

# Sowshank Attention

*Every pixel deserves a second chance*

NATIONAL TECHNICAL UNIVERSITY OF ATHENS  
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING



*Evangelos Chaniadakis*  
[echaniadakis@gmail.com](mailto:echaniadakis@gmail.com)

03400279



**Submission Date**  
May 17, 2025

# Contents

<b>I</b>	<b>Introduction</b>	<b>2</b>
<b>II</b>	<b>Dataset</b>	<b>3</b>
I	Geographical & Chronological Coverage . . . . .	3
II	Spectral & Temporal Characteristics . . . . .	3
III	Hierarchical Structure . . . . .	3
IV	Temporal Analysis at Different Spatial Scales . . . . .	5
IV.I	Spectral Evolution at the Pixel Level . . . . .	5
IV.II	Phenological Development at the Parcel Level . . . . .	6
<b>III</b>	<b>Pre-processing</b>	<b>7</b>
I	Category Filtering and Indexing . . . . .	7
II	Feature Extraction . . . . .	9
III	Data Augmentation . . . . .	9
IV	Data Normalization . . . . .	10
V	Cloud Coverage Mitigation . . . . .	10
VI	Pixel Sampling . . . . .	11
<b>IV</b>	<b>Model Architecture</b>	<b>12</b>
I	Pixel Set Encoder . . . . .	12
II	Transformer Time Encoder . . . . .	13
II.I	Attention and CLS Token . . . . .	14
III	Classifier Head . . . . .	14
III.I	Loss Function . . . . .	14
III.II	Optimization . . . . .	15
IV	Classification Metrics . . . . .	15
<b>V</b>	<b>Training</b>	<b>16</b>
I	Split Strategy . . . . .	16
II	Model Initialization and Optimization . . . . .	16
III	Training & Evaluation Procedure . . . . .	17
IV	Performance Assessment and Visualization . . . . .	17
V	K-fold Cross-Validation . . . . .	19
<b>VI</b>	<b>Inference</b>	<b>20</b>
I	Test Dataset Preparation . . . . .	20
I.I	Label Mapping . . . . .	20
II	Inference Pipeline . . . . .	21
III	Evaluation and Results . . . . .	22

## CHAPTER I

---

# Introduction

---

**S**his work tackles the challenge of classifying crop types using satellite-based time series data. Specifically, we focus on the *TimeMatch* dataset, which contains multispectral time series data collected from Sentinel-2 imagery. The dataset covers agricultural parcels across four distinct regions in Europe, providing valuable insights into crop classification. While the broader aim of TimeMatch is to support various agricultural tasks like crop classification and domain adaptation across different regions, this study zooms in on crop classification for a specific region and timeline. For practical purposes, we utilize a substantially reduced subset of the dataset, provided by the course instructor via [Google Drive](#), which specifically corresponds to the Denmark region for the year 2017. The goal is to develop a reliable pipeline for classifying agricultural parcels into one of 15 crop types. The model is designed to effectively classify crop types by leveraging the spatiotemporal nature of satellite time series data. It begins with a Pixel Set Encoder, which captures spatial relationships within each parcel by processing pixels, outputting only temporal and feature dimensions. A Transformer Encoder is then used to capture temporal dynamics, enhanced by sinusoidal temporal positional encodings derived from the acquisition dates of the samples. A classification token aggregates the temporal information, allowing the model to focus on the overall trends. Subsequently, a custom Multi-Layer Perceptron is employed to perform the final classification, enabling accurate identification of crop types across the dataset.

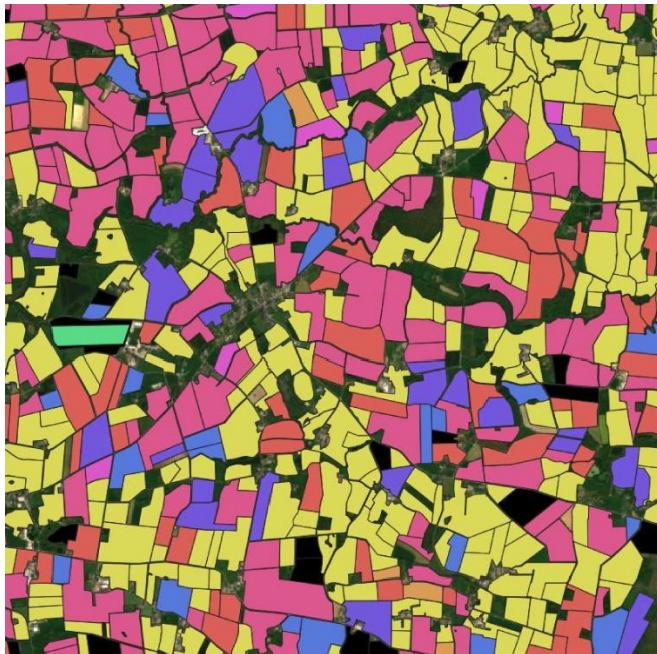


Figure I: Ground truth crop type labels in a sample region of the *TimeMatch* dataset.

## CHAPTER II

---

# Dataset

---



The TimeMatch dataset is a satellite-derived, multivariate time series dataset developed to support research in remote sensing and temporal modeling. It is based on Sentinel-2 imagery and captures the seasonal dynamics of agricultural parcels across multiple regions in Europe. The dataset is designed to facilitate a deeper understanding of vegetation phenology by providing rich spectral and temporal information at the pixel level.

## I Geographical & Chronological Coverage

TimeMatch comprises data from four geographically and climatologically distinct European countries: France, Spain, Italy and Denmark. Each region includes agricultural parcels observed over a single year. Imagery is uniformly sampled on a weekly basis, resulting in 52 temporally aligned acquisitions per year. All data is derived from Sentinel-2 Level-2A products, ensuring consistent surface reflectance calibration.

## II Spectral & Temporal Characteristics

Each pixel's time series consists of 52 weekly observations across 10 selected spectral bands:

- Blue (B2), Green (B3), Red (B4)
- Red Edge (B5, B6, B7)
- Near-Infrared (B8, B8A)
- Short-Wave Infrared (B11, B12)

These bands, selected for their sensitivity to vegetation traits like chlorophyll, biomass and water stress, produce a  $52 \times 10$  matrix per pixel, with temporal alignment across all pixels in a parcel.

$$\text{Pixel Data} \in R^{52 \times 10}$$

## III Hierarchical Structure

The dataset is structured hierarchically:

$$\text{Region} \rightarrow \text{UTM Tile} \rightarrow \text{Year} \rightarrow \text{Parcel} \rightarrow \text{Pixel} \rightarrow \text{Time Series}$$

At the top level, data are grouped by region and subdivided into Universal Transverse Mercator (UTM) tiles. Each tile contains data for a specific year, further organized into parcels, each representing an agricultural field. Parcels are composed of multiple pixels at 10-meter resolution, in accordance with Sentinel-2 standards. Each pixel is represented by a time series of length 52, with 10 spectral bands per time step.

The dataset is organized under the root directory with the following structure:

```

timematch_data
    denmark
        32VNH
            2017
                data
                    0.zarr
                        .zarray
                        0.0.0
                        0.1.0
                        1.0.0
                        1.1.0
                    1.zarr
                        .zarray
                        0.0.0
                        1.0.0
                    2.zarr
                    3.zarr
                    4.zarr
                meta
                    blocks
                        blocks_denmark_32VNH_2017.shp
                        blocks_denmark_32VNH_2017.shx
                        blocks_denmark_32VNH_2017.dbf
                        blocks_denmark_32VNH_2017.prj
                        blocks_denmark_32VNH_2017.cpg
                    dates.json
                    labels.json
                    filtered_labels.json
                    normalization_stats.json
                    metadata.pkl

```

## Data Files (data/)

Each file in the data/ directory (e.g., 0.zarr, 1.zarr) contains a single parcel stored in Zarr format. Each parcel directory includes:

- .zarray: Metadata about array shape, chunking and data type.
- Binary chunks named using Zarr's internal indexing scheme (e.g., 0.0.0, 1.0.0, 0.1.0).

Each Zarr array stores a 3D tensor of shape  $(52, 10, n_p)$  representing:

- 52 weekly time steps,
- 10 spectral bands,
- $n_p$  pixels (variable per parcel).

## Metadata Files (meta/)

The meta/ directory contains essential auxiliary files for dataset interpretation and spatial referencing:

- dates.json — JSON file listing the 52 acquisition dates common to all parcels in the year-tile, enabling temporal alignment of the time series data.
- labels.json — JSON mapping parcel IDs to their crop class labels, providing categorical annotations for supervised tasks.
- filtered\_labels.json — JSON file with filtered crop labels after preprocessing.
- metadata.pkl — A pickled Python dictionary containing parcel-level metadata such as:
  - id (string) — parcel identifier,
  - label (float) — numeric crop class,
  - n\_pixels (integer) — number of pixels in the parcel,

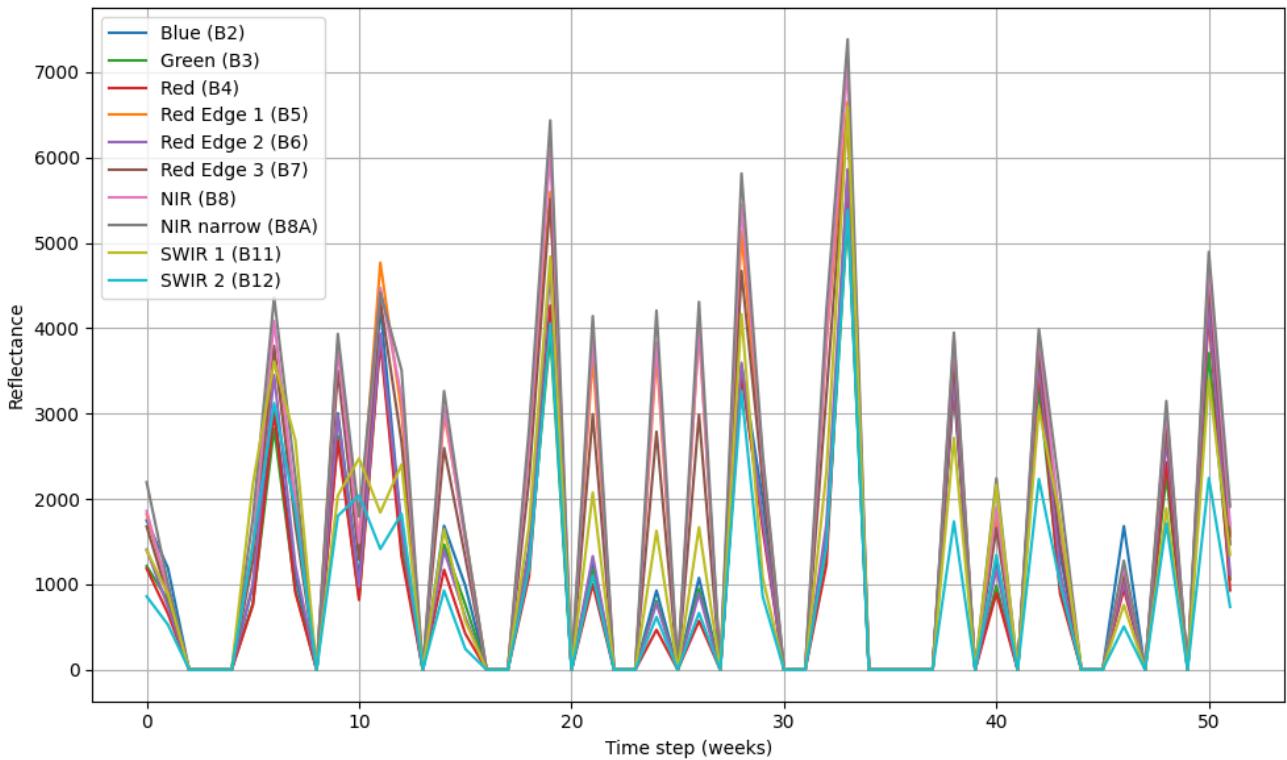
- block (integer) — identifier linking the parcel to a spatial block,
- geometric\_features (list of floats) — shape descriptors or spatial metrics (e.g., area, perimeter, compactness).
- blocks/ directory — contains shapefile components that together define the exact spatial boundaries of parcels:
  - .shp — main file storing parcel polygon geometries,
  - .shx — index file linking geometry records for efficient access,
  - .dbf — attribute table storing tabular parcel information (IDs, labels, etc.),
  - .prj — coordinate reference system (CRS) definition, specifying spatial projection (e.g., UTM),
  - .cpg — text encoding of the attribute table, ensuring correct character representation.

The directory also includes two files generated during preprocessing, detailed later.

## IV Temporal Analysis at Different Spatial Scales

### IV.I Spectral Evolution at the Pixel Level

The figure below illustrates the spectral time series for a single pixel 0 from parcel 0 of the Denmark 2017 subset, showcasing reflectance across all 10 bands. Reflectance values range from 0 to 7000, measured weekly over the 52-week period, corresponding to a full agricultural cycle. The reflectance profiles exhibit distinct seasonal variations, with pronounced peaks between weeks 17–20 (late April to mid-May) and weeks 31–35 (early August to early September).



These periods likely correspond to critical phenological stages in Danish agricultural cycles: the spring peak aligning with rapid vegetative growth and the late-summer peak reflecting senescence and harvest activities. Such temporal patterns are characteristic of temperate cropping systems and underline the importance of capturing seasonal dynamics for accurate classification. The NIR (B8) and Narrow NIR (B8A) bands display the highest reflectance, exceeding

6000 at their peaks, due to their sensitivity to vegetation biomass and canopy structure. The Red Edge bands (B5–B7) show intermediate reflectance, peaking around 4000–6000, reflecting their utility in detecting transitions in vegetation health. In contrast, the Blue (B2) and Red (B4) bands exhibit lower reflectance, due to chlorophyll absorption during active growth phases. The SWIR bands (B11, B12) demonstrate more stable reflectance, typically below 4000, indicative of their sensitivity to soil moisture and water content, which varies less dramatically over the season.

### IV.II Phenological Development at the Parcel Level

In Figure I we observe a curated time series of RGB composites for two distinct agricultural parcels: one classified as *Winter cereal* and the other as *Spring cereal*. The sequence spans multiple acquisition dates from January to October, capturing the evolving spectral signatures of these crop types.

Each row corresponds to a single parcel, with successive columns illustrating the parcel's appearance at different timestamps. The top row traces the development of a *Winter cereal* parcel, while the bottom row depicts a *Spring cereal* parcel. Preliminary observations suggest that *Winter cereals* exhibit an accelerated phenological cycle, with notable vegetative growth as early as March, indicative of their winter planting schedule. In contrast, *Spring cereals* appear to follow a delayed trajectory, with pronounced greening emerging only in mid to late spring. These divergent temporal signatures underscore the potential of longitudinal multispectral data to differentiate crop types that might otherwise appear spectrally similar in a single observation.

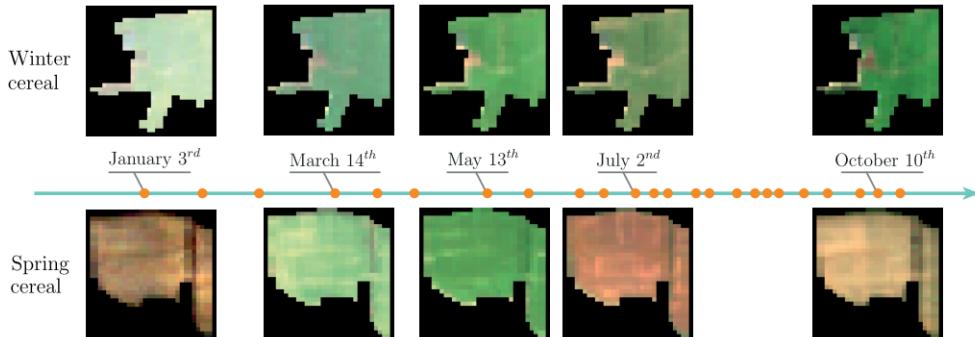


Figure I: Example of Sentinel-2 time series (RGB bands, 10m per pixel) for two agricultural parcels representing *Winter cereal* (top row) and *Spring cereal* (bottom row).

The horizontal timeline annotated with orange dots marks the actual acquisition dates of the satellite, emphasizing the irregular temporal sampling, typical of optical satellite data due to cloud cover and orbital constraints. This irregularity adds complexity to modeling efforts and motivates the use of temporal models capable of handling uneven and variable-length sequences.

## CHAPTER III

---

# Pre-processing

---



The preprocessing stage is a critical component of the crop classification pipeline, ensuring that the *TimeMatch* dataset is appropriately cleaned, filtered and normalized for downstream model training. Hereby we describe the two primary preprocessing steps: category filtering and indexing and data normalization. These steps address class imbalance, remove noisy or underrepresented data and standardize the spectral reflectance values to facilitate robust model performance. The implementation is provided in two scripts: `category_preprocessing.py` and `normalization.py`, which operate on the Denmark 2017 subset of the dataset.

## I Category Filtering and Indexing

The first preprocessing step involves filtering and reindexing the crop type labels to ensure a balanced and representative dataset. The raw labels are stored in `labels.json` within the `meta/` directory, mapping parcel IDs to their corresponding crop type labels. Due to the presence of underrepresented classes, which could negatively impact model training, we filter out categories with insufficient samples and assign new indices to the remaining classes.

We load the label data from `labels.json` and calculate the frequency of each crop type. For the Denmark 2017 dataset, the initial category counts are

Class	Count
corn	275
spring_barley	1140
meadow	1013
winter_wheat	856
winter_rapeseed	301
unknown	511
winter_barley	352
winter_rye	316
spring_peas	17
spring_oat	120
horsebeans	28
winter_triticale	42
spring_wheat	26
spring_triticale	2

Table I: Category Counts

To ensure robust model training, we exclude crop types with fewer than a minimum number of samples, defined as 200. For the Denmark 2017 dataset, this results in 8 retained classes, as several categories (e.g., `spring_peas`, `spring_oat`, `spring_triticale`) have fewer than 200 samples.

After filtering, the remaining 8 labels are reindexed to create a contiguous set of class indices suitable for model training. We assign indices to labels, prioritizing the unknown class to ensure it receives index 0, while sorting other labels in descending order of their respective counts, ensuring that more frequent classes receive lower indices. For the Denmark 2017 dataset, the filtered label indexing is:

Class	Index
unknown	0
spring_barley	1
meadow	2
winter_wheat	3
winter_barley	4
winter_rye	5
winter_rapeseed	6
corn	7

Table II: Filtered Label Indexing

The finalized dataset comprises 4,764 land parcels categorized into eight distinct classes. The corresponding labels are stored in the `filtered_labels.json` file within the metadata directory. To facilitate a better understanding of these crop categories, we provide representative images below, offering us a clear visual insight into their typical appearance and distinguishing features.



Figure I: Representative visual examples of each of the resulting land cover classes.

## II Feature Extraction

To enrich the spectral representation of crop parcels, we compute two widely adopted vegetation indices: the Normalized Difference Vegetation Index (NDVI) and the Enhanced Vegetation Index (EVI). These indices, as introduced by Huete et al. (2002), are commonly used in remote sensing for assessing plant health, growth stages and overall vegetation dynamics. By incorporating these biologically meaningful features, we aim to improve the model’s discriminative power for crop classification.

NDVI quantifies vegetation greenness by exploiting the strong reflectance of healthy vegetation in the near-infrared (NIR) spectrum and its absorption in the red spectrum. It is defined as the normalized difference between NIR and red reflectance, thereby enhancing the contrast between vegetated and non-vegetated surfaces.

EVI extends NDVI by integrating the blue band, which helps correct for atmospheric disturbances and soil background effects. This makes EVI particularly advantageous in conditions with high vegetation density and varying atmospheric interference—conditions frequently observed in our dataset, which exhibits an average cloud cover of 40.86%.

The NDVI and EVI values are computed per pixel and per time step using the following formulations:

$$\text{NDVI} = \frac{\text{NIR} - \text{Red}}{\text{NIR} + \text{Red} + 10^{-10}}$$

$$\text{EVI} = 2.5 \cdot \frac{\text{NIR} - \text{Red}}{\text{NIR} + 6 \cdot \text{Red} - 7.5 \cdot \text{Blue} + 1 + 10^{-10}}$$

where NIR corresponds to band 8 (B8), Red to band 4 (B4) and Blue to band 2 (B2). These indices are appended as additional channels to the existing 10-band input, expanding the feature tensor from  $(n_p, T_v, 10)$  to  $(n_p, T_v, 12)$ , where  $n_p$  is the number of pixels per parcel and  $T_v$  is the number of time steps.

## III Data Augmentation

To enhance the model’s generalization and robustness, especially given the limited size of our dataset, we implement data augmentation techniques during training within the `ParcelDataset` class. These techniques introduce controlled variability to the spectral and temporal data, simulating real-world conditions such as sensor noise, temporal shifts and atmospheric effects. The augmentation is applied probabilistically to a tensor of shape  $(N, T, C)$ , where  $N$  is the number of sampled pixels (32),  $T$  is the number of time steps (52) and  $C$  is the number of channels (12). The following techniques are employed:

- **Spectral Jitter:** With a 80% probability, Gaussian noise with a mean of 0 and a standard deviation of 0.05 is added to each channel value, simulating sensor noise or slight variations in reflectance.
- **Time Shift:** With a 50% probability, a random temporal shift between -5 and 5 time steps is applied using a circular shift (`np.roll`), mimicking slight misalignments in acquisition timing or phenological variations across parcels.
- **Random Band Dropout:** With a 30% probability, 1 to 2 channels are randomly selected and set to zero, simulating missing or corrupted spectral bands due to cloud cover or sensor failure. The number of dropped channels is a randomly selected integer between 1 and 2.

- **Gaussian Blur in Time:** With a 30% probability, a Gaussian filter with a standard deviation of 1.0 is applied along the time axis for each pixel, smoothing the temporal sequence to simulate atmospheric scattering or averaging effects.

## IV Data Normalization

The second preprocessing step normalizes the spectral reflectance values across the 10 spectral bands and the extracted features to standardize the data distribution. This is essential for improving model convergence and ensuring that features are on a comparable scale. The normalization statistics are computed over all pixels in the filtered parcels and saved in normalization\_stats.json.

### Computing Channel-wise Statistics

The normalization.py script processes the parcel data stored in Zarr format within the data/ directory. For each of the 4764 parcels listed in filtered\_labels.json, the script loads the corresponding Zarr array (e.g., 0.zarr), which originally has shape  $(52, 10, n_p)$ , representing 52 time steps, 10 spectral bands and  $n_p$  pixels. After augmenting the data with NDVI and EVI, the shape becomes  $(52, 12, n_p)$ . The data is then reshaped to  $(n_p, 52, 12)$  to facilitate pixel-wise computations and then flattened to  $(n_p \cdot 52, 12)$  for statistical aggregation. We compute the mean and variance for each spectral band using Welford's online algorithm, which is numerically stable for large datasets. For each pixel's spectral vector  $x \in R^{12}$ , the running mean and variance are updated as follows:

$$\begin{aligned}\delta &= x - \mu_t \\ \mu_{t+1} &= \mu_t + \frac{\delta}{t+1} \\ M2_{t+1} &= M2_t + \delta \cdot (x - \mu_{t+1})\end{aligned}$$

where  $\mu_t$  is the running mean at step  $t$ ,  $M2_t$  is the running sum of squared differences and  $t$  is the number of observations processed.

The final mean and standard deviation for each channel are

$$\mu = \mu_T \quad \& \quad \sigma = \sqrt{\frac{M2_T}{T-1}}$$

where  $T$  is the total number of pixel observations. This process is applied across all valid parcels, skipping any with missing files or incompatible shapes.

## V Cloud Coverage Mitigation

To address the adverse impact of cloud cover, which averages 40.86% across our dataset, we explored a temporal filtering strategy. Specifically, we experimented with selecting the top- $K$  time steps exhibiting the lowest cloud coverage, as measured by the cloudy\_pct metric. Setting top\_k = 32, we retained the 32 least cloudy observations out of 52 total time steps per parcel. However, this approach did not yield any notable performance improvements. A likely explanation is that the presence of cloud-contaminated observations introduces variability that may act as a form of implicit data augmentation. Consequently, we opted to retain all available time steps, regardless of their cloud coverage levels.

## VI Pixel Sampling

Because parcels contain varying numbers of pixels, we needed a consistent way to represent them as input to the model. To address this, we chose to sample a fixed number of pixels per parcel (`sample_pixels = 32`). Throughout development, we explored several sampling strategies in search of a balance between representativeness, computational efficiency, and overall model performance.

- **K-means clustering:** We initially tried grouping pixels into three clusters using K-means, following the approach in MacQueen (1967), and then sampled proportionally from each cluster to promote spectral diversity. Although theoretically appealing, the method’s computational cost ( $O(n \cdot k \cdot i)$ ) was significant, and we saw no clear performance gain—leading us to move on.
- **Stratified random sampling:** Drawing on ideas from Thompson (2012), we divided pixels into quartiles based on spectral variance and sampled evenly from each. While statistically sound, this method didn’t outperform simpler techniques in our experiments.
- **Farthest Point Sampling (FPS):** Inspired by Eldar et al. (1997), we tested FPS to select spectrally dissimilar pixels in an iterative way. It did provide diverse samples, but the added computational cost and inconsistent performance meant it wasn’t the most practical option.
- **Spatially-aware sampling:** We also considered spatial characteristics, following the guidelines of J. Li and Heap (2015), using either distance-based or grid-based methods. Unfortunately, due to missing or noisy spatial coordinates in some parcels, we couldn’t reliably apply this approach across the dataset.
- **Entropy-based sampling:** Motivated by the idea that more temporally dynamic pixels might carry richer information, we tried selecting high-entropy pixels over time, inspired by Zhang, Wang, and X. Li (2016). This was interesting conceptually, but in practice, the added complexity didn’t translate into consistent gains.

In the end, uniform random sampling stood out for its simplicity ( $O(n)$ ), speed, and reliable performance. Despite being the most straightforward option, it consistently matched or outperformed the more complex strategies, making it the most practical choice for our pipeline.

## CHAPTER IV

# Model Architecture



The crop classification model, denoted as ParcelModel, is designed to classify agricultural parcels into one of eight crop types using the preprocessed *TimeMatch* dataset, specifically the Denmark 2017 subset. The model leverages the spatiotemporal characteristics of Sentinel-2 satellite imagery through three carefully designed components. The PixelSetEncoder captures spatial features from variable pixel sets. The TransformerTimeEncoder models temporal dynamics using attention mechanisms. The ClassifierHead translates the learned representations into crop type probabilities.

The ParcelModel is engineered to process multispectral time series data, capturing both the spatial variability inherent in agricultural parcels and the temporal dynamics across 52 weekly observations. Each parcel comprises a variable number of pixels, each characterized by a sequence of spectral measurements across 12 channels (10 Sentinel-2 bands + NDVI + EVI), accompanied by temporal metadata specifying acquisition dates as day-of-year indices. The model’s architecture is tailored to handle this complexity, ensuring robustness against irregular pixel counts and non-uniform temporal sampling. The forward pass, illustrated in Figure I, delineates the sequential processing of input data through the model’s components, transforming raw spatiotemporal inputs into class predictions.

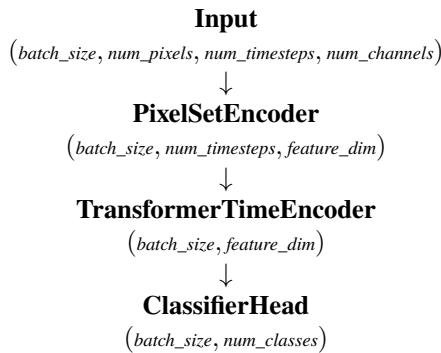


Figure I: Forward pass of the ParcelModel.

## I Pixel Set Encoder

Central to the ParcelModel’s design is the PixelSetEncoder, which addresses the challenge of processing parcels with variable pixel counts. This component draws inspiration from set-based neural architectures introduced by Zaheer et al. (2017), which models inputs as unordered sets to ensure permutation invariance. More precisely, the architecture builds upon the work of Garnot et al. (2020), who proposed the Pixel-Set Encoder as an effective mechanism for summarizing spatially distributed pixel-level features into a compact, permutation-invariant representation.

This summarization facilitates robust integration of spatial information prior to temporal modeling, which is crucial in high-resolution satellite imagery where intra-parcel heterogeneity can obscure crop-specific patterns. In the context of the *TimeMatch* dataset, the PixelSetEncoder processes each pixel’s spectral features—12 channels comprising 10 Sentinel-2 bands augmented with NDVI and EVI—using a shared multi-layer perceptron (MLP). The MLP consists of two linear layers: the first projects the 12-dimensional input to a 64-dimensional hidden space with a ReLU activation and the second maps this to a 128-dimensional output, producing a feature vector for each pixel at each time step.

To further enhance the encoder’s capacity to prioritize informative pixels (e.g., vegetation as opposed to soil or shadows), we integrate a self-attention mechanism inspired by the Transformer architecture of Vaswani et al. (2017). Following the MLP, a linear projection is applied to compute a scalar attention score for each pixel. These scores are normalized via a softmax function across the pixel dimension to yield attention weights, which are used to compute a weighted sum of pixel embeddings.

$$\text{attn\_scores} = W_{\text{attn}} \cdot x, \quad \text{attn\_weights} = \text{softmax}(\text{attn\_scores}), \\ x_{\text{agg}} = \sum_{\text{pixels}} (\text{attn\_weights} \cdot x),$$

where  $x$  is the MLP output of shape  $(batch\_size, 52, num\_pixels, 128)$  and  $W_{\text{attn}}$  is a learned linear transformation producing scalar attention scores. This mechanism enables the model to emphasize pixels containing significant spectral information, thereby generating a more informative and noise-resilient parcel-level representation. The final output of the PixelSetEncoder is a tensor of shape  $(batch\_size, 52, 128)$ , representing a temporally ordered sequence of spatially aggregated features, which is subsequently passed to downstream temporal models for classification.

## II Transformer Time Encoder

The output of the PixelSetEncoder, which retains temporal and feature dimensions, serves as input to the TransformerTimeEncoder, a component designed to model the temporal evolution of spectral features critical for capturing crop phenology. Phenological patterns, such as the growth stages of winter\_wheat or the flowering of spring\_barley, are pivotal for distinguishing crop types with overlapping spectral signatures. The TransformerTimeEncoder leverages the Transformer architecture, introduced by Vaswani et al. (2017), renowned for its self-attention mechanism that computes weighted interactions between time steps. This mechanism enables the model to prioritize significant phenological events—such as peak vegetation or senescence—irrespective of their precise timing, a capability particularly suited to the non-uniform temporal patterns in satellite time series, as demonstrated by Rußwurm and Körner (2020). To incorporate temporal context, the encoder employs sinusoidal positional encodings, also proposed by Vaswani et al. (2017), which map day-of-year indices to continuous vectors. These encodings, defined for a time step  $t$  and model dimension  $d_{\text{model}}$  as:

$$\text{PE}(t, 2i) = \sin\left(t/10000^{2i/d_{\text{model}}}\right) \quad \& \quad \text{PE}(t, 2i + 1) = \cos\left(t/10000^{2i/d_{\text{model}}}\right),$$

preserve temporal relationships and accommodate variable sequence lengths, ensuring the model captures seasonal cyclicity essential for distinguishing crops like winter\_rye from spring\_barley.

## II.I Attention and CLS Token

The TransformerTimeEncoder incorporates a learnable classification token ([CLS]), inspired by the BERT model developed by Devlin et al. (2018), which aggregates temporal information into a single vector. The [CLS] token is prepended to the sequence, increasing its length. The transformer, configured with 4 attention heads and 2 layers, processes the sequence using multi-head self-attention, allowing the model to focus on critical time steps. The output corresponding to the [CLS] token, of shape (*batch\_size*, 128), is extracted as the final parcel representation.

To enhance regularization and prevent overfitting, we added a dropout layer (with a rate of 0.1) after the transformer, following the attention mechanism. Additionally, batch normalization is applied to the [CLS] token output to stabilize training, especially given the small dataset size (4,764 parcels). The final output of the TransformerTimeEncoder is a tensor of shape (*batch\_size*, 128), ready for classification.

## III Classifier Head

The ClassifierHead constitutes the final stage of the ParcelModel, mapping the aggregated temporal representation to class logits for the eight crop types defined during preprocessing. Implemented as a multi-layer perceptron (MLP), this component exploits the non-linear mapping capabilities of neural networks, as delineated by Hastie, Tibshirani, and Friedman (2009), to delineate complex decision boundaries among crop classes. Dropout regularization mitigates overfitting, a critical issue in remote sensing where labeled data is limited, as established by Belgiu and Drăguț (2016). The MLP consists of a linear layer mapping the input dimension of 128 to a hidden dimension of 64, followed by batch normalization, a ReLU activation, a dropout layer and a final linear layer to 8 output classes. The output layer aligns with the filtered label set, excluding underrepresented classes such as spring\_peas, ensuring predictions are tailored to the dataset's class distribution.

### III.I Loss Function

To quantify our model's performance during training, we employ the Categorical Cross-Entropy loss function, introduced by Goodfellow, Bengio, and Courville (2016). This loss function quantifies the divergence between predicted class probabilities and true labels, making it optimal for multi-class classification tasks. For a given parcel with true class label  $y_i \in \{1, \dots, 8\}$  and predicted probabilities  $p_i$ , the loss is computed as:

$$L = - \sum_{i=1}^8 y_i \log(p_i),$$

where  $y_i$  is a one-hot encoded vector representing the true class and  $p_i$  is the predicted probability for class  $i$ . This formulation penalizes incorrect predictions by assigning higher loss to low-probability assignments for the true class, thereby encouraging the model to assign high confidence to correct crop type predictions.

To address class imbalance in the *TimeMatch* dataset (e.g., 1140 *spring\_barley* vs. 275 *corn* parcels), the loss is weighted using class weights derived from the frequency of each class in *filtered\_labels.json*. The weights are computed as the inverse of the class counts, normalized to sum to the number of classes (8), ensuring that underrepresented classes contribute more to the loss:

$$\text{class\_weights}_i = \frac{1/\text{count}_i}{\sum_{j=1}^8 (1/\text{count}_j)} \cdot 8,$$

where  $\text{count}_i$  is the number of samples for class  $i$ .

### III.II Optimization

Model optimization employs the Adam optimizer, introduced by Kingma and Ba (2014). Adam integrates first-order gradient-based optimization with adaptive moment estimation, computing exponentially weighted moving averages of gradients (first moment) and squared gradients (second moment) to adaptively adjust per-parameter learning rates. For the ParcelModel, this method optimizes the high-dimensional parameter space, encompassing the multi-layer perceptron weights of the ClassifierHead and the attention mechanism parameters of the TransformerTimeEncoder and PixelSetEncoder. The optimizer is configured with a learning rate ( $\alpha$ ) and a weight decay of  $1 \times 10^{-4}$  for L2 regularization. The algorithm minimizes the weighted Categorical Cross-Entropy loss through iterative updates, defined as:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1 - \beta_1) g_t, & v_t &= \beta_2 v_{t-1} + (1 - \beta_2) g_t^2, \\ \hat{m}_t &= \frac{m_t}{1 - \beta_1^t}, & \hat{v}_t &= \frac{v_t}{1 - \beta_2^t}, & \theta_{t+1} &= \theta_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}, \end{aligned}$$

where  $g_t$  is the gradient at time  $t$ ,  $m_t$  and  $v_t$  are the first and second moment estimates,  $\hat{m}_t$  and  $\hat{v}_t$  are bias-corrected estimates,  $\alpha$  is the learning rate,  $\beta_1$  and  $\beta_2$  are exponential decay rates,  $\epsilon$  is a small constant for numerical stability and  $\theta_t$  represents model parameters. This formulation ensures efficient convergence for the eight-class classification task. To further enhance convergence, a learning rate scheduler (ReduceLROnPlateau) is employed, which reduces the learning rate by a factor of 0.5 when the validation F1 score plateaus, with a patience of specified epochs.

## IV Classification Metrics

The integration of these components within the ParcelModel creates a hierarchical architecture that synergistically exploits spatial and temporal patterns. The PixelSetEncoder reduces spatial complexity by aggregating pixel-level data, the TransformerTimeEncoder captures seasonal dynamics through attention and positional encodings and the ClassifierHead delivers accurate classifications. This design mirrors advanced remote sensing architectures, such as those proposed by Garnot et al. (2020), balancing computational efficiency with expressive power. To evaluate performance, the model employs the ClassificationMetrics class, which computes metrics tailored to multi-class problems. Accuracy and micro-averaged F1-score provide overall performance insights, while weighted F1-score, precision and recall account for class imbalance, such as the disparity between spring\_barley (1140 parcels) and corn (275 parcels). The confusion matrix, as emphasized by Congleton and Green (1991), reveals misclassification patterns, facilitating analysis of errors between phenologically similar crops like winter\_wheat and winter\_barley.

## CHAPTER V

---

# Training

---



The training process for the ParcelModel optimizes its performance in classifying agricultural parcels into one of eight crop types using the preprocessed *TimeMatch* dataset, specifically the Denmark 2017 subset. Hereby, we describe the training pipeline, implemented in the script `train.py`, which leverages the spatiotemporal architecture to address challenges such as data variability, class imbalance and overfitting, ensuring robust and generalizable model performance.

## I Split Strategy

Training begins with the preparation of the *TimeMatch* dataset, utilizing the `ParcelDataset` class to load and preprocess the multispectral time series data, as outlined in the preprocessing chapter. Our code supports two splitting strategies: a stratified train-test split or K-fold cross-validation, controlled via a command-line argument (`-mode`).

For the stratified split, the dataset is divided into training (80%) and validation (20%) sets using `sklearn.model_selection.train_test_split` with stratification based on class labels. This ensures proportional representation of each class in both sets, addressing our class imbalance problem.

Alternatively, for K-fold cross-validation, the dataset is split into 5 folds (`k_folds=5`) with shuffling, as introduced by Kohavi (1995).

The training data is batched with a size of 32 and shuffled to promote stochastic gradient updates, using PyTorch's `DataLoader` with `drop_last=True` to ensure consistent batch sizes. The validation data is processed in batches of the same size without shuffling (`drop_last=False`) to maintain consistency during evaluation.

## II Model Initialization and Optimization

The ParcelModel is initialized with its core components: the `PixelSetEncoder`, `TransformerTimeEncoder` and `ClassifierHead`. The encoder maps the 12 input channels (10 Sentinel-2 bands + NDVI + EVI) to a 128-dimensional feature space (`in_channels=12`, `out_dim=128`) through a hidden dimension of 64 (`hidden_dim=64`). The transformer processes the temporal sequence with an input dimension of 128 (`input_dim=128`) and the classifier outputs logits for the 8 crop classes (`num_classes=8`). The model is transferred to the available device, preferably a GPU (`cuda`) or a plain CPU (`cpu`).

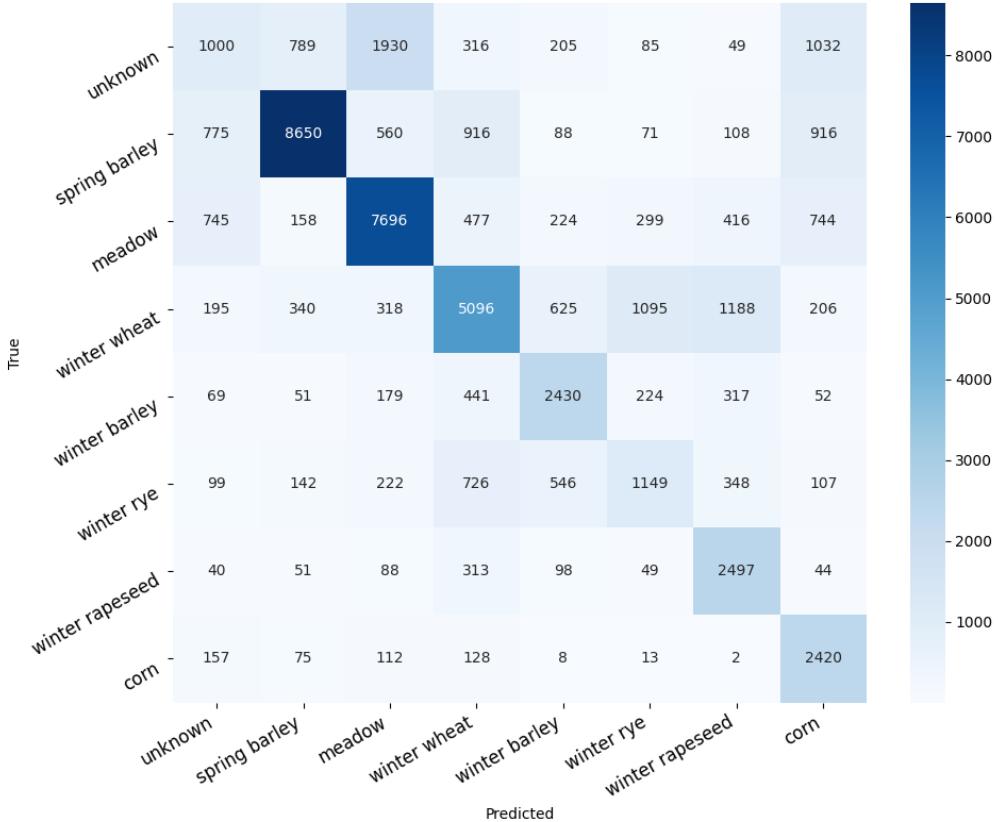
Optimization employs the Adam optimizer, with a learning rate of  $10^{-4}$  (`lr=1e-4`) and L2 regularization via `weight_decay=1e-4`. A learning rate scheduler, `ReduceLROnPlateau`, adjusts the learning rate by a factor of 0.5 if the validation F1-score does not improve for 3 epochs. The loss function is a weighted Categorical Cross-Entropy, to address class imbalance.

### III Training & Evaluation Procedure

Training proceeds over a maximum of 100 epochs, with each epoch involving a full pass over the training set. In training mode (`model.train()`), gradients are computed for each batch. The input data: spectral features ( $x$ ), labels ( $y$ ) and day-of-year metadata ( $doy$ ), are moved to the designated device and the optimizer's gradients are zeroed with `optimizer.zero_grad()`. The model produces class logits through its forward pass and the weighted loss is computed. Gradients are backpropagated with `loss.backward()` and the optimizer updates the model parameters with `optimizer.step()`, iteratively minimizing the loss. Progress bars via `tqdm` monitor batch processing. Following each epoch, the model is evaluated on the validation set in evaluation mode (`model.eval()`) and gradient computations are disabled with `torch.no_grad()`. Validation metrics are computed and the scheduler adjusts the learning rate based on the validation weighted F1-score. To prevent overfitting, an early stopping mechanism is employed, as proposed by Prechelt (1998). If the validation weighted F1-score does not improve for 10 consecutive epochs, training halts. The model with the best validation F1-score is checkpointed to disk in the `checkpoints/` directory, preserving optimal weights for subsequent evaluation.

### IV Performance Assessment and Visualization

After training, we assess the top-performing model using the validation set to gain a clear understanding of its overall effectiveness. This evaluation reveals key performance insights, showcasing how well the model distinguishes between crop types. Notably, the confusion matrix sheds light on classification challenges, especially with phenologically similar crops like winter wheat and winter barley, helping us understand where improvements might be needed.



The confusion matrix shows that the model performs well overall, with strong classification accuracy for key classes like meadow, spring\_barley and winter\_wheat, all having high values along the diagonal. Meadow in particular stands out with over 8600 correct predictions, indicating that it is easily distinguishable. However, there is noticeable confusion between several agriculturally similar classes. For example, spring\_barley is often misclassified as winter\_wheat and corn, which is understandable given their likely spectral or seasonal similarities. Winter\_wheat also sees frequent misclassification into winter\_rye and winter\_barley, which are phenologically close and as we saw in Figure I are visually quite similar. Corn is reasonably well predicted, but still shows confusion with other classes, particularly meadow and unknown. The unknown class performs the worst, with significant misclassifications into nearly all other categories, most notably meadow and corn, reflecting its inherently undefined and heterogeneous nature. This broad confusion arises because the unknown class encompasses diverse or ambiguous samples that do not clearly belong to any specific class, making it inherently difficult for the model to learn. Overall, while core crop types are recognized well, the model struggles with edge cases and visually similar cereals, highlighting the need for improved feature discrimination or enhanced training data representation.

To capture the model’s learning trajectory, we present training curves that track the training loss and validation F1-score over time. These plots provide a clear, high-level view of how the model’s performance evolves during training, highlighting both its learning progress and generalization ability.

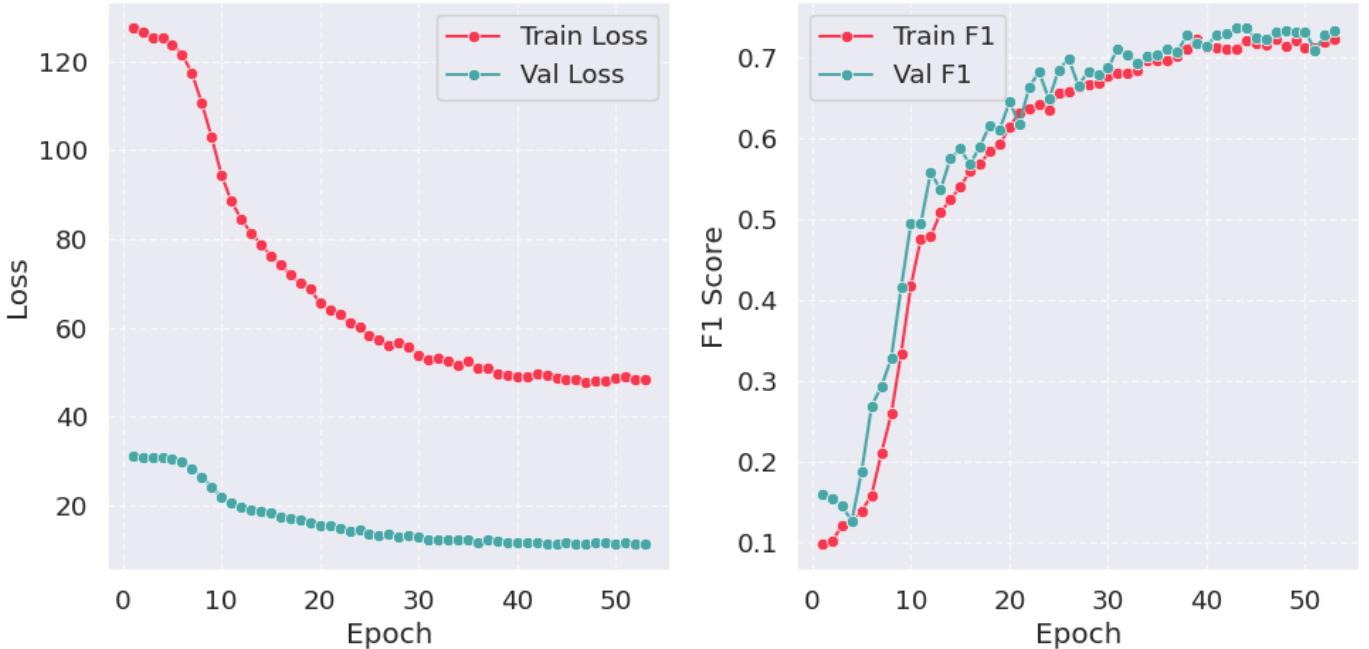


Figure I: Training Metrics

In the figure above, the training loss consistently decreases but remains higher than the validation loss. This behavior is attributed to the training phase, where parcels are sampled rather than using their full pixel representation, combined with the application of extensive data augmentation. The training F1 score exhibits a steady improvement, while the validation F1 score, although slightly more variable, follows a similar upward trend. Ultimately, the validation F1 score plateaus, and the learning process is terminated by the early stopping mechanism.

## V K-fold Cross-Validation

To evaluate the model’s ability to generalize, we apply a 5-fold cross-validation scheme. The *TimeMatch* dataset is partitioned into five equal parts and the model is trained and validated five times, each time using a different fold for validation and the remaining four for training. This process ensures that every sample is used for validation exactly once.

Cross-validation is theoretically grounded in the idea of estimating the expected generalization error. Let  $\mathcal{D}$  be the dataset and let  $m_i$  denote the evaluation metric, in our case F1-score, on the  $i$ -th validation fold. The overall estimate of model performance is given by the mean:

$$\mu = \frac{1}{k} \sum_{i=1}^k m_i,$$

and its variability is quantified by the standard deviation:

$$\sigma = \sqrt{\frac{1}{k} \sum_{i=1}^k (m_i - \mu)^2},$$

where  $k = 5$ .

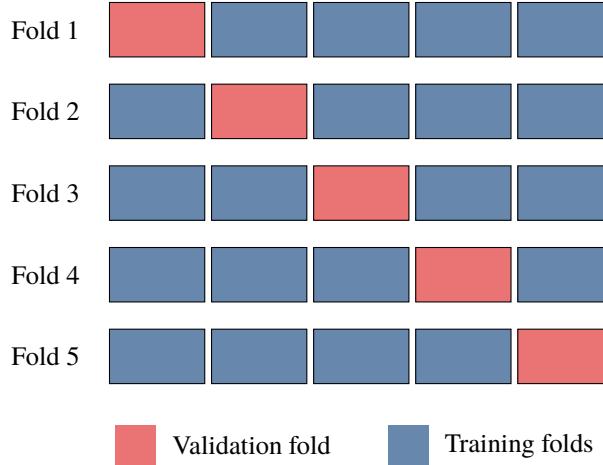


Figure II: Schematic of 5-fold cross-validation where each row represents one iteration.

In our case, the values  $m_1, m_2, \dots, m_5$  were nearly identical, resulting in quite low, almost zero  $\sigma$ . This suggests that the model’s performance is consistent across all folds, indicating strong stability and minimal sensitivity to the specific data partitioning. Theoretically, this implies that the model approximates the true underlying function  $f : \mathcal{X} \rightarrow \mathcal{Y}$  with little variance due to sampling. Such consistency across folds supports the conclusion that the data distribution is uniform across subsets and that the model is not overfitting to spurious patterns in particular splits. It also confirms that the learned representation generalizes well, aligning with the theoretical justification of cross-validation as an unbiased estimator of the true generalization error:

$$E_{\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}}} [\text{Error}] \approx \mu.$$

Finally, the model’s consistent confusion across folds between visually similar classes reflects inherent ambiguity in the data rather than instability in training, further validating the reliability of the evaluation process.

## CHAPTER VI

# Inference



The inference phase evaluates the trained ParcelModel on a previously unseen test dataset from the *TimeMatch* dataset, specifically the Austria 2017 subset, to assess its generalization across different geographical and temporal contexts. The model, trained on the Denmark 2017 subset, is applied to classify agricultural parcels into one of eight crop types using multispectral time series data derived from Sentinel-2 imagery.

## I Test Dataset Preparation

We downloaded the full *TimeMatch* dataset from [Zenodo](#) and extracted the Austria 2017 subset, which comprises of the UTM tile 33UVP. We observed that this subset, weirdly enough, contains 58 temporal observations, more than the 52 found in the Denmark 2017 subset. However, the Austrian data didn't come with usable labels out of the box, so we had to locate and process the corresponding metadata. To make the test set compatible with our trained model, we used the `prepare_test_labels.py` script to align the format and structure of the Austrian labels with those used during training.

### I.I Label Mapping

The test dataset's parcel labels, stored in a shapefile (`parcels_austria_33UVP_2017.shp`), contain German crop type names. We mapped the German labels to our eight target classes by defining a `mapping.json` file that specifies their English equivalents.

Superlabel	German Label	English Translation
corn	KÖRNERMAIS	Grain corn
	SILOMAIS	Silage corn
	MAIS CORN-COB-MIX (CCM)	Corn-cob-mix
	ZUCKERMAIS	Sweet corn
spring_barley	SOMMERGERSTE	Spring barley
	MÄHWIESE/-WEIDE ZWEI NUTZUNGEN	Mown meadow or pasture with two uses
	MÄHWIESE/-WEIDE DREI UND MEHR NUTZUNGEN	Mown meadow or pasture with three or more uses
	WECHSELWIESE (EGART, ACKERWEIDE)	Rotational meadow or field pasture
meadow	DAUERWEIDE	Permanent meadow or pasture
	EINMÄHDIGE WIESE	Meadow mown once per season
	HUTWEIDE	Grazed meadow or pasture
	ALMFUTTERFLÄCHE	Alpine meadow for forage
winter_wheat	WINTERWEICHWEIZEN	Winter soft wheat
	WINTERHARTWEIZEN (DURUM)	Winter durum wheat
winter_rapeseed	WINTERRAPS	Winter rapeseed
unknown	SONSTIGE ACKERFLÄCHEN	Other arable land
	SONSTIGE ACKERKULTUREN	Other field crops
winter_barley	WINTERGERSTE	Winter barley
winter_rye	WINTERROGGEN	Winter rye

Table I: Label mapping for the Austria 2017 dataset, showing the correspondence between model superlabels and German crop labels

The mapping, partially shown in Table I, associates each target class with one or more German labels, accommodating the diversity of crop designations in the Austria 2017 dataset. A JSON file `filtered_labels.json` is generated, retaining only parcels with labels matching our target classes. This filtering ensures that only relevant parcels are used for inference, addressing potential class mismatches between regions. Additional classes, like `spring_peas` and `spring_oat`, present in the mapping are excluded due to their underrepresentation, as determined during training. Each label is assigned a numeric index (0 to 7) consistent with the training dataset. Metadata, including acquisition dates, is extracted from `metadata.pkl` and saved as `dates.json` to be consistent with our training pipeline. The Austria 2017 subset contains 58 unique acquisition dates, compared to 52 in the Denmark 2017 subset.

## II Inference Pipeline

The inference procedure, implemented in `inference.py`, loads the pretrained `ParcelModel` from `best_model.pt` and performs evaluation on the Austria 2017 dataset, either in its entirety or on a stratified subset, depending on the specified configuration.

### Stratified Sampling

Due to the considerable size of the test dataset, nearly five times larger than the training set, inference over the full test set can be computationally prohibitive. To address this, a representative subset of the test data could be selected via stratified sampling to maintain the original class distribution. This is implemented using the `StratifiedShuffleSplit` utility from scikit-learn library. The stratification is based on crop type labels provided in `filtered_labels.json`, ensuring that each class is proportionally represented in the sampled subset. The `ParcelDataset` loads temporal parcel-level data from Zarr files located in the `data/` directory.

### Data Normalization and Variable-Length Collation

Prior to inference, test data are normalized too, using statistics, mean and standard deviation, computed specifically for the Austrian dataset via the `normalization.py` script. All pixels from each parcel are used to maximize the amount of spectral and temporal information available to the model.

Because parcels may contain different numbers of pixels and valid time steps, batching the data requires a flexible collation strategy. A custom collation function, `variable_length_collate`, is used to group samples without enforcing fixed dimensions. It returns lists of spectral sequences (`xs`), their corresponding labels (`ys`), and associated day-of-year values (`doy`s). This format accommodates the variable-length nature of the input data while remaining compatible with the model's input requirements.

### Model Inference

During inference, each parcel is processed individually due to the variable number of pixels per parcel. The spectral feature tensor `x` with dimensions `(num_pixels, 58, 12)` and the corresponding day-of-year metadata `doy` of shape `(58, )` are passed as inputs to the model. A batch dimension is added to the spectral features, resulting in an input shape of `(1, num_pixels, 58, 12)`. The model produces class logits through its forward pass, from which the predicted crop type is determined by selecting the class with the highest logit value (`arg max`). Predictions and true labels are aggregated across all parcels for subsequent evaluation.

### III Evaluation and Results

Model performance is assessed using the ClassificationMetrics class, which computes key metrics to evaluate the model’s effectiveness on our test dataset.

Metric	Value
Accuracy	0.6845
F1-micro	0.6845
F1-weighted	0.6753
Precision (weighted)	0.8095
Recall (weighted)	0.6845

Table II: Summary performance metrics for the ParcelModel on the Austria 2017 test dataset.

#### Per-Class Performance

A detailed classification report, presented in Table III, provides per-class precision, recall, F1-score and support, mapped to the class names for clarity. The report highlights the model’s performance across the eight crop types, revealing the challenges faced in classification.

Class	Precision	Recall	F1-Score	Support
unknown	0.00	0.11	0.00	18
spring_barley	0.24	0.44	0.31	288
meadow	0.99	0.77	0.87	16,633
winter_wheat	0.70	0.01	0.02	6,398
winter_barley	0.36	0.77	0.49	4,629
winter_rye	0.04	0.01	0.02	586
winter_rapeseed	0.42	0.97	0.59	1,026
corn	0.89	0.95	0.92	9,953
Macro Average	0.45	0.50	0.40	39,531
Weighted Average	0.81	0.68	0.68	39,531

Table III: Per-class classification report for the ParcelModel on the whole Austria 2017 33UVP dataset.

The model achieves an overall accuracy of 0.6845 and a weighted F1-score of 0.6753, indicating moderate generalization to the Austria 2017 dataset. The high weighted precision suggests that when the model predicts a class, it is often correct, but the somewhat lower recall indicates missed instances, particularly for underrepresented classes. The per-class metrics reveal strong performance for meadow and corn, driven by their large support and distinct spectral profiles. However, the unknown class performs poorly, likely due to its small support and heterogeneous nature. Similarly, winter\_wheat and winter\_rye show quite low performance, but we shall talk more about it later on.

## Confusion Matrix Visualization

The confusion matrix provides further insights into classification patterns across the eight crop types.

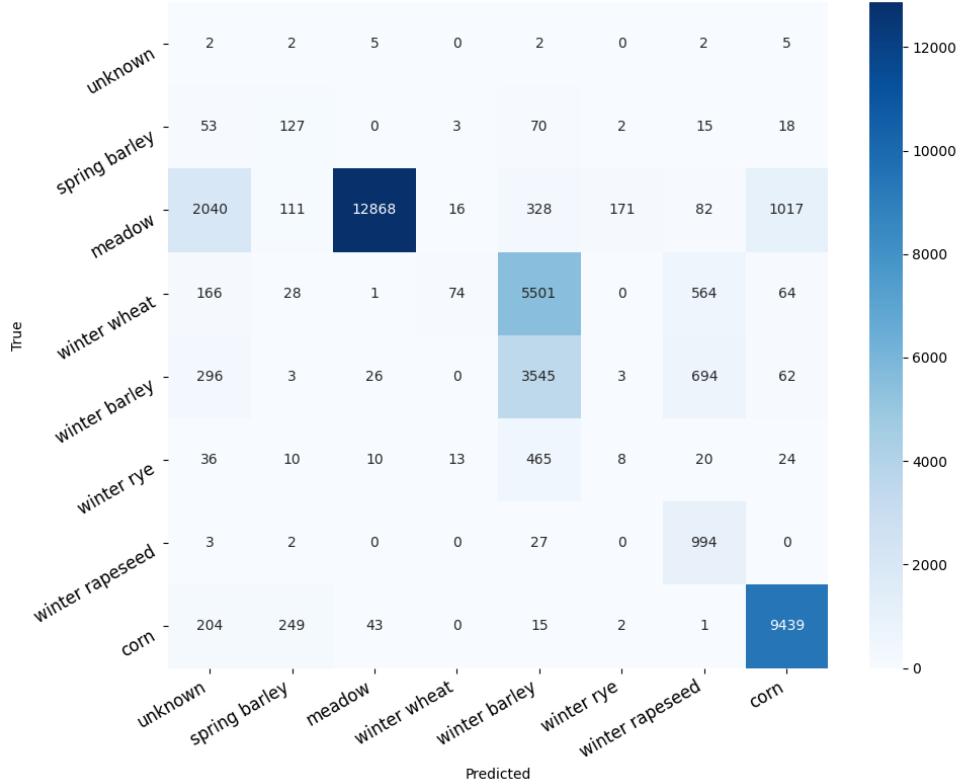


Figure I: Confusion matrix for the ParcelModel on the Austria 2017 test dataset.

The confusion matrix demonstrates robust classification performance for the meadow and corn classes, as evidenced by high values along the diagonal, indicating accurate predictions. In contrast, notable misclassification occurs between winter\_wheat and winter\_barley, likely attributable to their similar spectral signatures and phenological patterns. The unknown class is underrepresented in the dataset and exhibits widespread confusion with multiple classes, reflecting its inherent ambiguity. Additionally, winter\_rye is predominantly misclassified as winter\_barley.

It is critical to highlight that the overall performance of the model is heavily impacted by the mapping and grouping strategy applied when translating German class labels to the English target classes. This approach increases the difficulty of the classification task, as the model must generalize across potentially unseen and heterogeneous data distributions, thereby limiting its ability to distinguish certain classes effectively.

The increased temporal resolution (58 versus 52 time steps) does not materially affect model performance, indicating that the TransformerTimeEncoder effectively manages variable-length sequences. These results corroborate previous findings by Rußwurm and Körner (2020) regarding the resilience of transformer architectures in remote sensing applications. The results validate the model's ability to generalize across regions, despite differences in climate, soil, and observation schedules between Denmark and Austria, highlighting the strength of the ParcelModel architecture proposed by Garnot et al. (2020).

# Bibliography

- [1] Mariana Belgiu and Lucian Drăguț. “Random Forest in Remote Sensing: A Review of Applications and Future Directions”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 114 (2016), pp. 24–31.
- [2] Russell G. Congleton and Robert O. Green. “A Review of Remote Sensing Accuracy and Error Assessment”. In: *Photogrammetric Engineering and Remote Sensing* 57.11 (1991), pp. 1441–1447.
- [3] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv preprint arXiv:1810.04805. 2018.
- [4] Yuval Eldar et al. “Farthest point strategy for progressive image sampling”. In: *IEEE Transactions on Image Processing* 6.9 (1997), pp. 1305–1315.
- [5] Vivien S. F. Garnot et al. “Satellite Image Time Series Classification with Pixel-Set Encoders and Temporal Self-Attention”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.
- [6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.
- [7] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed. Springer, 2009.
- [8] Alfredo Huete et al. “Overview of the radiometric and biophysical performance of the MODIS vegetation indices”. In: *Remote Sensing of Environment* 83.1-2 (2002), pp. 195–213.
- [9] Diederik P Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).
- [10] Ron Kohavi. “A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection”. In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)* 14.2 (1995), pp. 1137–1145.
- [11] Jie Li and Andrew D Heap. “Spatial sampling design for monitoring spatial phenomena”. In: *Geographical Analysis* 47.3 (2015), pp. 207–228.
- [12] J. MacQueen. “Some methods for classification and analysis of multivariate observations”. In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 14. 1967, pp. 281–297.
- [13] Lutz Prechelt. “Early Stopping – But When?” In: *Neural Networks: Tricks of the Trade* (1998), pp. 55–69.
- [14] Marc Rußwurm and Marco Körner. “Self-Attention for Raw Optical Satellite Time Series Classification”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 169 (2020), pp. 421–435.
- [15] Steven K Thompson. *Sampling*. John Wiley & Sons, 2012.
- [16] Ashish Vaswani et al. “Attention is All You Need”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.
- [17] Manzil Zaheer et al. “Deep Sets”. In: *Advances in Neural Information Processing Systems (NeurIPS)* (2017).
- [18] Li Zhang, Yi Wang, and Xi Li. “An entropy-based sampling method for remote sensing image classification”. In: *IEEE Geoscience and Remote Sensing Letters* 13.5 (2016), pp. 631–635.