior is most prevalent in the efficientnetv2 backbone, where the larger parameter space and domain-specific pre-processing distortion leads to sub-optimal convergence.

The first 13 channels of the landcover raster to the RGB-NIR channels does not converge to a useful model. We find that our performance drops down to 0.02 when we try to utilize these features. Keeping the subset of features from landcover aids in a model that performs relatively well, but lacks large improvements over the RGB-NIR model.

The time-series model learns some structure despite the strange input representation with a score of 0.10 on the public leaderboard.

6.4. Tile2Vec Model

Tile2vec learns a useful representation that leads to smooth convergence of downstream classifiers. We observe convergence occurring within four epochs in our experimental setting, and the increase in the F1 metric for both validation and training sets increases monotonically on the classifier. In contrast, the models without the tile2vec backbone have validation F1 scores that fluctuate, typically within the first five epochs, and then decrease over time. The learned predictions are marginally less effective than learning the CNN model directly on the presence-absent dataset.

When we trained the model with ASL as part of the optimization objective, we observed that the triplet loss term no longer decreased monotonically over time. Instead, it sharply decreased to a minimum, increased, and decreased slowly over time. The behavior is likely due to the difference in magnitude of the triplet loss and the ASL loss. The triplet loss is normalized, while the ASL loss is not, so the ASL hyper-parameter dominates the gradient updates. This version of the model performs better on the transfer learning task to present-absent data than the triplet loss alone.

6.5. Competition Performance

Our best models are reported against the public leaderboard in table 4. The best score of the competition is 0.4089, while baseline models provided by the competition organizers lies around 0.25. Our models lie between the granular and coarse-grain frequency-based models.

Table 4
Kaggle private and public leaderboard scores with best models compared to baselines and top teams. Models included in the results are post-competition submissions.

Name	Public
Rank 1 - webmaking	0.4089
Rank 2 - AI2Lab	0.36837
Baseline with Landsat Cubes	0.26576
Baseline with Bioclimatic Cubes	0.2594
Baseline with Sentinel Images	0.23629
Top-25 species per district & biogeographical zones (PA)	0.20302
Top-25 species per Country/Region (PA)	0.17514
(Ours) Geolocation ASL Topk	0.16128
(Ours) RGB-IR Top-k	0.16108
(Ours) Tile2Vec RGB-IR Top-k	0.13943
(Ours) k-NN 50k	0.1251
Top-25 species in Presence Absence	0.11614
Top-25 species in Presence Only	0.08133

7. Discussion

We find difficulty overcoming basic baselines in the competition. In particular, frequency-based baseline submissions can be significantly more effective than the solutions proposed in our research of the problem. These solutions are done by predicting the top 25 species at varying levels of locality (e.g., globally or regionally) and by dataset. However, we find that latitude and longitude are surprisingly predictive of plant species in the dataset given an appropriate loss function. Using these geospatial features provides a useful diagnostic for more complex datasets, since the number of input features are small and are easier to debug. One possible limitation of our methodology is that we do not utilize the presence-only dataset with the exception of pre-training the tile2vec model.

7.1. Alternatives Methods

Learning a relationship between latitude and longitude to the species labels with classical machine learning techniques and off-the-shelf libraries is computationally intractable. We note alternative approaches that were explored but did not produce results for various reasons.

7.1.1. Classical Supervised Learning

Intead of using a neural network to learn a mapping from location to species, we tried learning the mapping via logistic regression. This numerically simple model can be learned using Spark via stochastic gradient descent (SGD). As a validation, we build a model to predict the 10 most frequent species in the dataset per site using only the location features. This achieves an F1-macro score of 0.09 when splitting the sites into a 90-10 train-validation split, which is better than random but roughly equivalent to always choosing the most frequent species.

We run into out-of-memory (OOM) issues when learning on 5 million rows and 10,358 species with scikit-learn or statsmodels. When we run the same procedure in Spark via distributed stochastic gradient descent (SGD), we find it will run for over 48 hours on a GCP n1-standard-8 instance (8 vCPU, 16GB RAM, 350GB NVME SSD) using 3-fold cross-validation (CV). We suspect this is due to the size of the coefficients involving J features and K output classes. Presuming an 8-byte double, the coefficients alone will be at least 8MB, larger than the typical CPU cache.

We investigate other algorithms for modeling multi-label classification, including Naive Bayes, SVM, Random Forests, and Factorization Machines. Naive Bayes assumes non-negative count data. SVMs are not tractable for our problem, and are slower to solve than linear/logistic regression for other problems in the Spark toolbox. Random Forests only support up to 100 classes in Spark, likely due to the branching factor to support each class. Factorization machines suffer a similar issue to logistic regression and SVMs. Our final attempt to model multi-class classification via classical supervised techniques is through XGBoost [13], which maintains a Spark binding. We run out of memory when trying to model many classes.

7.1.2. Low-rank Multilabel-Space Regression

Instead of trying to learn the mapping between features and label-space directly, it is possible to learn a relationship between features and a low-rank multi-label space instead [14]. It takes 30 minutes to learn a regression between location and a single binary response using either linear regression or XGBoost. Given these constraints, we would like to constrain our model to 4-8 response variables.

We try reducing the label-space via the DCT since the relationship is trivially invertible in the machine learning pipeline. We find that this is untenable since we need many more coefficients than are available in our budget to represent discontinuities in species presence.

Another approach is to use singular value decomposition (SVD) to compute a projection of label-space into the first few eigenvectors, and then learn the relationship between the features and the projection [15]. Then, predictions are quantized uses nearest neighbors in the projection of the label-space. This process is similar to latent semantic indexing (LSI), and would allow a model to take into consideration cooccurrences between labels. While interesting, this approach requires significant engineering effort for results that are no more interpretable than neural networks.

7.1.3. Node2Vec

Node2vec [16] learns to preserve properties of network nodes using biased random walks. Using the K-NN graph, we attempt embedding the survey sites using the co-occurrences of species among sites. We could then use the survey site embedding as a feature for the classification task. The survey node embedding is intractable due to the network size of 4 million nodes and 1 billion edges. A species node embedding can be computed in 20 minutes, which results in a vector representation of species that can be used for clustering or classification.

A survey node embedding would be useful as a feature for the classification task, since it would require no further processing to go from survey site to species. To take advantage of the species embeddings, we would need to compute some average of the embedding vectors before passing into a supervised classification model.

8. Future Work

We have explored various techniques for finding useful representations to model species distribution. One area for future work is to capture better nuance associated with the self-supervised representation learning of the tiles. We quickly reached a limit in how well the model could represent our training data, so it would be helpful to rigorously explore alternative model parameterizations and hyperparameters for the various loss functions. Additionally, it is unclear how best to incorporate the many raster layers provided in the competition. Ideally we would be able to determine which layers are most important to the multi-label classification task, possibly through extensive ablation testing of the features.

We would also like to continue down network or graph models of the survey and species. A rich interconnection exists between sites and species where sparse co-occurrences could be exploited through spatial locality. One way this could be done is by constructing node features through message passing of survey site nearest neighbors. Graph neural networks could also be an effective mechanism for generating embedding spaces by propagating information via diffusion. However, implementing these techniques could be challenging, especially since we failed to generate a survey embedding through the survey-species bipartite network due to computational constraints.

Our findings indicate a significant variation in the occurrence of the labels, with some labels with less than 10 data points and others with more than 10,000. Thus, this causes bias and imbalance in the training. A possible solution would be to bin the labels according to their frequency so that each label is relatively in the same range in terms of data points. This would also allow us to utilize XGBoost since it would reduce the number of classes that need to be classified.

It would also be interesting to implement a proper model of the dynamics of the remote sensing data. We can build manifold representation of satellite imagery, demonstrated by our experiments with tile2vec. It should be possible to model the linearized dynamics of a system by learning a Koopman operator that steps forward state space from one timestep to another. We hypothesize that this could be done by conditioning the tile embeddings on state evolutions, e.g., the 20 years of bio-climatic rasters and quarterly time series data. One potential way to do this is to learn a spatiotemporal embedding of the tiles via an explicit sequence model like an LSTM or transformer alongside methods to enforce the geographical distributional semantics afforded by tile embeddings. Another approach is to perform data-driven system identification to understand the dynamics of the bio-climatic rasters that have been embedded into the space and to understand the governing equations of the system with a method like SINDy [17].

9. Conclusions

In this study, we addressed the multi-label classification challenge of GeoLifeCLEF 2024, which aims to predict the presence or absence of plant species at specific locations based on spatial and temporal remote sensing data. We explored using a compressed version of the remote sensing data to train deep learning models, with varying levels of success. We take advantage of the geospatial nature of the data by building a neighborhood model with locality-sensitive hashing. Predictions from the neighborhood model perform better than some of the simplest frequency models made by the competition organizers. The neighborhood model is used as part of a self-supervised embedding model that learns a low-dimensional representation of the data that is effective for classification. Despite poor performance on the leaderboard, some of the ideas presented in this working note have potential for future work and have not been fully explored. Source code and models are available at https://github.com/dsgt-kaggle-clef/geolifeclef-2024.

Acknowledgements

Thank you to Professor Patricio Vela for supervising the project for Anthony Miyaguchi's ECE8903 Special Problems course at Georgia Tech. Thank you to the DS@GT CLEF group for access to cloud computing resources through Google Cloud Platform, and for a supportive environment for collaboration.

References

[1] L. Picek, C. Botella, M. Servajean, B. Deneu, D. Marcos Gonzalez, R. Palard, T. Larcher, C. Leblanc, J. Estopinan, P. Bonnet, A. Joly, Overview of GeoLifeCLEF 2024: Species presence prediction based on occurrence data and high-resolution remote sensing images, in: Working Notes of CLEF 2024 - Conference and Labs of the Evaluation Forum, 2024.

[2] A. Joly, L. Picek, S. Kahl, H. Goëau, V. Espitalier, C. Botella, B. Deneu, D. Marcos, J. Estopinan, C. Leblanc, T. Larcher, M. Šulc, M. Hrúz, M. Servajean, et al., Overview of lifeclef 2024: Challenges on species distribution prediction and identification, in: International Conference of the Cross-Language Evaluation Forum for European Languages, Springer, 2024.

[3] C. Botella, B. Deneu, D. M. Gonzalez, M. Servajean, T. Larcher, C. Leblanc, J. Estopinan, P. Bonnet, A. Joly, Overview of geolifeclef 2023: Species composition prediction with high spatial resolution at continental scale using remote sensing, in: Conference and Labs of the Evaluation Forum, 2023. URL: https://api.semanticscholar.org/CorpusID:264441401.

[4] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[5] H. Q. Ung, R. Kojima, S. Wada, Leverage samples with single positive labels to train cnn-based models for multi-label plant species prediction, in: Conference and Labs of the Evaluation Forum, 2023. URL: https://api.semanticscholar.org/CorpusID:264441458.

[6] A. Rouhani, D. Buchfuhrer, E. Bernhardsson, E. Freider, G. Poulin, D. Stadther, Honnix, U. Barbans, A. Krasnukhin, D. Whiting, J. Crobak, M. Rieger, B. Kiosidis, A. Nyman, F. Demaria, J. Kukul, R. Yon, R. Raposo, C. McGinty, B. Peksag, A. Brausewetter, R. Tavory, hirosassa, G. Balaraman, T. Engström, T. Grainger, M. Grey, M. Czygan, N. Arapé, B. Caswell, spotify/luigi, 2024. URL: https://github.com/spotify/luigi.

[7] M. Armbrust, R. S. Xin, C. Lian, Y. Huai, D. Liu, J. K. Bradley, X. Meng, T. Kaftan, M. J. Franklin, A. Ghodsi, et al., Spark sql: Relational data processing in spark, in: Proceedings of the 2015 ACM SIGMOD international conference on management of data, 2015, pp. 1383–1394.

[8] J. Leskovec, A. Rajaraman, J. D. Ullman, Mining of massive data sets, Cambridge university press, 2020.

[9] N. Jean, S. Wang, A. Samar, G. Azzari, D. Lobell, S. Ermon, Tile2vec: Unsupervised representation

learning for spatially distributed data, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, 2019, pp. 3967–3974.

[10] T. Ridnik, E. Ben-Baruch, N. Zamir, A. Noy, I. Friedman, M. Protter, L. Zelnik-Manor, Asymmetric loss for multi-label classification, in: Proceedings of the IEEE/CVF international conference on computer vision, 2021, pp. 82–91.

[11] Y. Zhang, Y. Cheng, X. Huang, F. Wen, R. Feng, Y. Li, Y. Guo, Simple and robust loss design for multi-label learning with missing labels, arXiv preprint arXiv:2112.07368 (2021).

[12] G. Bénédict, V. Koops, D. Odijk, M. de Rijke, Sigmoidf1: A smooth f1 score surrogate loss for multilabel classification, arXiv preprint arXiv:2108.10566 (2021).

[13] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, 2016, pp. 785–794.

[14] A. Dasgupta, S. Katyan, S. Das, P. Kumar, Review of extreme multilabel classification, arXiv preprint arXiv:2302.05971 (2023).

[15] F. Tai, H.-T. Lin, Multilabel classification with principal label space transformation, Neural Computation 24 (2012) 2508–2542.

[16] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, 2016, pp. 855–864.

[17] S. L. Brunton, J. L. Proctor, J. N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proceedings of the national academy of sciences 113 (2016) 3932–3937.