Time Series Tokenization Methods for Species Distribution Modelling with Deep Learning

Semester Project Report
8 ECTS

Julie Charlet

ECEO

Under the supervision of
Robin Zbinden and Devis Tuia

January 10, 2025

1 Abstract

This project investigates the role of time series data in species distribution modeling, utilizing environmental features from the *GeoPlant* dataset to predict plant species occurrences across Europe. The study incorporates four climatic and six satellite time series, spanning 18 years with monthly to quarterly resolution, as a new data modality alongside tabular data. Four *Time Series Tokenizer* methods — multilayer perceptron, fully convolutional network, ResNet, and Transformer Encoder — are evaluated for their ability to represent time series data efficiently.
Results show that adding satellite time series to tabular data increases the test AUC by 2.34%, likely due to the absence of satellite-derived information in the tabular dataset. Furthermore, in the absence of tabular data, time series data — both satellite and climatic — outperform tabular predictors, highlighting the limitations of pre-aggregated tabular indices. Among the evaluated methods, the Transformer Encoder demonstrates the best performance, effectively capturing a large context size while preserving the temporal structure of input data through positional encodings. This study emphasizes the importance of high-quality time series data in species distribution modelling and demonstrates the potential of Transformer-based architectures for time series embedding.

Contents

2 Introduction

In the context of multi-modal deep learning, this semester project aims to assess the role of time series in species distribution modelling alongside existing tabular data, using the *GeoPlant* dataset [1]. Additionally, the project seeks to identify an appropriate method for tokenizing time series to ensure compatibility with the `FTTransformer` architecture. The primary research question of this project is:

– What is the importance of time series in species distribution modelling?

Our current hypothesis is that the temporal aspect of variables expressed as time series may provide valuable information to the model. Temporal context, such as inter- and intra-annual variability or knowledge of past extreme events (e.g., heatwaves or droughts), can significantly influence the distribution of certain plant species. The second research question focuses on identifying the most suitable method for working with time series:

– How can we tokenize time series in a meaningful manner?

The main challenge here is to identify a method that effectively represents and captures the temporal structure of time series, which distinguishes them from other types of data. In this study, we leverage the supervised problem context to test and compare four different methods. This challenge is not only relevant to species distribution models and ecology, where time series data are common, but also to a wide range of fields such as medical research and financial mathematics, where time series form an integral part of the problems at hand.

2.1 Species Distribution Modelling

Species distribution modelling (SDM) is a key tool in ecology and conservation, aiming to predict the realized niche of species based on environmental variables [2]. These variables typically include climatic and topographic data [3]. SDM is widely applied in conservation efforts, invasive species management, and climate change adaptation. However, accurately capturing the complex interactions between species, particularly in multi-species contexts, remains a significant challenge. By integrating large datasets on species presence or absence with environmental variables, deep learning algorithms provide a promising framework for modelling these interactions and predicting species distributions using environmental predictors [4].

2.2 Related Work

A crucial step in multi-modal deep learning is identifying an effective representation of the data for input into a common network. Deep learning tasks involving time series are diverse, as they are used for prediction and classification in both supervised and unsupervised contexts alongside various data modalities. As a result, time series embedding techniques are varied and often combine different methods specifically tailored to particular applications [5], [6], [7].

Historically, time series modelling primarily relied on statistical methods, but the increased amount of available and the advances in machine learning has driven the adoption of approaches such as *recurrent neural networks (RNNs)* [8] and *1-dimensional convolutional neural networks (CNNs)* [9], [10] for solving the challenge of capturing temporal dynamics. Notably, [11] compares three algorithms — a multi-layer perceptron, a fully connected network, and a ResNet—for classification tasks, evaluating them across 44 time series datasets. Although the results are highly dataset-dependent, the fully connected network (FCN) emerges as a competitive model choice.

More recently, the powerful *Transformer-Encoder* architecture, originally popularized by [12], has been adapted for time series tasks, demonstrating its potential in this domain [13]. *Transformers* leverage

the self-attention mechanism, where each token "attends" to all others in the sequence. To account for the sequential nature of the data, a *positional encoding* is added to each token, enabling the model to infer its context based on its relative position in the input. These *positional encodings* must uniquely represent each position and can be derived from sine and cosine functions [12].

3 Methods and Material

3.1 Dataset

The dataset used in this project is the *Geoplant* dataset [1] which consists of presence-absence (PA) survey data of plants in Europe, collected from several datasets hosted by the European Vegetation Archive (EVA, e.g., Denmark Naturdata, IGN National Forest Inventory, Belgium INBOVEG). PA data has been collected by experienced biologists given small observation plots (between $10m^2$ and $400m^2$) whose location coordinates are known [1]. For each given spatial plot and survey date, diverse environmental features are provided in the corresponding dataset hosted on *Kaggle* [14]. Those features include environmental tabular data, satellite images, and climatic and satellite time series. Environmental tabular data was extracted from several raster type presented in the following subsubsection 3.1.1. Available time series are presented in more details in subsubsection 3.1.2.

3.1.1 Environmental Rasters

For each data sample, five types of environmental values, extracted from the corresponding rasters, are provided. For each sample, the value of the pixel at the observation's location is taken as the value for the feature extraction from the raster. The 5 types of rasters are the following:

- Bioclimatic rasters, 19 variables
- Soil rasters, 9 variables
- Elevation, 1 variable
- Land-cover, 1 variable
- Human footprint, 16 variables

Bioclimatic rasters: Comprehensive temperature and precipitation summary statistics from *CHELSA climate* [15], [16].

Soil rasters: Soil composition information, such as clay, nitrogen or sand content or soil pH value from [17].

Elevation: Elevation in meter from [18].

Land-cover: Landcover class prediction from [19].

Human footprint: Quantification of various human impacts, such as lights, roads or population density provided for 1994 and 2009. This data comes from [20].

All these values are concatenated with the latitude and longitude for each survey, forming what will be referred to as the *tabular data* throughout the remainder of the report.

3.1.2 Satellite and Climatic Time Series

In addition to the environmental variables, time series data is available for each sample. The climatic time series includes four variables: monthly minimum temperature (t_{min}), monthly maximum temperature (t_{max}), monthly average temperature (t_{mean}), and monthly average precipitation amount, spanning from January 2000 to December 2018. However, due to a high proportion of missing data between 2016 and 2018, the last three years are excluded. Similar to the climatic tabular values, this data is sourced from *CHELSA climate* [15]. Overall each variable contains 228 timestamps.

The satellite time serie data consists of 6 channels: *Red, Blue, Green, NIR, SWIR1* and *SWIR2*. It was obtained through the Landsat ARD program and pre-processed by EcoDataCube. The data is trimestrial, hence has four values per year. Overall 72 timestamps are available.

3.1.3 Dataset Split

The dataset is divided into training, validation, and test sets using a split of 70%, 15%, and 15%, respectively. The resulting sizes of each subset are 62'291 and 13'348 samples. The validation set is employed for tuning the hyperparameters of the models, while the test set is reserved for evaluating performance on previously unseen data and assessing the model's generalization capabilities.

Because our data is spatially structured, dataset splitting must be done with spatial blocking to avoid data leakage between the datasets [21]. Ignoring the spatial structure of the data can lead to over optimistic performances as the model can revisit very close locations across the training, validation and testing sets. To perform the spatial blocking, the dataset is split using `verde.train_test_split` function with the argument `spacing` equal to 1° [22]. The influence of the `spacing` parameter can be seen in Figure 1.

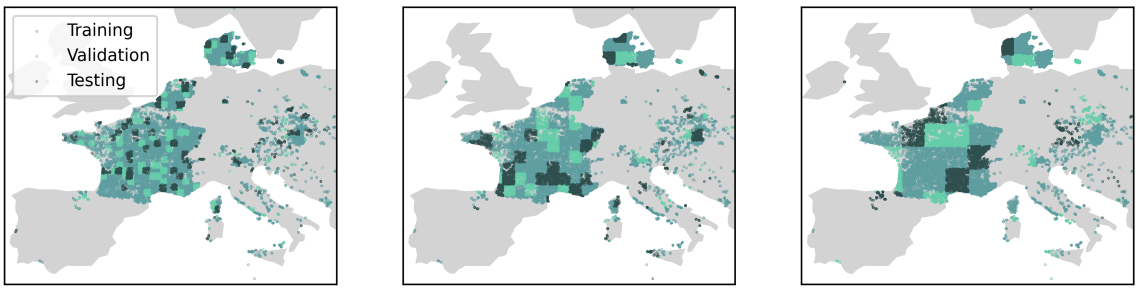

Figure 1: Train, validation and test split with spatial blocking. (a) `spacing` = 0.5°, (b) `spacing` = 1°, (c) `spacing` = 2°

3.2 MaskSDM

To handle missing values during training and inference, we use the MaskSDM approach introduced by [23]. By randomly masking tokens during the training phase, the network can handle missing features at inference time. Additionally, this gives the opportunity to the user to only consider the features deemed relevant for the application.

During training, a boolean mask is applied to the data. It is right away zero if the feature is missing. Additionnaly, a binary tensor is generated by sampling from a Bernoulli distribution, where the probability of success is uniformly set to a fixed probability. The tensor's dimensions match the dimension of the input data, producing a mask for stochastic operations. The final mask is taken as the intersection of both tensors.

During inference, masking specific features enables the selection of features deemed relevant for prediction. In this study, it allows the evaluation of model performance by selecting specific features during testing and assessing their influence on overall performance.

3.3 Overall Method Structure and *FTTransformer* Architecture

Figure 2 illustrates the overall structure of the method. Each data modality is initially transformed into embedding vectors of size $d = 192$. When masking is applied, it is performed separately for

each data modality. The resulting embeddings are then concatenated into a single tensor, which is subsequently fed into the `FTTransformer`.

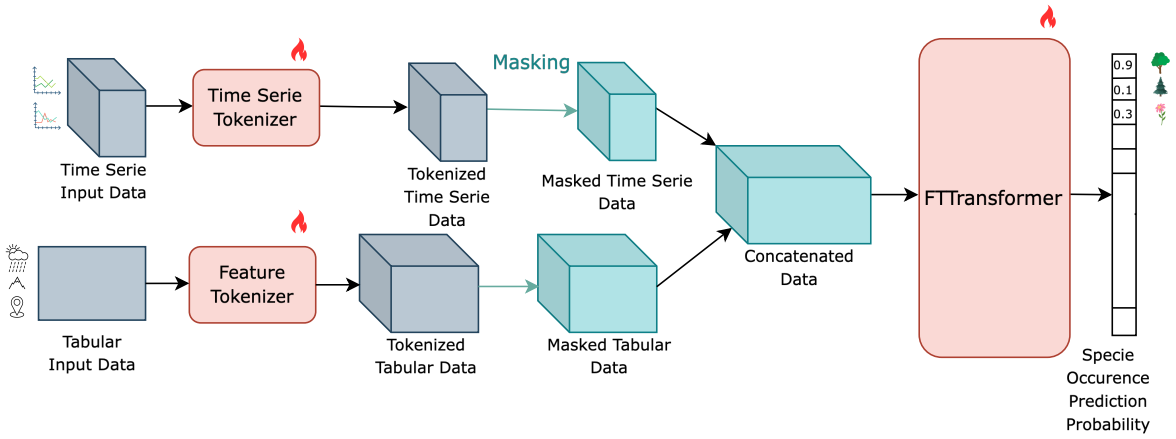

Figure 2: Overall Method Structure

During inference, the `FTTransformer` takes as input the tabular and time series data for a specific location, potentially including masked missing entries. The model outputs a vector representing the probability of occurrence for each species in the training dataset.

The `FTTransformer` (*Feature Tokenizer Transformer*), as introduced in [23], utilises the tokenized form of multi-modal data to predict the probability of species occurrence for each patch. Its implementation is based on the approach outlined in [12]. Details regarding the hyperparameters used for training are provided in Appendix A.

3.4 *Time Series Tokenizer* Methods

This project involves multimodal training, utilizing two modalities: tabular and time series data. Tabular data is already tokenized using the `Feature Tokenizer`, which is implemented as a simple linear layer with a bias term, mapping each tabular feature to a token of size $d = 192$. To incorporate additional data modalities, new *Time Series Tokenizer* methods are proposed to transform each time series into tokens compatible with the `FTTransformer` architecture.

Based on the hypothesis that significant information can be derived from the temporal variation of the data, the primary challenge of this tokenization process lies in effectively representing its temporal characteristics. Moreover, following the principles of the MaskSDM approach, the tokenization of each channel must operate independently, ensuring that each channel has distinct weights and can be removed without affecting the others. This independence is crucial as the dataset consists of 10 channels, with 6 corresponding to satellite time series and 4 to climatic time series. Furthermore, the satellite and climatic time series differ in length, necessitating that the tokenization method accommodates variations in input lengths seamlessly. To address these constraints, four architectures are proposed for tokenizing the time series.

1. Multilayer Perceptron (MLP)
2. Fully Convolutional Network (FCN)
3. Residual Network (ResNet)
4. TransformerEncoder (TE)

Architectures 1–3 are reproduced from the methodology presented in [11], while the Transformer Encoder architecture is developed specifically for this project. All architectures are described and justified in greater detail in the following sections. Additional hyperparameters for the different *Time Series Tokenizers* methods are presented in the Appendix A.

3.4.1 Multilayer Perceptron

Although MLPs do not explicitly leverage the temporal structure of the input data, each internal node attends to all input entries. The rationale for employing an MLP lies in the hypothesis that it may implicitly learn to capture temporal dependencies, thereby enabling a meaningful representation of the time series data.

The architecture is adapted from the work of [11]. The network consists of three blocks, each comprising a dropout layer to enhance generalization capabilities, a linear layer with 500 neurons, and a nonlinearity implemented as a rectified linear unit (ReLU). A fourth block is included, consisting of a linear layer with $d = 192$ neurons, to generate the final token. This transformation is applied independently to each channel of the time series data, with no weight sharing between channels.

3.4.2 Fully Convolutional Network

Convolutions are widely recognized for their ability to exploit spatial correlations in two-dimensional image data. For one-dimensional time series data, the hypothesis is that convolutions with an appropriately defined receptive field can effectively capture meaningful attributes, such as seasonal variations. Motivated by this premise, the FCN architecture described in [11] has been adapted for use as a second *tokenizer* method.

This network consists of three blocks, each comprising a one-dimensional convolution layer, followed by a batch normalization layer and a nonlinearity (ReLU). The kernel sizes of the convolution layers are $\{8, 5, 3\}$, and the number of output channels for the convolution layers in the three blocks is $\{128, 256, 128\}$, respectively. Padding is applied to the convolution layers to maintain consistent data length throughout the network. To produce the final token, the output from the third block is collapsed along the time axis using `torch.nn.AdaptiveAvgPool1d` [24] and subsequently passed through a linear layer with $d = 192$ neurons.

3.4.3 Residual Network

Residual networks utilize residual connections to create shortcuts within each block of CNNs. Due to their depth, these architectures excel in vision-related tasks and are believed to effectively learn complex relationships within data. This capability to capture complex long-term dependencies motivates the use of residual networks for representing temporal dependencies in time series data.

The architecture, adapted from [11], consists of three residual blocks. Each residual block contains three convolutional blocks, followed by the addition of the residual connection and a nonlinearity (ReLU). Each convolutional block is constructed similarly to the FCN and comprises a one-dimensional convolution layer, a batch normalization layer, and a nonlinearity (ReLU). The kernel sizes of the convolution layers are $\{8, 5, 3\}$, with padding applied to preserve the input length. The number of output channels for the convolution layers across the three blocks is $\{64, 128, 128\}$. The final token is generated as in the FCN by collapsing the output of the third block along the time dimension using `torch.nn.AdaptiveAvgPool1d` [24] and passing it through a linear layer to produce a token of size $d = 192$.

3.4.4 Transformer Encoder

Transformer architectures, introduced by [12], have recently proven their capabilities in a variety of deep learning applications. Recently, they have been successfully used in time series representation learning [13]. As for multilayer perceptrons, the attention mechanism can attend to all tokens of the input data. Additionally, temporality in the input data is explicitly represented by the addition of a positional encoding to each input token. Positional encodings are a key component of Transformers ability to understand the context of a token by providing information about its relative position in the input data [25].

The architecture implemented is inspired from the Encoder architecture presented in [12]. The present implementation processes each channel independently, i.e. weights are not shared between different time serie channels at any time during the tokenization process.

Timestamp values are initially projected through a linear layer to create a token of dimension $d_{\text{feature}} = 16$. A positional encoding, generated using sine and cosine functions as described in [12], is then added to the token. The input data is subsequently normalized using layer normalization and passed through a self-attention mechanism with 4 attention heads. A residual connection is added before applying another layer normalization. Finally, the data is processed through a feed-forward network, followed by the addition of another residual connection. Those steps are repeated $n_{blocks} = 2$ times before the output of the block is projected through a linear layer of dimension $d = 192$ to produce the final token.

4 Results

4.1 Performance Improvements with Time Series

To assess the contribution of each time series to the model's performance, the model is evaluated using only the tabular variables initially, followed by the addition of one type of time series at a time. This approach quantifies the performance improvement resulting from the inclusion of each time series and identifies which type contributes the most. Those results are generated using the Transformer Encoder architecture presented in section 3. The choice of this architecture will be justified in subsection 4.3.

For improved interpretability, the time series are grouped into five categories. For satellite data, the *red*, *green*, and *blue* channels form the **RGB** group. **NIR** is treated as a separate category, while **SWIR** encompasses both the *SWIR 1* and *SWIR 2* channels. For climatic time series, temperature-related data are grouped under **Temperature**, and the **Precipitation** series is treated as a standalone category.

Feature	Test AUC
RGB	97.1
NIR	96.9
SWIR	97.0
Temperature	95.6
Precipitation	95.5
Baseline	94.7

Table 1: Test AUCs when adding Time Series Features to Tabular Data for the Tansformer Encoder *Time Series Tokenizer* method. The baseline consists of tabular features only.

Table 1 presents the Test AUC (Area Under the Receiver Operating Characteristic (ROC) Curve) results when adding specific types of time series to the model. For reference, the performance using only tabular data is 94.7%. The category that provides the greatest improvement is the RGB satellite time series, increasing the AUC by 2.34% to 97.1%. The other satellite categories also yield significant gains, with AUCs of 97.0% for SWIR and 96.9% for NIR. Climatic time series contribute less to the performance improvement, with the model achieving an AUC of 95.6% when including temperature time series and 95.5% with precipitation time series.

4.2 Time Series Influence on Prediction

To gain a more qualitative understanding of the influence of time series on model predictions, Figure 3 visualizes the differences in model predictions for a specific species under three scenarios: (a) using only tabular data, (b) using tabular data and satellite time series, and (c) using tabular data and climatic time series. Panel (d) provides a comparison with the actual distribution of the species. Again, those predictions are made using the Transformer Encoder as *Time Series Tokenizer* method. This choice is justified in subsection 4.3.

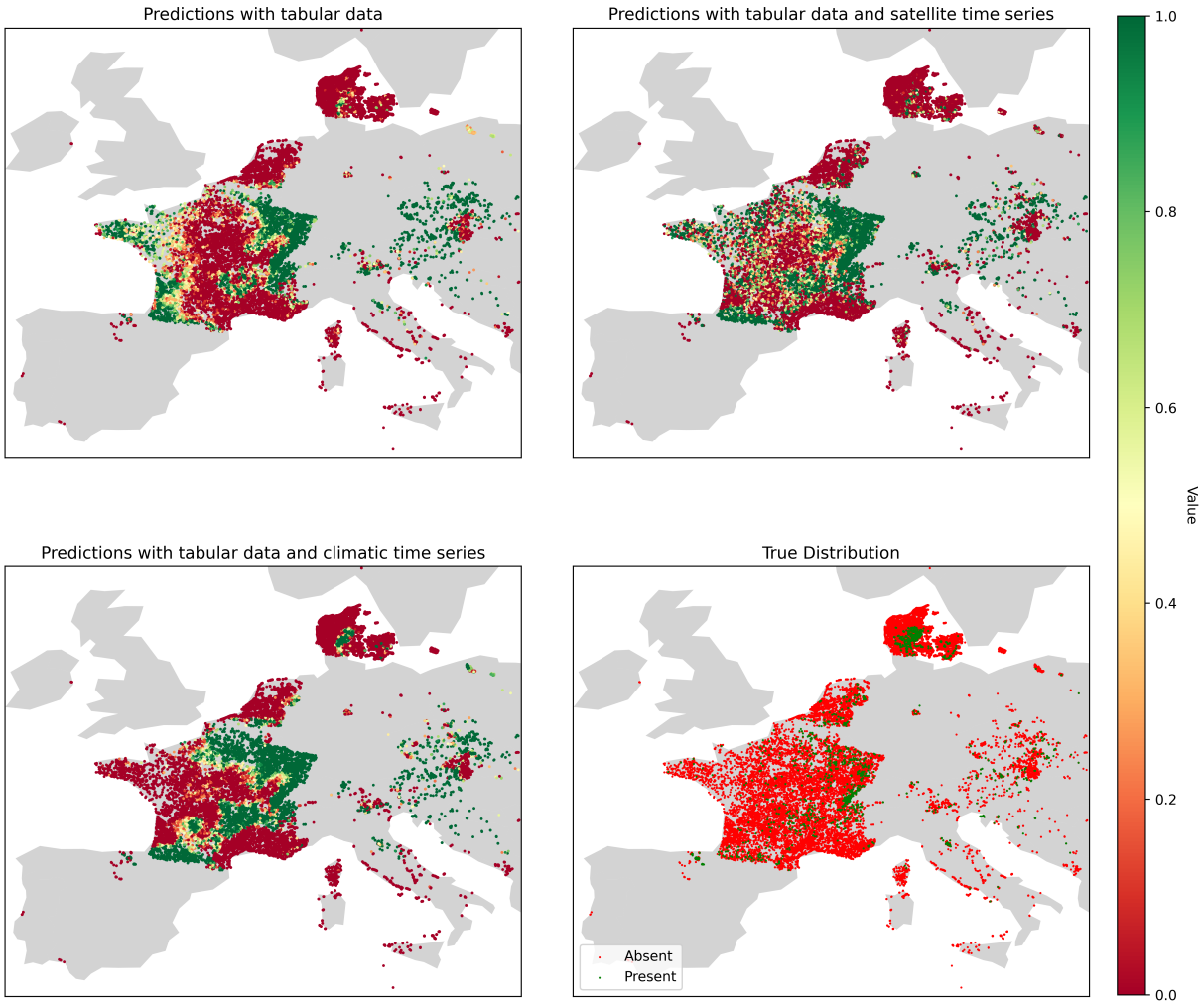

Figure 3: Occurence prediction for specie n° 9028 based on (a) tabular features, (b) tabular features and satellite time series, (c) tabular features and climatic time series and (d) the true (observed) distribution

4.3 *Time Series Tokenizer* Methods Performances

All *Time Series Tokenizer* methods presented in subsection 3.4 are trained and evaluated on the test set to determine which architecture best represents the time series data. The models are assessed under the following conditions: (1) tabular data only, (2) tabular data with satellite time series, (3) tabular data with climatic time series, (4) satellite time series only, (5) climatic time series only, (6) satellite and climatic time series, and (7) tabular data combined with all time series (all data).
The first column of Table 2 presents the number of trainable parameters for all *Time Series Tokenizer* methods.

The ResNet architecture is the largest model, with 10'802'545 trainable parameters, resulting from its deep structure. The MLP is the second largest, with 9'709'385 parameters. The FCN has 5'953'905 trainable parameters, while the Transformer Encoder is the smallest, with 3'475'596 parameters.

The remaining columns of Table 2 presents the AUC performance of various *Time Series Tokenizer* methods, with the highest value in each column highlighted in bold. Overall, the FCN and Transformer Encoder consistently achieve the best performance across the range of predictors.

All models perform better when predicting from time series alone (6) than from tabular data. The Transformer Encoder shows the largest performance difference, achieving 97.2% with time series com-

	# of parameters	Tabular (1)	Tabular + TS Sat (2)	Tabular + TS Clim (3)	TS Sat (4)	TS Clim (5)	TS Sat + Clim (6)	Tabular + TS Sat + Clim (7)
MLP	9'709'385	95.2	97.3	95.5	91.5	94.3	97.1	97.5
FCN	5'953'905	**95.7**	**97.3**	**95.8**	88.2	94.7	97.0	97.4
ResNet	10'802'545	95.0	97.0	95.5	88.1	94.7	97.0	97.2
Transformer Encoder	3'475'569	94.7	97.1	95.6	**91.8**	**94.9**	**97.2**	**97.6**

Table 2: *Time Series Tokenizer* methods number of trainable parameters and performance comparison

pared to 94.7% with tabular data. Generally, climatic time series are more predictive than satellite time series. Using satellite time series alone (6) results in the lowest performance, while using only climatic time series (5) achieves performance comparable to tabular data (1), slightly better for the Transformer Encoder and slightly worse for the other methods.

However, as noted for the Transformer Encoder in subsection 4.1, adding satellite time series to tabular data (2) improves performance more than adding climatic time series (3). This trend extends to other *Time Series Tokenizer* methods, with column (2) showing performance very close to predictions made using all data (7).

The Transformer Encoder exhibits the best overall performance and demonstrates the highest capacity to extract predictive information from time series (4, 5, 6). Consequently, it will be used for the subsequent analyses presented in subsection 4.1 and subsection 4.2.

5 Discussion

5.1 Performance Improvements with the Inclusion of Time Series Data

Best Overall Performance

Table 1 demonstrates that augmenting the tabular data with time series improves model performance. Table 2 shows that combining both types of time series (satellite and climatic) yields the best results. This suggests that, in this dataset, time series contain valuable information that enhances prediction capabilities beyond what is captured by the tabular data alone. However, if this improvement is attributable to the temporal dimension of the time series data remains uncertain. Notably, the MLP Time Series Tokenizer, which does not explicitly account for the order of the data, achieves performance comparable to or better than the FCN and ResNet when using either or both types of time series (4, 5, 6) or all three data modalities (7). This suggests that the content of the time series may be more important than their temporal structure.

Another point is the varying performance achieved by the four methods when using only tabular data (1). Discrepancies in performance should not be influenced by the time series tokenization method, as the tokenization of the tabular variable is done separately.

Effects of adding Time Series to Tabular Data

As shown in both Table 1 and Table 2, adding satellite time series to tabular data results in a greater performance improvement compared to adding climatic time series. This is likely because complete climatic data is already included in the tabular data, which contains 19 bioclimatic raster variables describing temperature and precipitation patterns. These variables include metrics such as annual mean temperature, mean temperature for the coldest and warmest months and quarters, temperature seasonality, and mean diurnal range, providing a comprehensive description of temperature distribution. Consequently, the monthly mean, minimum, and maximum temperature time series may contribute little additional information, as their resolution is comparable to the aggregated statistics already included.

However, since the time series span 19 years, they could provide valuable insights into the climate of a given location in the past decade. This historical perspective may be relevant in regions experiencing rapid climate change, where certain species may partially adapt to new conditions, leading to their presence in atypical environments. Additionally, this may also apply to ecosystems where non-yearly recurring events, such as wildfires, significantly influence the ecosystem's dynamics.

In summary, even though the added value brought by the inclusion of the climatic time series remains limited in this context, it could be valuable if the temporal resolution was higher than of the aggregated data or in scenarios where historical records have a more pronounced impact on current flora.

In contrast, satellite data, whose inclusion alongside tabular data yields better performance, is entirely absent from the tabular data. This suggests that satellite data is predictive for species distribution in this dataset. As shown in Table 1, the most predictive category is the RGB category (97.7% AUC), which covers the visible light spectrum. These channels capture larger structures, such as vegetation or human-made constructions, which can help the model predict species occurrences.

The short-wave infrared (SWIR) channels are also highly predictive, with an AUC of 97.0%. SWIR images are often used for material identification, as they capture geological features [26]. Additionally, SWIR can penetrate thin cloud cover, unlike RGB channels, which are less effective in adverse weather conditions. Finally, the inclusion of near-infrared (NIR) channels alongside tabular data achieves a similar performance improvement (96.9% AUC). NIR images provide information about plant and soil properties and are commonly used for atmospheric composition analysis.

Lastly, better performance is achieved when all three satellite channel categories are included, resulting in an AUC of 97.6%. This highlights the importance of all six satellite channels, indicating that they provide complementary information for predicting species distribution.

Tabular vs Time Series Data

The tabular data comprises 42 tokens, while the time series data consists of 10 tokens (4 from climatic and 6 from satellite time series). Despite this difference, Table 2 demonstrates that using only time series data results in better predictive performance compared to using only tabular data. This effect is most pronounced for the Transformer Encoder (97.6% vs. 94.7%) but is consistent across all four *Time Series Tokenizer* methods.

This result indicates that certain tabular variables, such as the human footprint or landcover indices, are either not highly predictive for species occurrence or are better represented by satellite data. These tabular variables consist of aggregated indicator values intended to describe human footprint and landcover. However, their derivation from multiple indicators results in the loss of specificity and spatial accuracy, reducing their effectiveness as predictive features.

In contrast, satellite imagery may inherently contain the necessary information to represent these components, provided the model has a sufficiently expressive neural network. Unlike aggregated tabular indices, satellite data avoids unnecessary preprocessing steps that degrade data quality and introduce noise, thereby offering a more accurate representation for prediction.

5.2 Influence of Time Series on Prediction

Figure 3 displays the predicted species occurrence for a selected species using (a) tabular data, (b) tabular and satellite time series, and (c) tabular and climatic predictors, with the true species distribution shown in panel (d). The figure illustrates how predictions vary depending on the predictors used, offering insight into the impact of each data type influence on the model's outputs.

In panel (c), the inclusion of climatic time series reduces the predicted occurrence probability along the western coast of France compared to panel (a). This adjustment better reflects the true distribution, where this species is less prevalent in that region. Conversely, in panel (b), the addition of satellite time series increases the predicted occurrence probability in northern and central France. This change aligns with the true distribution shown in panel (d), which records observations of this species in those areas. These results suggest that the time series data provide additional context, which enhances the model's predictions.

This case demonstrates that tabular, satellite, and climatic data emphasize different aspects of species distribution. It highlights the qualitative influence of each data type on the final prediction and underscores the importance of incorporating time series into the model.

5.3 Comparison of *Time Series Tokenizer* Methods

Among the four *Time Series Tokenizer* methods tested in this project, Table 2 indicates that the Transformer Encoder is the most effective at representing time series data. However, the FCN also demonstrates strong performance, with results that are only slightly lower than those of the Transformer Encoder. This results is in accordance with those from [11], whose FCN demonstrated better performance than the MLP and ResNet networks for most datasets.

In addition, both models have significantly fewer trainable parameters compared to the MLP and ResNet (Table 2). This suggests that larger architectures may not necessarily be better suited to capture the information present in these time series. However, the role of hyperparameter tuning should be considered when comparing different architectures, as not all models may have been trained with optimally selected hyperparameters. The tested hyperparameters and their corresponding performance on the validation set are provided in Appendix A. Further hyperparameter tuning is required to validate the current results and is left for future work.

The relatively low resolution of the time series — monthly for climatic data and quarterly for satellite data — might limit the need for larger architectures. With higher-resolution time series, a larger model might be required to effectively capture the increased complexity of the data.

The Transformer, however, has a key advantage over convolution-based networks (FCN and ResNet) in its ability to consider the entire sequence context simultaneously. This capability makes Transformers better suited for capturing long-term dependencies, which convolution-based networks are less capable of addressing.

Unlike MLPs, Transformers incorporate temporal context and the order of the data through the addition of positional encodings to each input token. This positional information likely contributes to the superior performance of the Transformer Encoder compared to the MLP, despite having approximately 60% fewer trainable parameters. Furthermore, the MLP may be more prone to overfitting, which can reduce its generalization capabilities and lower its test AUC performance.

In conclusion, the Transformer Encoder architecture emerges as a promising approach for time series encoding, effectively balancing a large context window with the preservation of the temporal structure of the input data.

6 Conclusion

This project investigated the importance of the inclusion of time series in species distribution modelling. The performed experiments demonstrated the valuable addition of information by time series to the deep learning model. In particular, when tabular data is already utilized, the addition of satellite time series results in significant performance improvements. This is likely because satellite data is entirely absent from the tabular dataset. Consequently, incorporating *GeoPlant*'s satellite data as a third data modality for this project could be highly relevant. Comparing performance in this context would help determine whether the observed improvements are due to the temporal structure of the satellite data or simply the information it contains.

Additionally, time series data alone outperform tabular data when used independently. This is likely because tabular indices, such as those for human footprint or landcover, are derived from aggregated data of lower quality. This underscores the importance of high-quality raw data in deep learning, as models often generate better feature representations than those provided by pre-aggregated indices. However, tabular data and time series affect the predicted occurrence probabilities in different ways, and the highest performance is achieved when all available data modalities are combined.

Finally, among the four *Time Series Tokenizer* methods, the Transformer Encoder demonstrates the best performance. This architecture shows great promise in time series embedding, as it effectively captures a large context size while preserving the temporal structure of the input data through the addition of positional encodings.

Data Availability The code for this experiments is available on GitHub.

Generative AI Generative AI has been used to improve the syntax and the formulation of this report.

References

[1] L. Picek, C. Botella, M. Servajean, C. Leblanc, R. Palard, T. Larcher, B. Deneu, D. Marcos, P. Bonnet, and A. Joly, "Geoplant: Spatial plant species prediction dataset," 2024. [Online]. Available: https://arxiv.org/abs/2408.13928

[2] C. R. Townsend, *Ecological applications towards a sustainable world*. Blackwell publishing, 2008.

[3] A. Farashi and M. Alizadeh-Noughani, *Basic Introduction to Species Distribution Modelling*. Singapore: Springer Nature Singapore, 2023, pp. 21–40. [Online]. Available: https://doi.org/10.1007/978-981-99-0131-9_2

[4] S. Beery, E. Cole, J. Parker, P. Perona, and K. Winner, "Species distribution modeling for machine learning practitioners: A review," 2021. [Online]. Available: https://arxiv.org/abs/2107.10400

[5] H. Xia, X. Chen, Z. Wang, X. Chen, and F. Dong, "A multi-modal deep-learning air quality prediction method based on multi-station time-series data and remote-sensing images: Case study of beijing and tianjin," *Entropy*, vol. 26, no. 1, 2024. [Online]. Available: https://www.mdpi.com/1099-4300/26/1/91

[6] S. Talukder, Y. Yue, and G. Gkioxari, "Totem: Tokenized time series embeddings for general time series analysis," 2025. [Online]. Available: https://arxiv.org/abs/2402.16412

[7] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin, "Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting," 2022. [Online]. Available: https://arxiv.org/abs/2201.12740

[8] F. Karim, S. Majumdar, H. Darabi, and S. Harford, "Multivariate lstm-fcns for time series classification," *Neural Networks*, vol. 116, pp. 237–245, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608019301200

[9] J. Franceschi, A. Dieuleveut, and M. Jaggi, "Unsupervised scalable representation learning for multivariate time series," *CoRR*, vol. abs/1901.10738, 2019. [Online]. Available: http://arxiv.org/abs/1901.10738

[10] A. Fraikin, A. Bennetot, and S. Allassonnière, "T-rep: Representation learning for time series using time-embeddings," 2024. [Online]. Available: https://arxiv.org/abs/2310.04486

[11] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," *CoRR*, vol. abs/1611.06455, 2016. [Online]. Available: http://arxiv.org/abs/1611.06455

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023. [Online]. Available: https://arxiv.org/abs/1706.03762

[13] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, and C. Eickhoff, "A transformer-based framework for multivariate time series representation learning," *CoRR*, vol. abs/2010.02803, 2020. [Online]. Available: https://arxiv.org/abs/2010.02803

[14] M. C. Lucas Picek, Christophe Botella and A. Joly, "Geoplant: Spatial plant species prediction dataset," 2024. [Online]. Available: https://www.kaggle.com/datasets/picekl/geoplant/data

[15] C. Climate, "Chelsa bioclim," 2025, accessed: 2025-01-04. [Online]. Available: https://chelsa-climate.org/bioclim/

[16] D. N. Karger, O. Conrad, J. Böhner, T. Kawohl, H. Kreft, R. W. Soria-Auza, N. Zimmermann, P. Linder, and M. Kessler, "Climatologies at high resolution for the earth's land surface areas," *Scientific Data*, vol. 4, 09 2017.

[17] I. . S. Information, "Soilgrids," 2025, accessed: 2025-01-04. [Online]. Available: https://soilgrids.org/

[18] N. Aeronautics and S. I. L. C. S. Space Administration (NASA), "Aster gdem version 3," 2025, accessed: 2025-01-04. [Online]. Available: https://lpdaac.usgs.gov/products/astgtmv003/

[19] D. S.-M. Mark Friedl, "Mcd12q1 modis/terra+aqua land cover type yearly l3 global 500m sin grid," 2015, accessed: 2025-01-04. [Online]. Available: http://doi.org/10.5067/MODIS/MCD12Q1.006

[20] O. Venter, E. Sanderson, A. Magrach, J. Allan, J. Beher, K. Jones, H. Possingham, W. Laurance, P. Wood, B. Fekete, M. Levy, and J. Watson, "Global terrestrial human footprint maps for 1993 and 2009," *Scientific data*, vol. 3, p. 160067, 08 2016.

[21] D. Roberts, V. Bahn, S. Ciuti, M. Boyce, J. Elith, G. Guillera-Arroita, S. Hauenstein, J. Lahoz-Monfort, B. Schröder, W. Thuiller, D. Warton, B. Wintle, F. Hartig, and C. Dormann, "Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure," *Ecography*, vol. 40, 12 2016.

[22] T. V. Developers, "verde v1.8.1," https://www.fatiando.org/verde/latest/index.html, 2024, accessed: 2025-01-04.

[23] G. S. B. K. Robin Zbinden, Nina van Tiel and D. Tuia, "Masksdm: Adaptive species distribution modeling through data masking," 2024.

[24] T. P. Foundation, "torch v2.4.2," https://pytorch.org/, 2024, accessed: 2025-01-04.

[25] H. Phillips, "Positional encoding, medium," 2023, accessed: 2024-12-03. [Online]. Available: https://medium.com/@hunter-j-phillips/positional-encoding-7a93db4109e6

[26] E. S. Imaging, "Short-wave infrared satellite imagery," accessed: 2025-01-10. [Online]. Available: https://www.euspaceimaging.com/swir/

A Appendix

***FTTransformers* and Training Hyperparameters**

Hyperparameter	Value
Training epochs	150
Heads	8
Blocks	7
Drop-out	0.1
$d_{feature}$	192
Batch size	256
Optimizer	*Schedule-free AdamW*
Initial learning rate	0.001
Warm-up steps	1000

Table 3: *FTTransformers* and Training Hyperparameters

***Time Series Tokenizers* Validation Performance**

Time Serie Tokenizer	**Evaluated hyperparameter**	**Hyperparameter value**	**Validation AUC**
MLP	Number of neurons	250	96.55
MLP	Number of neurons	500	96.46
FCN	Number of feature maps	64	96.31
FCN	Number of feature maps	128	96.27
ResNet	Number of blocks	2	96.22
ResNet	Number of blocks	3	96.26
Transformer Encoder	Number of heads	4	96.58
Transformer Encoder	Number of heads	8	96.56

Table 4: *Time Series Tokenizers* hyperparameter tuning experiments and associated validation AUC

Transformer Encoder Hyperparameters

Hyperparameter	Value
Heads	4
Blocks	2
Drop-out	0.1
$d_{feature}$	16

Table 5: ransformer Encoder Hyperparameters